User Manual

日本語

翻訳: アドバン オートメーション株式会社

HBM Common API 6.1関数マニュアル



HBM Common API

Hottinger Brüel & Kjaer GmbH
Im Tiefen See 45
D-64239 Darmstadt
Tel. +49 6151 803-0
Fax +49 6151 803-9100
Email: info@hbm.com
Internet: www.hbm.com

アドバン オートメーション株式会社
101-0047
東京都千代田区内神田1-9-5 SF内神田ビル 5 F
Tel. 03-5282-7047
Fax. 03-5282-0808
Email: info-advan@adv-auto.co.jp
Internet: https://adv-auto.co.jp/

2024年9月 初版

本書の内容は予告なく変更する場合があります。
すべての製品説明は一般的な情報のみを紹介しています。
これらは製品の品質や耐久性を保証するものではありません。

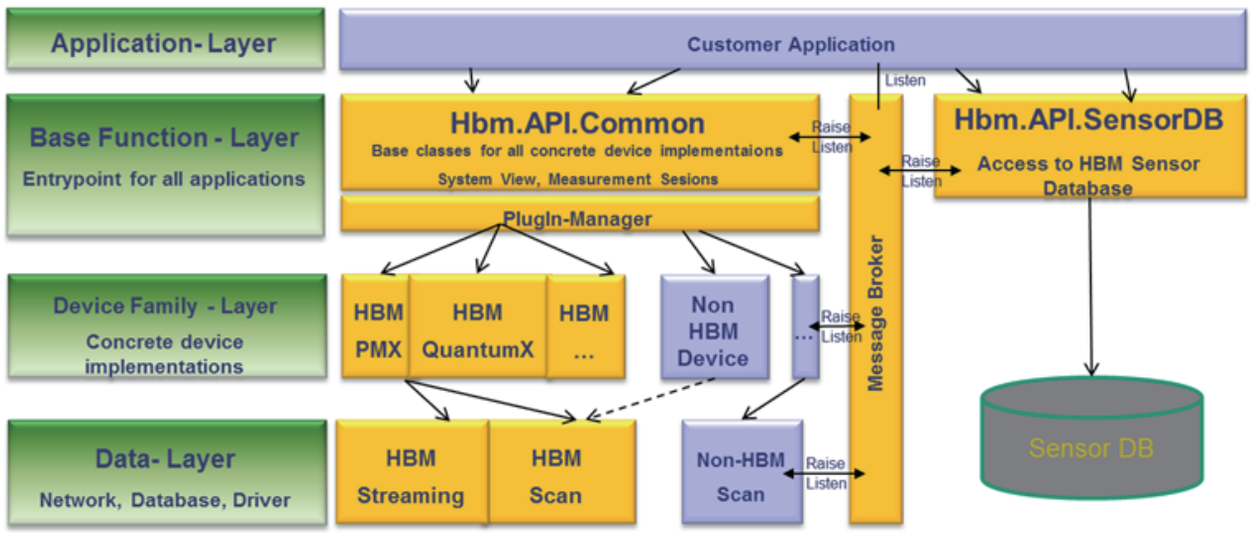
目次

概要 ・・・

概要

Common APIの目的は、さまざまなデバイスをまったく同じ方法でプログラムできるようにすることです。したがって、Hbm.Api.Common、Hbm.Api.SensorDB名前空間、およびHbm.Api.Logging名前空間のタイプを使用し続ける場合、プログラムはサポートされているデバイスで実行されます(QuantumXファミリ、PMX、MGC+のデバイス、またはその他のサポートされているデバイスであるかどうかは関係ありません)。別のデバイスタイプを使用する場合、ソースコードを変更する必要はありません。

完全なシステムアーキテクチャの概要



しかしながら、各デバイスには他のデバイスとは異なる特別な関数(QuantumX、PMX、MGC+)があります。これらの特別なデバイス依存関数にも、このAPI経由でアクセスできます。ソフトウェアでそのような関数を使用すると、コードの汎用性が損なわれることに注意してください。特別な関数を使用すると、ソースコードがデバイスに依存するようになりますが、同時に追加機能への拡張アクセスが可能になります。

例
次の例は、デバイスに依存しないコードとデバイスに依存するコードの違いを示しています。

```
...
// Device independent source code:
// Using only types of the Hbm.API.Common namespace
DaqEnvironment daqEnvironment = DaqEnvironment.GetInstance();
// Scan for devices..."PMX" could also be a variable, that defines which family to scan
List<Device> devices = daqEnvironment.Scan(new List<String> { "PMX" });
// Let myCommonDevice be of type Hbm.API.Common.Entities.Device
Device myCommonDevice=devices[0];
// Connect device
List<Problem> lastProblems; // Used to return warnings and errors
daqEnvironment.Connect(myCommonDevice, out lastProblems);
// Get all signals of the device
List<Signal> mySignals=myCommonDevice.GetAllSignals();
// etc.
...
// Dispose DaqEnvironment before program ends
daqEnvironment.Dispose();
....
```

```
...
// Device ****dependent**** source code:
// Using types of the concrete device: Here Hbm.Api.PMX namespace
DaqEnvironment daqEnvironment = DaqEnvironment.GetInstance();;
// Scan for devices..."PMX" could also be a variable, that defines which family to scan
List<Device> devices = daqEnvironment.Scan(new List<String> { "PMX" });
// Let myPMXDevice be of type Hbm.API.PMX.PmxDevice
PmxDevice myPMXDevice=devices[0] as PmxDevice;// **** Cast to a concrete device object ****
// Connect device
List<Problem> lastProblems; // Used to return warnings and errors
daqEnvironment.Connect(myPMXDevice, out lastProblems);
// Get all signals of the device
List<Signal> mySignals=myPMXDevice.GetAllSignals();
// Access an ADDITIONAL FEATURE of PMX device, which is not visible or accessible via type
Device!
myPmxDevice.AdditionalFeatures.LoadParameterSet(1); // **** Usage of an additional feature
****
...
// Dispose DaqEnvironment before program ends
daqEnvironment.Dispose();
....
```

要件

Hbm.CommonAPIの現在のバージョンを使用するには、PCが次の最小要件を満たしている必要があります:

- Windows® 7以降(32-Bitまたは64-Bit)
- Intel Pentium 1GHzまたは同等のプロセッサ
- 1024MBのメインメモリ(RAM)
- 解像度1024x768ピクセルのグラフィックスカード
- ハードディスク上に約45MBの空き容量
- QuantumXデバイスを接続するためのイーサネットインタフェース(10/100MBit)またはFireWire

Hbm.CommonAPI for .Netを使用してプログラムを構築するための前提条件:

- Microsoft Windows® 7以降
- Microsoft Visual Studio(.Net Framework 4.0以降)
- HBM DLL(Hbm.Api.Common.dll、Hbm.Api.Scan.dll、Hbm.Api.SensorDB.dll、Hbm.Api.Logging.dll、Hbm.Api.Utils.dll、HBM_Streaming.dll、HBM_Scan.dll、HBM_Update .dll、HBMTedsDll.dll、IBR.XPDF.dll、ICSharpCode.SharpZipLib.dll、Gibraltar.Agent.dll、NLog.dll、VistaDB.5.NET40.dll)
- プログラムで使用するHBMデバイスの DLL(およびその他のファイル)
  - PMXデバイスの場合: Hbm.Api.Pmx.dll、PmxSkills.json
  - QuantumXデバイスの場合: Hbm.Api.QuantumX.dll、HBM_QX_Framework.dll、XPDF.dll、QuantumXSkills.json
  - MGCデバイスの場合: Hbm.Api.Mgc.dll、MgcSkills.json
- DeviceDriver.plugins(使用するデバイスドライバを定義するXMLファイル)
- センサデータベースファイル - 例: HBMセンサデータベース(HBMSensorDatabase.vdb5)およびユーザセンサデータベース(アプリケーションで使用する場合)
- 完全なTEDSDefinitionディレクトリ(default.tdl、hbm.tdl、ManufactureList.xdl、model001f.xdl、propertylist.xdl、SensorTdlRelations.xml、およびSpecialUnitRelations.xmlファイルを含む)。

National InstrumentsのLabVIEW®内からHbm.API を使用するための前提条件:

- Windows® 7以降(32-Bitまたは64-Bit)
- LabVIEW 2012以降
- HBM DLL (Hbm.Api.Common.dll、Hbm.Api.Scan.dll、Hbm.Api.SensorDB.dll、Hbm.Api.Logging.dll、Hbm.Api.Utils.dll、Hbm.LicenseManager.dll、 HBM_Streaming.dll、HBM_Scan.dll、HBM_Update.dll、HBMTedsDll.dll、IBR.XPDF.dll、ICSharpCode.SharpZipLib.dll、Gibraltar.Agent.dll、NLog.dll、VistaDB.5.NET40.dll)
- プログラムで使用するHBMデバイスのDLL
  - PMXデバイスの場合: Hbm.Api.Pmx.dll、PmxSkills.json
  - QuantumXデバイスの場合: Hbm.Api.QuantumX.dll、HBM_QX_Framework.dll、XPDF.dll、QuantumXSkills.json
  - MGCデバイスの場合: Hbm.Api.Mgc.dll、MgcSkills.json
- DeviceDriver.plugins(使用するデバイスドライバを定義するXMLファイル)
- センサデータベースファイル - 例: HBMセンサデータベース(HBMSensorDatabase.vdb5)およびユーザセンサデータベース(アプリケーションで使用する場合)
- 完全なTEDSDefinitionディレクトリ(default.tdl、hbm.tdl、ManufactureList.xdl、model001f.xdl、propertylist.xdl、SensorTdlRelations.xml、およびSpecialUnitRelations.xmlファイルを含む)。
- LabVIEW 2012未満でHBM.CommonApiを使用するには、LabVIEW.exeと同じフォルダ内に次の内容のファイル(LabVIEW.exe.config)を作成する必要があります。

```
<configuration>
  <startup useLegacyV2RuntimeActivationPolicy="true">
    <supportedRuntime version="v4.0.30319"/>
  </startup>
</configuration>
```

Hbm.CommonAPIの関数とデータ型を使用するプログラムを配布するには、次のことを行う必要があります。

- ターゲットシステムに.NET Framework 4.0をインストールします。
- 次のHBM DLLをプログラムと一緒にターゲットシステムにコピーします:
  - HBM DLL(Hbm.Api.Common.dll、Hbm.Api.Scan.dll、Hbm.Api.SensorDB.dll、Hbm.Api.Logging.dll、Hbm.Api.Utils.dll、Hbm.LicenseManager.dll、 HBM_Streaming.dll、HBM_Scan.dll、HBM_Update.dll、HBMTedsDll.dll、IBR.XPDF.dll、ICSharpCode.SharpZipLib.dll、Gibraltar.Agent.dll、NLog.dll、VistaDB.5.NET40.dll)
- プログラムで使用するHBMデバイスのDLL
  - PMXデバイスの場合: Hbm.Api.Pmx.dll、PmxSkills.json
  - QuantumXデバイスの場合: Hbm.Api.QuantumX.dll、HBM_QX_Framework.dll、XPDF.dll、QuantumXSkills.json
  - MGCデバイスの場合: Hbm.Api.Mgc.dll、MgcSkills.json
  - DeviceDriver.plugins(使用するデバイスドライバを定義するXMLファイル)
  - センサデータベースファイル - 例: HBM センサデータベース(HBMSensorDatabase.vdb5)およびユーザセンサデータベース(アプリケーションで使用する場合)
- 完全なTEDSDefinitionディレクトリ(default.tdl、hbm.tdl、ManufactureList.xdl、model001f.xdl、propertylist.xdl、SensorTdlRelations.xml、およびSpecialUnitRelations.xmlファイルを含む)。
- ファイアウォールを変更します。
  - スキャンが機能するには、UDP受信ポート31416および31417がアプリケーションに対して開かれている必要があります。通常、システムはアプリケーションにネットワークの使用を許可するかどうかを尋ねますが、その後、このアプリケーションに対するすべての受信通信が許可されます。
  - Windows® 7以降、これはコマンドラインで実行できます。

```
netsh advfirewall firewall add rule name="AppName" direction=in action=allow enable=yes
profile=any localport=31416,31417 protocol=UDP edge=yes program="AppExe"
```

プログラムをテストするには、サポートされているデバイス(PMX、QuantumX、MGCなど)がPCと同じネットワークセグメントに正しく接続されていること、およびPCのファイアウォールがデバイスへの通信をブロックしていないことを確認してください。

重要なリリースノート

新機能

前回のリリース以降に追加された、最も関連性の高い機能、オブジェクト、プロパティ、および関数をここに示します:

- ファイバブラッググレーティング(Fiber Bragg Grating: FBG)に基づく光デバイスがサポートされるようになりました(MXFS)
- FBGのコンテキストにさまざまなオブジェクトと関数が追加されました(FbgConnector、FbgChannel、FbgSignal、FbgSpectrum、FbgBandsDetectionResultなど)。
- FBGをサポートするために追加された新しいセンサタイプ(センサデータベースには含まれていません)。
- CAN Rawデータ取得に関する各種オブジェクトを追加(CanRawChannel、CanRawSignalなど)。
- 非同期信号がサポートされるようになりました(現在、非同期信号はCanRawSignalのみです)。
- 任意のデータ型を計測値として(CanRawMessagesなど)を使用できます(これまではdoubleのみをサポート)。
- 同期信号と非同期信号が混在したデータ収集をサポートします。
- MGCは接続中に風袋値を0.0にリセットしなくなりました。MGCデバイスの風袋値を設定/取得するには、SetTareValueおよびGetTareValue関数を使用してください。

デバイスの新機能とFBGに関連するその他のオブジェクト:

デバイスオブジェクト内にいくつかの新しい関数が追加されており、光学計測デバイスと対話するための多数の新しいオブジェクトも追加されています:

- IsReadingFbgSpectrumSupported()を介して、デバイスがFBG関連関数をサポートしているかどうかを確認します(デバイスから複数のスペクトルを読み取るなど)。
- ReadFbgSpectrum()を呼び出して、デバイスの特定のFbgSpectrumTypeのFbgSpectrumを取得します。
- SetFbgDetectionThreshold()を使用して、FbgConnector上のFbgChannelを定義する接続されたFbgSensorの検出に使用する検出しきい値を設定します。
- AutoDetectFbgBands()を実行して、特定のFbgConnectorのFbgChannelについて検出されたFbgBandDetailsを記述するFbgBandsDetectionResultを取得します。
- SetFbgBandsDetectionResult()を使用して、自動検出または変更されたFbgBandsDetectionResultをFbgConnectorに割り当てます。
- SetFbgChannelDetectionRange()で特定のFbgChannelの検出範囲を設定します。
- SetFbgChannelActivation()を使用して、特定のFbgChannelをアクティブまたは非アクティブにします。

以下のFBG関連センサが追加されました:

- FbgAccelerometerSensor
- FbgGenericSensor
- FbgStrainGageSensor
- FbgTemperatureSensor
- FbgWavelengthSensor

これらのセンサはまだセンサデータベースに含まれていないことに注意してください。これらのセンサタイプを保存したり、センサデータベースからそのタイプを読み取ることはできません。現時点ではオンザフライで作成し、チャネルに割り当てることができます。

非同期信号

非同期信号によるデータ取得のコンテキストでは、次の問題が重要です:

- 非同期信号(サンプル周波数が固定されていない信号)は、同期信号(複数のデバイスおよびデバイスファミリ上で同期)と一緒に計測できます。
- このような信号の計測値には、同期信号データの場合と同じ方法でアクセスすることができ、さらに任意のタイプの一般的な計測データにアクセスすることもできます。
- 計測開始時の同期中、非同期データは考慮されません。つまり、利用可能な非同期計測データがない場合でも、計測は開始されます。
- 非同期信号のコンテキストではタイムアウトはありません(計測開始時も計測中も)。
- 計測に関与する非同期信号のみがあり、計測開始時の最初の100ミリ秒以内に利用可能なデータがない場合、現在のPC時間が計測の開始時間を示すために使用されます。

CAN Rawのデータ取得をサポート

CommonAPI V6.0以前は、Signal.ContinuousMeasurementValues経由でdouble型の計測値のみが利用できましたが、計測値自体のデータ型に関してより柔軟になるように変更されました。doubleの場合のデータへのアクセスは依然として同じですが、Signal.ContinuousMeasurementValuesの基になる型は、一般的な計測値を保持できるように変更されています(MeasurementValuesBaseを参照)。新しいプロパティGenericMeasurementValuesが追加され、あらゆるタイプの計測値にアクセスできるようになりました(ただし、これまではCanRawSignalsのCanRawMessagesのみで実現していました)。

Signal.GenericMeasurementValuesにアクセスするには、信号をインタフェースIGenericSignalにキャストする必要があります。CanRawSignalが配信するGenericMeasurementValues にアクセスするには、CanRawSignalのGenericMeasurementValuesをMeasurementValuesBase<CanRawMessage>にキャストする必要があります。

したがって、CanRawSignalsの計測値(CanRawMessages、Timestamps、States、UpdatedValueCountなど)にアクセスするには、次のようにキャストする必要があります:

(((myCanRawSignal as IGenericSignal). GenericMeasurementValues**) as** MeasurementValuesBase**<CanRawMessage>).**Values**[x].**Header **or .**Timestamps**[x]. or .**States**[x], etc**

DemoProjectには、その方法を示すサンプルコードがあります。

Signal.ContinuousMeasurementValuesを介して計測値を取得する"通常の"方法との一貫性を保つために、CanRawMessagesはdouble値としても使用できます。これにより、単一のCanRawMessageは10個のdouble値で表されます(これらはすべてまったく同じタイムスタンプを持ちます)。これは、APIを使用してプロジェクトで計測データを保存する方法を変更する必要がないことを示しています。理解を深めるために、CanRawSignalのドキュメントを参照し、CommonAPIに付属するDemoProjectの関連タブも確認してください。

Common APIがサポートしている関数の概要。
緑色のチェック マーク(✔)は、バージョン6.0で新しく追加された機能を示します。

機能	QuantumX/ SomatXR	PMX	MGCplus
Device Scan	✔	✔	✔(CP52のみ)
Measurement configuration	✔	✔	✔
Calibration certificates	✔	✔	利用不可
TimeSource configuration	✔	✔	✔
Sensor configuration	✔	✔	✔
Bridge	✔	✔	✔
CanBus	✔	利用不可	✔
Counter	✔	✔	✔
Current	✔	✔	✔
CurrentProbe	利用不可	利用不可	利用不可
FbgAccelerometerSensor	✔(MXFSのみ)	利用不可	利用不可
FbgGenericSensor	✔(MXFSのみ)	利用不可	利用不可
FbgStrainGageSensor	✔(MXFSのみ)	利用不可	利用不可
FbgTemperatureSensor	✔(MXFSのみ)	利用不可	利用不可
FbgWavelengthSensor	✔(MXFSのみ)	利用不可	利用不可
Frequency	✔	✔	✔
IEPE	✔	利用不可	✔
LVDT	✔	✔	✔
PiezoPassive	利用不可	利用不可	✔
Potentiometer	✔	✔	✔
Pt	✔	利用不可	✔
PWM	✔	✔	✔
Resistance	✔	利用不可	✔
SSI	✔	✔	利用不可
ThermoCouple		利用不可	✔
Voltage	✔	✔	✔
VoltageProbe	利用不可	利用不可	利用不可
Analog In DAQ	✔	✔	✔
Analog Out (direct setting)	✔	✔	✔
Analog Out (channel routing)	✔	✔	✔
Digital In DAQ	✔	✔	✔
Digital Out DAQ	✔	✔	✔
Digital Out (direct setting)	✔	✔	✔
CAN DAQ	✔	✔(CODESYS/calc.チャネル経由)	✔
CAN Raw-DAQ	✔	利用不可	利用不可
Optical DAQ	✔	利用不可	利用不可

✔: Common APIバージョン5.0でサポート
✔: Common APIバージョン6.0以降に対応

サポート

サポートや詳細情報が必要な場合は、最寄りのHBM営業担当者または最寄りのサポートホットラインにお問い合わせください。

E-mail support: support@hbm.com
- Telephone support: +49 6151 803-0
- HBM on the Internet: http://www.hbm.com
- Hottinger Brüel & Kjaer BK (http://www.hbkworld.com)

例

これは、関連するすべての例の概要です。すべての例はコードスニペットであり、周囲に何もないと機能しません。少なくとも、それらを関数に配置し、必要な名前空間を追加する必要があります。詳細なコードスニペットは、Common API概要マニュアルに記載されています。

- デバイスに依存するコードと依存しないコードの違い
- デバイスをスキャンして接続し、センサを特定のチャンネルに割り当てる方法
- デバイス間の同期を設定する方法
- デバイスをスキャンして接続し、連続計測を実行する方法
- デバイスの特殊機能の使い方
- HBMセンサデータベースの特定のセンサを取得する方法
- ユーザセンサデータベースのセンサを挿入、更新、削除する方法
- 提供されたロギングメカニズムの使用方法
- 状態監視からのイベント受信方法
- 非同期計測値として送信されるCAN Rawメッセージにアクセスする方法

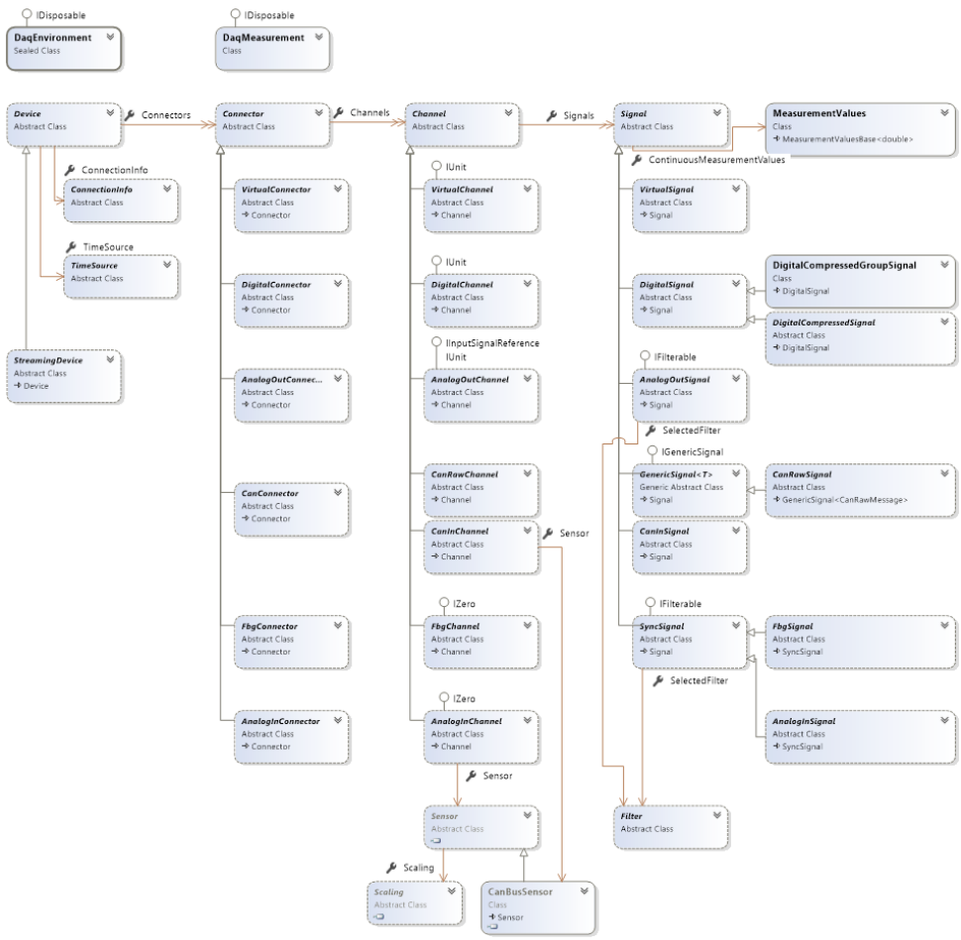
Hbm.Api.DemoProjectを参照することを強くお勧めします。このサンプルプロジェクトを(トップダウンで)確認することで、必要なすべての手順を示します:

- デバイスをスキャン
- デバイスを接続
- デバイスにプロパティを割り当て
- 計測を設定
- 計測値を取得する
- HBMおよびユーザセンサデータベースを使用
- 提供されたロギングメカニズムを使用
- さまざまなスペクトルタイプの光デバイスを読み取り(ファイバーブラッググレーティング関連)
- 取り付けられた光学センサのバンドを自動検出
- 検出された(および適応された)帯域を光学デバイスに割り当てる
- CAN Rawメッセージへのアクセス

Hbm.Api.DemoProjectはHBM.CommonAPIパッケージの一部です。

HBM Common APIドキュメント

トピックス

名前	説明
Hbm.Api.Common Namespace	Hbm.Api.Common namespaceには、抽象クラスHbm.Api.Common.Entities.Deviceから派生した、任意のデバイスで計測を設定および実行する関数が含まれています。 • DaqEnvironmentを使用してデバイスをスキャンして接続 • デバイス、コネクタ、チャネル、センサ、信号オブジェクトを使用して、接続されたデバイスを設定/パラメータ化 • DaqMeasurementを使用して、これらのデバイス/信号で計測を実行 次の図は、この名前空間の関連クラスとその最も重要なプロパティの要約を示しています: 
Hbm.Api.SensorDB Namespace	Common APIのバージョン3.0より、センサデータベースへのアクセスがサポートされました。センサとスケーリングタイプの定義に加えて、この名前空間にはSensorDBManagerオブジェクトが含まれています。このオブジェクトは、センサの取得、挿入、更新、削除を可能にするISensorDBインタフェースを実装し、センサデータベース内のセンサの階層カテゴリ(グループ)をさまざまな言語で管理することを可能にします。 APIには、ほとんどのHBMセンサとユーザデータベースの空のテンプレートを含むHBMセンサデータベースが含まれています。Hbm.Api.DemoProjectは、これらのデータベースにアクセスする方法を示しています。 **注意:** **HBMセンサデータベースは変更しないでください**。変更したい場合...参照

名前	説明
Hbm.Api.Logging Namespace	Hbm.Api.Logging Namespaceには、さまざまなログフレームワークを使用してメッセージをログに記録できるようにする関数が含まれています(これまではNLogとLoupeをサポートしていました)。 コンテキスト(データベース アクセス、デバイス通信など)に従ってログ エントリをグループ化するには、さまざまなLogContextを定義して、ログエントリが挿入される階層構造を構築できます。 ロギングの一般的な使用方法は次のとおりです: • 特定のLoggingFrameworkと特定のLogLevelを選択して、ログ記録を開始します。 • LogManager.CreateLoggerを使用して、少なくとも1つのロガーオブジェクト(Ilogger)を作成します。 • ログエントリを分類するために必要な数のLogContextを作成します。 • 特定の関数を使用してログエントリを生成します...参照
Supported devices	

Hbm.Api.Common Namespace

目次

Hbm.Api.Common Namespace ・・・

RemoveSignals Method ・・・

VirtualChannel Members ・・・

CanConnector Class ・・・

NotSupportedFilter.NotSupportedFilter Constructor ・・・

Problem Constructor ・・・

TedsReadingWarning Class ・・・

ValueNotSetError.ValueNotSetError Constructor (Device, Connector, SettingType, string, string) • • • • • • • • • • • • 490
ValueNotSetError.ValueNotSetError Constructor (Device, Sensor, SettingType, string, string) • • • • • • • • • • • • 491
ValueNotSetError.ValueNotSetError Constructor (Device, SettingType, string, string) • • • • • • • • • • • • • • • • 492
ValueNotSetError.ValueNotSetError Constructor (Device, Signal, SettingType, string, string) • • • • • • • • • • • • 493
ValueNotSetError Properties • 494
ValueNotSetError.Message Property • 495
ValueOutOfRangeError Class • 496
ValueOutOfRangeError Members • 497
ValueOutOfRangeError Constructor • 498
ValueOutOfRangeError.ValueOutOfRangeError Constructor (Device, Channel, SettingType, string, string, string, string) • • 499
ValueOutOfRangeError.ValueOutOfRangeError Constructor (Device, Connector, SettingType, string, string, string, string) • 500
ValueOutOfRangeError.ValueOutOfRangeError Constructor (Device, Sensor, SettingType, string, string, string, string) • • 501
ValueOutOfRangeError.ValueOutOfRangeError Constructor (Device, SettingType, string, string, string, string) • • • • • 502
ValueOutOfRangeError.ValueOutOfRangeError Constructor (Device, Signal, SettingType, string, string, string, string) • • • 503
ValueOutOfRangeError Properties • 504
ValueOutOfRangeError.MaxPossibleValue Property • 505
ValueOutOfRangeError.Message Property • 506
ValueOutOfRangeError.MinPossibleValue Property • 507
Warning Class • 508
Warning Members • 509
Warning Constructor • 510
Warning.Warning Constructor (Device, Channel, SettingType, string) • 511
Warning.Warning Constructor (Device, Connector, SettingType, string) • 512
Warning.Warning Constructor (Device, Sensor, SettingType, string) • 513
Warning.Warning Constructor (Device, SettingType, string) • 514
Warning.Warning Constructor (Device, Signal, SettingType, string) • 515
Warning Properties • 516
Warning.Message Property • 517
Hbm.Api.Common.Entities.Scalings Namespace • 518
NotSupportedOutputScaling Class • 519
NotSupportedOutputScaling Members • 520
NotSupportedOutputScaling Properties • 521
NotSupportedOutputScaling.ScalingType Property • 522
OutputScaling Class • 523
OutputScaling Members • 524
OutputScaling Properties • 525
OutputScaling.OutUnit Property • 526
OutputScaling.ScalingType Property • 527
OutputScaling.SourceUnit Property • 528
OutputScalingPoint Class • 529
OutputScalingPoint Members • 530
OutputScalingPoint Constructor • 531
OutputScalingPoint.OutputScalingPoint Constructor () • 532
OutputScalingPoint.OutputScalingPoint Constructor (decimal, decimal) • 533
OutputScalingPoint Properties • 534
OutputScalingPoint.Out Property • 535
OutputScalingPoint.Source Property • 536
OutputTableScaling Class • 537
OutputTableScaling Members • 538
OutputTableScaling Constructor • 539
OutputTableScaling.OutputTableScaling Constructor () • 540
OutputTableScaling.OutputTableScaling Constructor (TableScaling) • 541
OutputTableScaling Properties • 542
OutputTableScaling.ScalingPoints Property • 543
OutputTableScaling.ScalingType Property • 544
Hbm.Api.Common.Entities.Signals Namespace • 545
AnalogInSignal Class • 546
AnalogOutSignal Class • 547
AnalogOutSignal Members • 548
AnalogOutSignal.AnalogOutSignal Constructor • 549
AnalogOutSignal Properties • 550
AnalogOutSignal.Scaling Property • 551
AnalogOutSignal.SelectedFilter Property • 552

CanInSignal Class ・・・ 553
CanRawSignal Class ・・・ 554
DigitalCompressedGroupSignal Class ・・・ 557
DigitalCompressedGroupSignal Members ・・・ 558
DigitalCompressedGroupSignal.DigitalCompressedGroupSignal Constructor ・・・ 560
DigitalCompressedGroupSignal Methods ・・・ 561
DigitalCompressedGroupSignal.GetUniqueID Method ・・・ 562
DigitalCompressedGroupSignal Properties ・・・ 563
DigitalCompressedGroupSignal.DigitalCompressedSignals Property ・・・ 564
DigitalCompressedGroupSignal.Group Property ・・・ 565
DigitalCompressedSignal Class ・・・ 566
DigitalCompressedSignal Members ・・・ 567
DigitalCompressedSignal Methods ・・・ 569
DigitalCompressedSignal.DecompressMeasurementValues Method ・・・ 570
DigitalCompressedSignal Properties ・・・ 571
DigitalCompressedSignal.Group Property ・・・ 572
DigitalCompressedSignal.MaskedBit Property ・・・ 573
DigitalCompressedSignal.MaskedBitValue Property ・・・ 574
DigitalSignal Class ・・・ 575
FbgSignal Class ・・・ 576
GenericSignal Class ・・・ 577
GenericSignal Members ・・・ 578
GenericSignal.GenericSignal Constructor ・・・ 580
GenericSignal Methods ・・・ 581
GenericSignal.AddValuesToCircularBuffer Method ・・・ 582
GenericSignal.CopyAllValuesFromCircularBufferToMeasurementValues Method ・・・ 583
GenericSignal.CreateGenericCircularBuffer Method ・・・ 584
GenericSignal.FreeGenericCircularBuffer Method ・・・ 585
GenericSignal.InitGenericMeasurementValues Method ・・・ 586
GenericSignal.SetGenericMeasurementValues Method ・・・ 587
GenericSignal Properties ・・・ 588
GenericSignal.GenericCircularBuffer Property ・・・ 589
Signal Class ・・・ 590
Signal Members ・・・ 591
Signal.Signal Constructor ・・・ 593
Signal Methods ・・・ 594
Signal.CreateCircularBuffer Method ・・・ 595
Signal.FreeCircularBuffer Method ・・・ 596
Signal.GetSingleMeasurementValue Method ・・・ 597
Signal.GetUniqueID Method ・・・ 598
SetMeasurementValues Method ・・・ 599
Signal.SetMeasurementValues Method (double[], double[], MeasurementValueState[]) ・・・ 600
Signal.SetMeasurementValues Method (double[], double[], MeasurementValueState[], int) ・・・ 601
Signal.SetMeasurementValues Method (List<double>, List<double>, List<MeasurementValueState>) ・・・ 602
Signal.SetSingleMeasurementValue Method ・・・ 603
Signal Properties ・・・ 604
Signal.ContinuousMeasurementValues Property ・・・ 605
Signal.ExcludeFromDaqTimeout Property ・・・ 606
Signal.HasSampleRate Property ・・・ 607
Signal.IdleTime Property ・・・ 608
Signal.IsExcludedFromDaq Property ・・・ 609
Signal.IsMeasurable Property ・・・ 610
Signal.Name Property ・・・ 611
Signal.SampleRate Property ・・・ 612
Signal.SynchronousMode Property ・・・ 613
SyncSignal Class ・・・ 614
SyncSignal Members ・・・ 615
SyncSignal Properties ・・・ 616
SyncSignal.SelectedFilter Property ・・・ 617
SyncSignal.UseAutoFilter Property ・・・ 618
VirtualSignal Class ・・・ 619
IFilterable Interface ・・・ 620
IFilterable Members ・・・ 621

IFilterable Properties ・・・

FbgBandsDetectionResult Class ・・・

MeasurementValuesBase.BufferSize Property ・・・

Device.ChangeIp Method ・・・

Device.SetAnalogOutChannelActivation Method ・・・

Device.SyncMode Property ・・・

FirmwareUpdater.Dispose Method ・・・

CommunicationFailedException Constructor ・・・

FamilyNameDuplicateException.FamilyNameDuplicateException Constructor (string, Exception) ・・・・・・・・・・・・ 1145
FamilyNameNotSetException Class ・・ 1146
FamilyNameNotSetException Members ・・ 1147
FamilyNameNotSetException Constructor ・・ 1148
FamilyNameNotSetException.FamilyNameNotSetException Constructor () ・・・・・・・・・・・・・・・・・・・・・・・・ 1149
FamilyNameNotSetException.FamilyNameNotSetException Constructor (string) ・・・・・・・・・・・・・・・・・・・・・ 1150
FamilyNameNotSetException.FamilyNameNotSetException Constructor (string, Exception) ・・・・・・・・・・・・・・・ 1151
ParameterizationNotCompletedException Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1152
ParameterizationNotCompletedException Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1153
ParameterizationNotCompletedException Constructor ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1154
ParameterizationNotCompletedException.ParameterizationNotCompletedException Constructor () ・・・・・・・・・・・・ 1155
ParameterizationNotCompletedException.ParameterizationNotCompletedException Constructor (string) ・・・・・・・・・ 1156
ParameterizationNotCompletedException.ParameterizationNotCompletedException Constructor (string, Exception) ・・・・ 1157
PrepareDaqFailedException Class ・・ 1158
PrepareDaqFailedException Members ・・ 1159
PrepareDaqFailedException Constructor ・・ 1160
PrepareDaqFailedException.PrepareDaqFailedException Constructor () ・・・・・・・・・・・・・・・・・・・・・・・・ 1161
PrepareDaqFailedException.PrepareDaqFailedException Constructor (string) ・・・・・・・・・・・・・・・・・・・・・ 1162
PrepareDaqFailedException.PrepareDaqFailedException Constructor (string, Exception) ・・・・・・・・・・・・・・・ 1163
StreamingInitializationFailedException Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1164
StreamingInitializationFailedException Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1165
StreamingInitializationFailedException Constructor ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1166
StreamingInitializationFailedException.StreamingInitializationFailedException Constructor () ・・・・・・・・・・・・ 1167
StreamingInitializationFailedException.StreamingInitializationFailedException Constructor (string) ・・・・・・・・・・ 1168
StreamingInitializationFailedException.StreamingInitializationFailedException Constructor (string, Exception) ・・・・・・ 1169
StreamingInvalidDataException Class ・・・ 1170
StreamingInvalidDataException Members ・・ 1171
StreamingInvalidDataException Constructor ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1172
StreamingInvalidDataException.StreamingInvalidDataException Constructor () ・・・・・・・・・・・・・・・・・・・・ 1173
StreamingInvalidDataException.StreamingInvalidDataException Constructor (string) ・・・・・・・・・・・・・・・・・ 1174
StreamingInvalidDataException.StreamingInvalidDataException Constructor (string, Exception) ・・・・・・・・・・・・ 1175
StreamingTimeoutException Class ・・・ 1176
StreamingTimeoutException Members ・・ 1177
StreamingTimeoutException Constructor ・・ 1178
StreamingTimeoutException.StreamingTimeoutException Constructor () ・・・・・・・・・・・・・・・・・・・・・・・・ 1179
StreamingTimeoutException.StreamingTimeoutException Constructor (string) ・・・・・・・・・・・・・・・・・・・・・ 1180
StreamingTimeoutException.StreamingTimeoutException Constructor (string, Exception) ・・・・・・・・・・・・・・・ 1181
TooManySampleRatesException Class ・・ 1182
TooManySampleRatesException Members ・・ 1183
TooManySampleRatesException Constructor ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1184
TooManySampleRatesException.TooManySampleRatesException Constructor () ・・・・・・・・・・・・・・・・・・・・・ 1185
TooManySampleRatesException.TooManySampleRatesException Constructor (string) ・・・・・・・・・・・・・・・・・・ 1186
TooManySampleRatesException.TooManySampleRatesException Constructor (string, string, string, string, int) ・・・・・・ 1187
TooManySampleRatesException Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1188
TooManySampleRatesException.DeviceFamily Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1189
TooManySampleRatesException.DeviceName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1190
TooManySampleRatesException.DeviceUniqueId Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1191
TooManySampleRatesException.MaxSupprtedSampleRates Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 1192
UnknownChannelException Class ・・ 1193
UnknownChannelException Members ・・ 1194
UnknownChannelException Constructor ・・ 1195
UnknownChannelException.UnknownChannelException Constructor () ・・・・・・・・・・・・・・・・・・・・・・・・ 1196
UnknownChannelException.UnknownChannelException Constructor (string) ・・・・・・・・・・・・・・・・・・・・・ 1197
UnknownChannelException.UnknownChannelException Constructor (string, Exception) ・・・・・・・・・・・・・・・ 1198
UnknownConnectorException Class ・・・ 1199
UnknownConnectorException Members ・・・ 1200
UnknownConnectorException Constructor ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1201
UnknownConnectorException.UnknownConnectorException Constructor () ・・・・・・・・・・・・・・・・・・・・・・ 1202
UnknownConnectorException.UnknownConnectorException Constructor (string) ・・・・・・・・・・・・・・・・・・・ 1203
UnknownConnectorException.UnknownConnectorException Constructor (string, Exception) ・・・・・・・・・・・・・・ 1204
UnknownFilterException Class ・・・ 1205
UnknownFilterException Members ・・・ 1206
UnknownFilterException Constructor ・・・ 1207

UnknownFilterException.UnknownFilterException Constructor () ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1208
UnknownFilterException.UnknownFilterException Constructor (string) ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1209
UnknownFilterException.UnknownFilterException Constructor (string, Exception) ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1210
UnknownSignalException Class ・・・・・・・・・・・・・・・・・・・

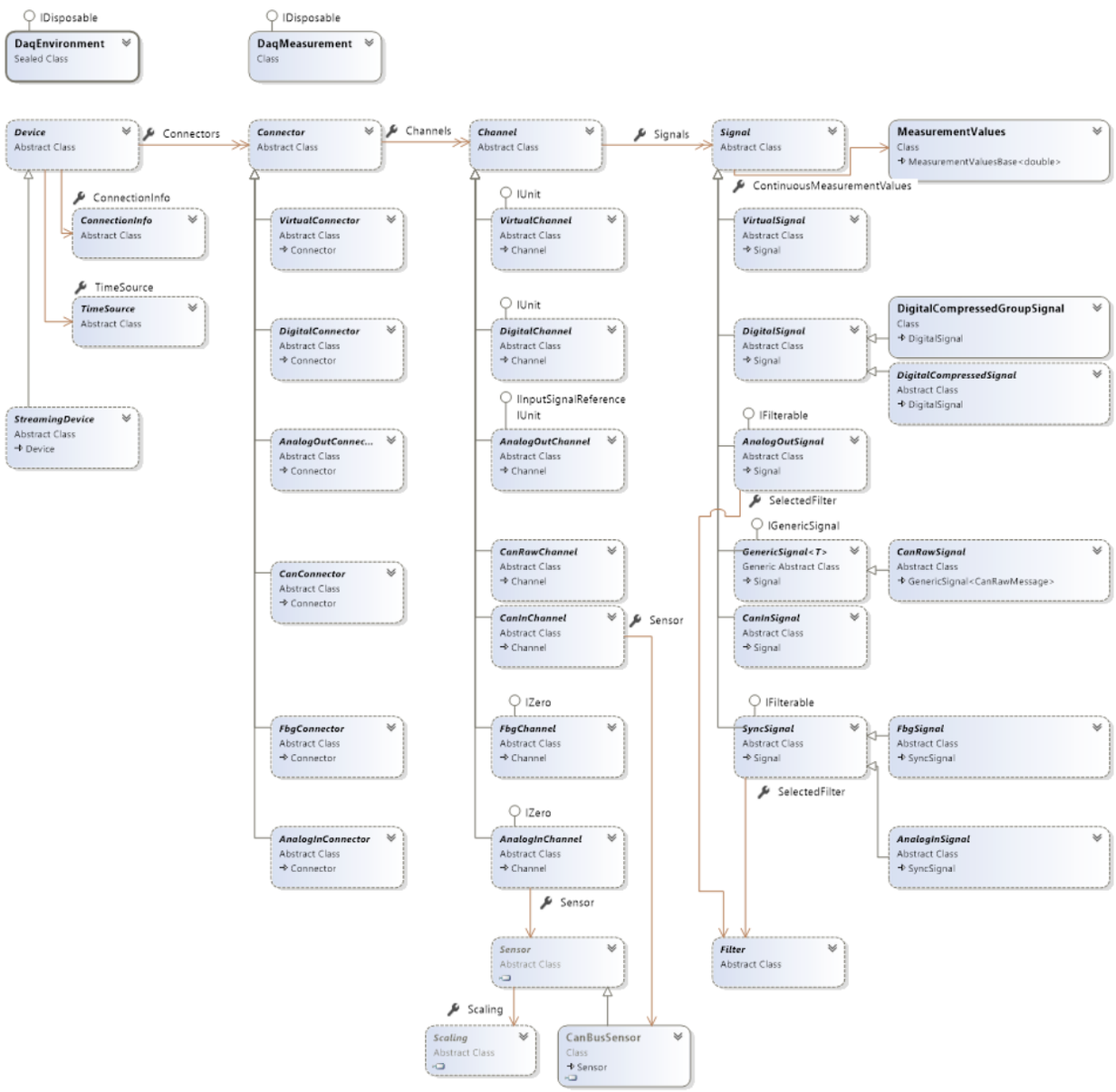
SignalFilterChange Class • • •

MessageBroker Class ・・・

Hbm.Api.Common namespaceには、抽象クラスHbm.Api.Common.Entities.Deviceから派生した、任意のデバイスで計測を設定および実行する関数が含まれています。

- DaqEnvironmentを使用してデバイスをスキャンして接続
- デバイス、コネクタ、チャネル、センサ、信号オブジェクトを使用して、接続されたデバイスを設定/パラメータ化
- DaqMeasurementを使用して、これらのデバイス/信号で計測を実行

次の図は、この名前空間の関連クラスとその最も重要なプロパティの要約を示しています:

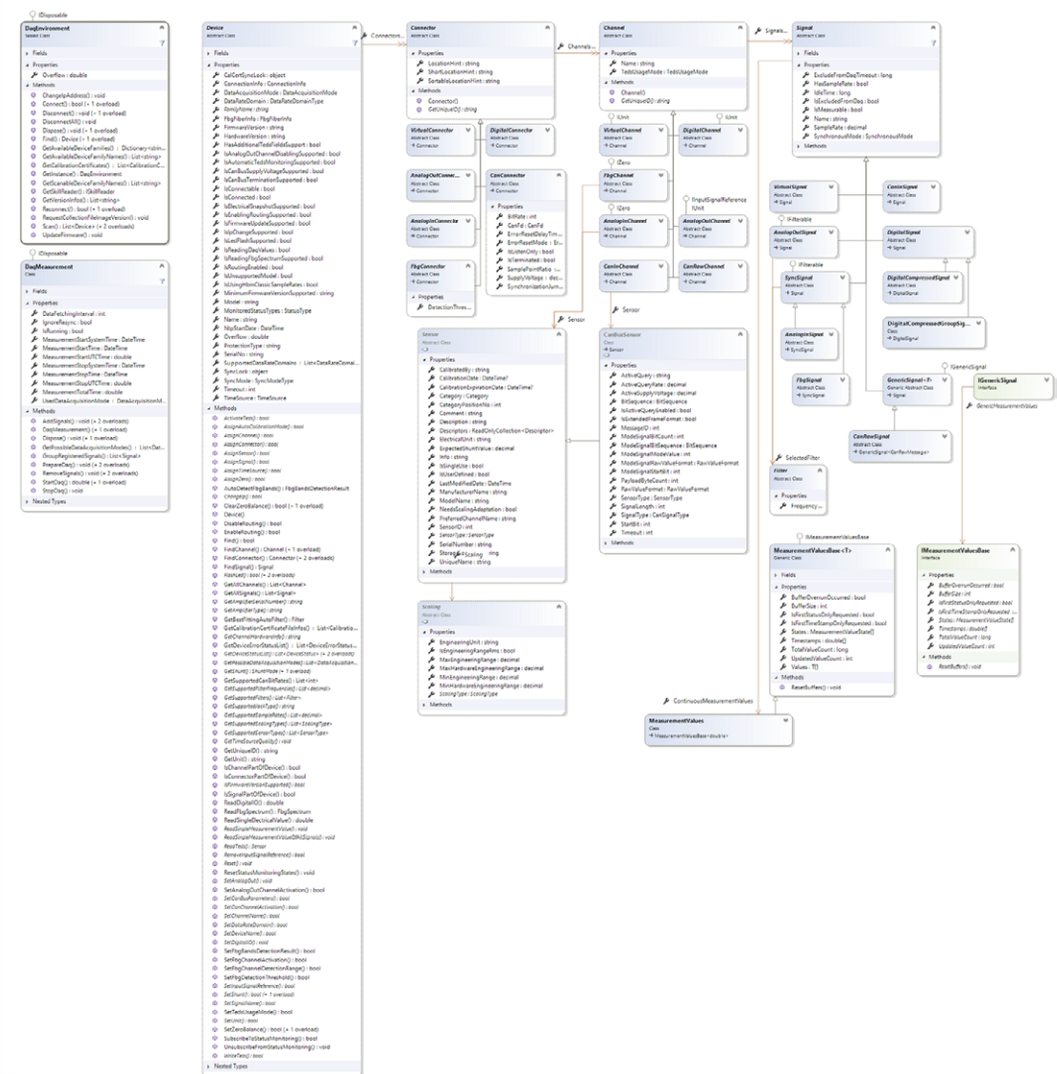


現在位置: Hbm.Api.Common Namespace

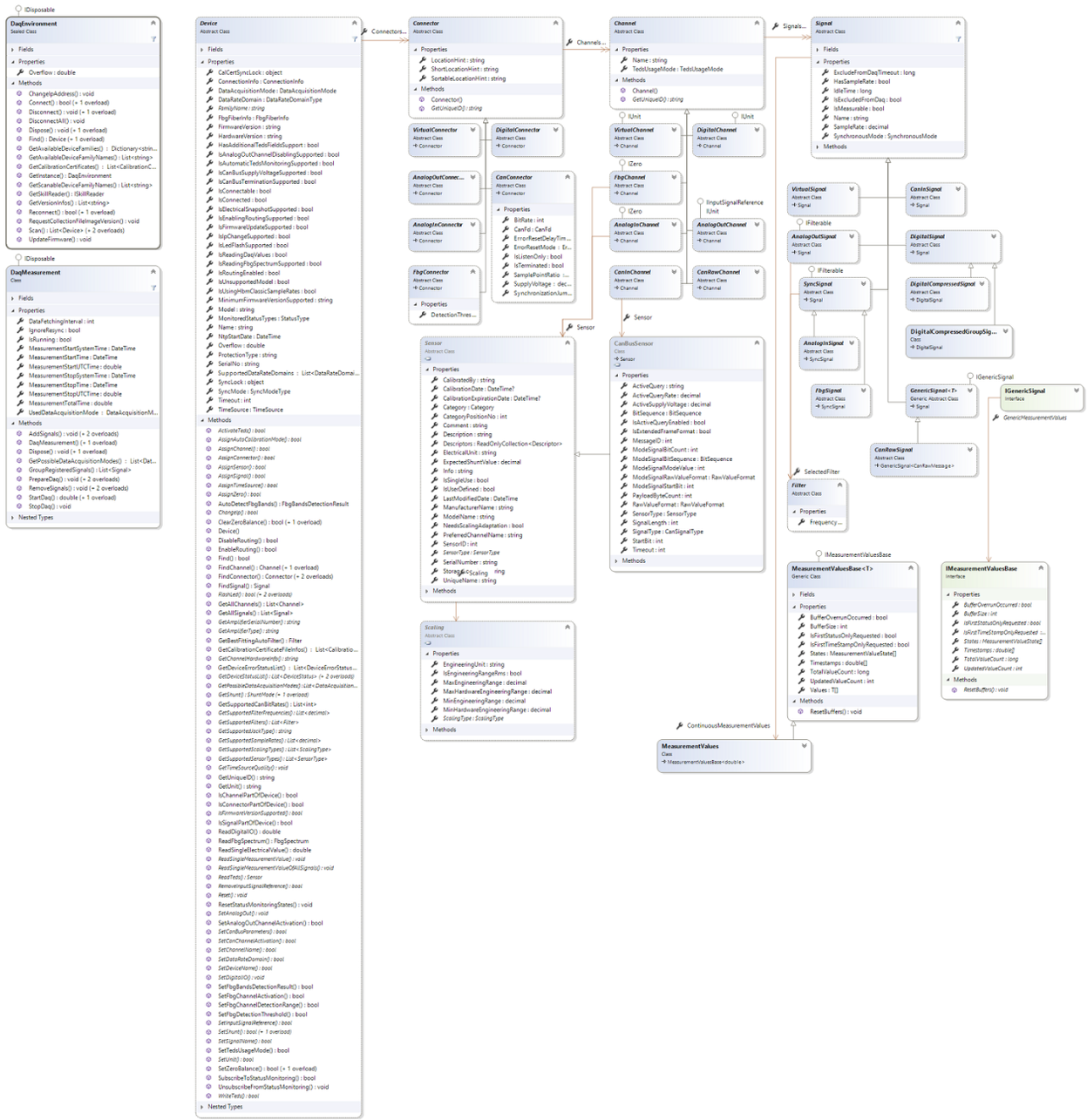
名前空間

名前	説明
Hbm.Api.Common	これは名前空間Hbm.Api.Commonです。
Hbm.Api.Common.Entities	これは名前空間Hbm.Api.Common.Entitiesです。
Hbm.Api.Common.Enums	これは名前空間Hbm.Api.Common.Enumsです。
Hbm.Api.Common.Exceptions	これは名前空間Hbm.Api.Common.Exceptionsです。
Hbm.Api.Common.Messaging	Hbm.Api.Common.Messaging Namespaceには、デバイスやチャネルの特定のステータス変更に関するさまざまなイベントを受信できるようにする関数が含まれています。 チャネルがオーバーフロー状態に切り替わったときなど、状態の変化について通知を受けるには、ステータスのタイプと監視するチャネルを指定する必要があります。 一部のデバイスでは、監視対象の特定の変更(例: PMXのTEDSステータスの変更)が発生した場合、全体的なパフォーマンスが大幅に低下する可能性があることに注意してください。

トピック

名前	説明
クラス概要の詳細	

クラス概要の詳細



現在位置: Hbm.Api.Common Namespace

Hbm.Api.Common Namespace

これは名前空間Hbm.Api.Commonです。

クラス

名前	説明
DaqEnvironment	DaqEnvironment.GetInstance()(DaqEnvironmentはシングルトンです!) を呼び出すことにより、システムは実行可能ファイルのディレクトリ内にあるすべてのプラグ可能なDeviceFamily実装を(DeviceDriver.pluginsファイルを分析することにより)決定し、登録します。 その後、登録されたデバイスファミリをクエリできるようになり、登録されたファミリの1つに属する任意のデバイスをスキャンして接続できるようになります。 デバイスが接続されると、Deviceオブジェクトには物理デバイスの完全なオブジェクト表現が含まれます。各デバイスにはConnectorオブジェクトのリストがあります。これらの各コネクタオブジェクトにはChannelオブジェクトのリストがあり、各チャネルオブジェクトには... 詳細
DaqMeasurement	データの取得を管理するクラス。 連続計測を実行するにはこのクラスを使用します。したがって、次のワークフローに従う必要があります: • デバイスのスキャン(Scan) • デバイスに接続(Connect) • 計測から信号の追加、削除(AddSignals、RemoveSignals) • データ収集の準備(PrepareDaq) • データ収集開始(StartDaq) • Loop ◦ 計測値の取得(FillMeasurementValues) ◦ これらの値を使用して何かを行う • Loop end • データ収集停止(StopDaq) デバイスの接続後(Device.AssignConnectorなどで計測タスクの準備が整った後)、計測したいデバイスの信号を選択できます... 詳細

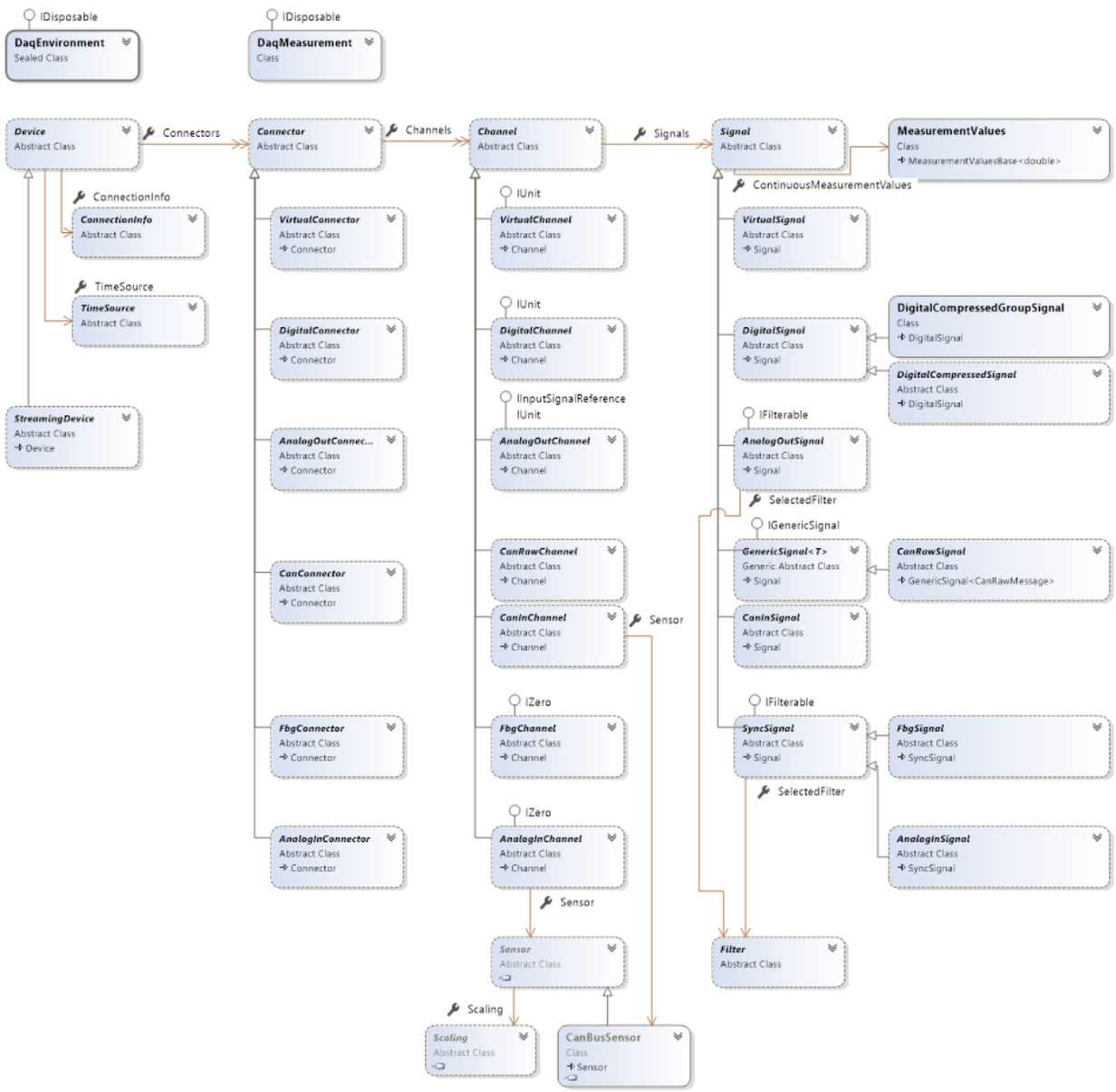
DaqEnvironment Class

DaqEnvironment.GetInstance()(DaqEnvironmentはシングルトンです!)を呼び出すことにより、システムは実行可能ファイルのディレクトリ内にあるすべてのプラグ可能なDeviceFamily実装を(DeviceDriver.pluginsファイルを分析することにより)決定し、登録します。

その後、登録されたデバイスファミリをクエリできるようになり、登録されたファミリの1つに属する任意のデバイスをスキャンして接続できるようになります。

デバイスが接続されると、Deviceオブジェクトには物理デバイスの完全なオブジェクト表現が含まれます。各デバイスにはConnectorオブジェクトのリストがあります。これらの各ConnectorオブジェクトにはChannelオブジェクトのリストがあり、各ChannelオブジェクトにはSignalオブジェクトとSensorオブジェクト(AnalogInChannelの場合)のリストがあります。

ここで、Channel (例: NameまたはSensor)またはSignal(例: FilterまたはSampleRate)オブジェクトのプロパティを変更して、計測タスクを準備できます。

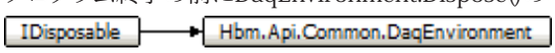


これらオブジェクトのプロパティの変更後、変更内容を物理デバイスに送信する必要があります。変更されたオブジェクトに応じて、いずれかの関数を呼び出す必要があります:
• Device.AssignConnector(..) - コネクタ以降のすべてのチャネル、信号、センサ、ゼロ設定も割り当てます。
• Device.AssignChannel(..) -- すべての信号、センサ、およびチャネル以降のゼロ設定も割り当てます。
• Device.AssignSensor(..)、
• Device.AssignSignal(..) または
• Device.AssignZero(..)

固有のデバイスに適合しない設定または無効な設定がある場合は、割り当てプロセス中に発生した問題を説明するProblemオブジェクトのリストを取得します。このProblemオブジェクトのリスト内で、設定が適合しない場合はWarningが発生し、設定が無効な場合はErrorが発生します。

設定の割り当てが成功したら、選択したデバイス固有特定の信号を計測に追加し、データ収集の準備を続けることができます。これはDaqMeasurementオブジェクトによってサポートされています。

プログラム終了の前にDaqEnvironment.Dispose()の呼び出しをアサートします。



C++

```
class DaqEnvironment : public IDisposable;
```

C#

```
public sealed class DaqEnvironment : IDisposable;
```

Visual Basic

```
public sealed class DaqEnvironment
Inherits IDisposable
```

ファイル

DaqEnvironment

備考

すべてのデバイスでシステム全体へのアクセスできます。すべてのデバイスを検索するには、ここから始める必要があります。

例

```
...
List<Problem> lastProblems; // Used to return warnings and errors
DaqEnvironment daqEnvironment = DaqEnvironment.GetInstance(); // Determine all plugable
DeviceFamily implementations
List<Device> devices = daqEnvironment.Scan();    // Scan entire system for all supported
devices
// List<Device> devices=daqEnvironment.Scan(new List<String> { "PMX" }); // Scan system for
PMX devices only!
daqEnvironment.Connect(devices[0], out lastProblems);  // Connect to first found device
// Try to configure a voltage sensor at the first channel of the device
// Check if channel is an AnalogInChannel (VirtualChannels do not accept a sensor!)
if (devices[0].Connectors[0].Channels[0] is AnalogInChannel)
{
  // Create a new voltageSensor
  Sensor voltageSensor = SensorFactory.CreateSensor(SensorType.Voltage);
  // and attach it to a certain channel
  (devices[0].Connectors[0].Channels[0] as AnalogInChannel).Sensor = voltageSensor;
  // Assign functions always fill a List<Problems> that contains warnings or errors
  // that occurred during the assignment process
  // Assign new VoltageSensor at the used channel..
  if (devices[0].AssignSensor(devices[0].Connectors[0].Channels[0], out lastProblems))
  {
    // Success, no error has occurred.
  }
  else
  {
    //all AssignXXX functions return false, if any Error occurred during assignment
  }
  Debug.Print("There were {0} problems during assign:", lastProblems.Count);
  foreach (Problem problem in lastProblems)
  {
    Debug.Print("ProblemType={0}; Message={1}", problem.GetType(), problem.Message);
  }
}
...
daqEnvironment.Dispose();

....
```

DaqEnvironment Members

次の表に、DaqEnvironmentによって公開されるメンバーをリストします。

Private Methods

	名前	説明
	~DaqEnvironment	デストラクタ。まだ完了していない場合は、すべてをクリーンアップします。
	DaqEnvironment	コンストラクタ。利用可能なすべてのDeviceFamilyを読み取ります。

Public Methods

	名前	説明
	ChangeIpAddress	デバイスのIPアドレスを変更します。
	Connect	Connectメソッドの多重定義の概要です。
	Disconnect	Disconnectメソッドの多重定義の概要です。
	DisconnectAll	接続しているすべてのデバイスを切断します。
	Dispose	使用しているすべてのリソースを解放します。
	Find	Findメソッドの多重定義の概要です。
	GetAvailableDeviceFamilies	利用可能なデバイス ファミリの目録を返します。
	GetAvailableDeviceFamilyNames	使用可能なすべてのDeviceFamilyNameのリストを確定します。
	GetCalibrationCertificates	指定されたデバイスのリストに従って、校正証明書ファイル情報のリストを取得します。要求した証明書は、outputDirectoryに保存されます。
S	GetInstance	DaqEnvironment(Singleton)のインスタンスを取得します
	GetScanableDeviceFamilyNames	スキャン可能なすべてのデバイスファミリのリストを確定します。
	GetSkillReader	指定されたデバイスファミリ名に従ってSkillReaderを取得します。
	GetVersionInfos	使用しているアセンブリとそのバージョン番号のリストを返します。
	Reconnect	Reconnectメソッドの多重定義の概要です。
	RequestCollectionFileImageVersion	選択したファイルのイメージバージョンを要求します(HBMデバイスのみ)
	Scan	Scanメソッドの多重定義の概要です。
	UpdateFirmware	選択したデバイスのファームウェアを更新します(HBMデバイスのみ)。

Public Properties

	名前	説明
	Overflow	オーバーフロー値(デフォルトは1000000)。この値は、通常はデバイスファミリ間で異なる、無効な計測値にスポットライトを当てるために使用します。ステータスが無効な各計測値については、対応する計測値がこのオーバーフロー値に設定されます。

DaqEnvironment.~DaqEnvironment Destructor

デストラクタ。まだ完了していない場合は、すべてをクリーンアップします。

C++

```
~DaqEnvironment();
```

C#

```
private ~DaqEnvironment()
```

Visual Basic

```
Private Sub Finalize
```

DaqEnvironment.DaqEnvironment Constructor

コンストラクタ。利用可能なすべてのDeviceFamilyを読み取ります。

C++

```
DaqEnvironment();
```

C#

```
private DaqEnvironment()
```

Visual Basic

```
Private Sub New()
```

DaqEnvironment Methods

DaqEnvironmentクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ChangeIpAddress	デバイスのIPアドレスを変更します。
	Connect	Connectメソッドの多重定義の概要です。
	Disconnect	Disconnectメソッドの多重定義の概要です。
	DisconnectAll	接続しているすべてのデバイスを切断します。
	Dispose	使用しているすべてのリソースを解放します。
	Find	Findメソッドの多重定義の概要です。
	GetAvailableDeviceFamilies	利用可能なデバイス ファミリの目録を返します。
	GetAvailableDeviceFamilyNames	使用可能なすべてのDeviceFamilyNameのリストを確定します。
	GetCalibrationCertificates	指定されたデバイスのリストに従って、校正証明書ファイル情報のリストを取得します。要求した証明書は、outputDirectoryに保存されます。
S	GetInstance	DaqEnvironment(Singleton)のインスタンスを取得します
	GetScanableDeviceFamilyNames	スキャン可能なすべてのデバイスファミリのリストを確定します。
	GetSkillReader	指定されたデバイスファミリ名に従ってSkillReaderを取得します。
	GetVersionInfos	使用しているアセンブリとそのバージョン番号のリストを返します。
	Reconnect	Reconnectメソッドの多重定義の概要です。
	RequestCollectionFileImageVersion	選択したファイルのイメージバージョンを要求します(HBMデバイスのみ)
	Scan	Scanメソッドの多重定義の概要です。
	UpdateFirmware	選択したデバイスのファームウェアを更新します(HBMデバイスのみ)。

DaqEnvironment.ChangeIpAddress Method

デバイスのIPアドレスを変更します。

C++

```
[PublicAPI]
void ChangeIpAddress(
    Device device,
    bool isDhcp,
    string ipAddress,
    string subnetMask,
    out List<Problem> problems
```

C#

```
[PublicAPI]
public void ChangeIpAddress(
    Device device,
    bool isDhcp,
    string ipAddress,
    string subnetMask,
    out List<Problem> problems
);
```

Visual Basic

```
<PublicAPI>
public Function ChangeIpAddress(
    device As Device,
    isDhcp As bool,
    ipAddress As string,
    subnetMask As string,
    problems As out List<Problem>
) As void
```

引数

引数	説明
device	IPを変更しようとしているデバイス。
isDhcp	dhcpを使用する必要がある場合はTrue(自動IP割り当て)。
ipAddress	適用するIPアドレス。
subnetMask	適用するサブネットマスク。
problems	発生した問題。

Connect Method

Connectメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqEnvironment.Connect (Device, out List<Problem>)	物理デバイス設定を読み取り、デバイスオブジェクトモデルを入力し、1つのデバイスを接続します。
	DaqEnvironment.Connect (List<Device>, out List<Problem>)	物理デバイス設定を読み取り、デバイスオブジェクトモデルを入力することにより、複数のデバイスを一度に接続します。

DaqEnvironment.Connect Method (Device, out List<Problem>)

物理デバイス設定を読み取り、デバイス オブジェクトモデルを入力し、1つのデバイスを接続します。

C++

```
[PublicAPI]
bool Connect(
    Device device,
    out List<Problem> problems
);
```

C#

```
[PublicAPI]
public bool Connect(
    Device device,
    out List<Problem> problems
);
```

Visual Basic

```
<PublicAPI>
public Function Connect(
    device As Device,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
device	接続するデバイス。
problems	発生した問題。

戻り値

エラーが発生しなかった場合はtrue、それ以外の場合はfalse。

例外

例外	説明
System.ArgumentNullException	デバイスがnullです。

DaqEnvironment.Connect Method (List<Device>, out List<Problem>)

物理デバイス設定を読み取り、デバイスオブジェクトモデルを入力することにより、複数のデバイスを一度に接続します。

C++

```
[PublicAPI]
bool Connect(
    List<Device> device,
    out List<Problem> problems
);
```

C#

```
[PublicAPI]
public bool Connect(
    List Device device,
    out List<Problem> problems
);
```

Visual Basic

```
<PublicAPI>
public Function Connect(
    device As Device,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
device	接続するデバイス。
problems	発生した問題。

戻り値

エラーが発生しなかった場合はtrue、それ以外の場合はfalse。

備考

指定したデバイスを並列に接続します。各デバイスは、接続に成功した直後に、Messaging.MessageBrokerを通じて**DeviceConnected**イベントを発生させます。

例外

例外	説明
System.ArgumentNullException	デバイスがnullです。

Disconnect Method

これは、Disconnectメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqEnvironment.Disconnect (Device)	単一のデバイスを切断します。
	DaqEnvironment.Disconnect (List<Device>)	複数のデバイスを切断します。

DaqEnvironment.Disconnect Method (Device)

単一のデバイスを切断します。

C++

```
[PublicAPI]
void Disconnect(
    Device device
);
```

C#

```
[PublicAPI]
public void Disconnect(
    Device device
);
```

Visual Basic

```
<PublicAPI>
public Function Disconnect(
    device As Device
) As void
```

引数

引数	説明
device	切断するデバイス。

DaqEnvironment.Disconnect Method (List<Device>)

複数のデバイスを切断します。

C++

```
[PublicAPI]
void Disconnect(
    List<Device> device
);
```

C#

```
[PublicAPI]
public void Disconnect(
    List<Device> device
);
```

Visual Basic

```
<PublicAPI>
public Function Disconnect(
    device As List<Device>
) As void
```

引数

引数	説明
device	切断するデバイスのリスト。

備考

指定されたデバイスを並列に切断します。各デバイスは、切断した直後に、Messaging.MessageBrokerを通じて**DeviceDisconnected**イベントを発生させます。

DaqEnvironment.DisconnectAll Method

接続しているすべてのデバイスを切断します。

C++

```
[PublicAPI]
void DisconnectAll();
```

C#

```
[PublicAPI]
public void DisconnectAll();
```

Visual Basic

```
<PublicAPI>
public Function DisconnectAll() As void
```

DaqEnvironment.Dispose Method

使用されているすべてのリソースを解放します。

C++

```
[PublicAPI]
void Dispose();
```

C#

```
[PublicAPI]
public void Dispose();
```

Visual Basic

```
<PublicAPI>
public Function Dispose() As void
```

Find Method

これは、Find メソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqEnvironment.Find (string, ConnectionInfo, out List<Problem>)	指定したファミリとconnectionInfoに従ってデバイスを検索します。
	DaqEnvironment.Find (string, List<ConnectionInfo>, out List<Problem>)	指定したファミリとconnectionInfosに従ってデバイスを検索します。

DaqEnvironment.Find Method (string, ConnectionInfo, out List<Problem>)

指定したファミリとconnectionInfoに従ってデバイスを検索します。

C++

```
[PublicAPI]
Device Find(
    string familyName,
    ConnectionInfo connectionInfo,
    out List<Problem> problems
);
```

C#

```
[PublicAPI]
public Device Find(
    string familyName,
    ConnectionInfo connectionInfo,
    out List<Problem> problems
);
```

Visual Basic

```
<PublicAPI>
public Function Find(
    familyName As string,
    connectionInfo As ConnectionInfo,
    problems As out List<Problem>
) As Device
```

引数

引数	説明
familyName	指定したconnectionInfoを持つデバイスが検索されるファミリ名。
connectionInfo	接続情報。
problems	検索プロセス中に発生した警告とエラーのリスト。

戻り値

見つかったデバイスまたは null。

DaqEnvironment.Find Method (string, List<ConnectionInfo>, out List<Problem>)

指定したファミリとconnectionInfosに従ってデバイスを検索します。

C++

```
[PublicAPI]
List<Device> Find(
    string familyName,
    List<ConnectionInfo> connectionInfos,
    out List<Problem> problems
);
```

C#

```
[PublicAPI]
public List<Device> Find(
    string familyName,
    List<ConnectionInfo> connectionInfos,
    out List<Problem> problems
);
```

Visual Basic

```
<PublicAPI>
public Function Find(
    familyName As string,
    connectionInfos As List<ConnectionInfo>,
    problems As out List<Problem>
) As List <Device>
```

引数

引数	説明
familyName	指定したconnectionInfoを持つデバイスが検索されるファミリ名。
connectionInfo	接続情報。
problems	検索プロセス中に発生した警告とエラーのリスト。

戻り値

検出されたすべてのデバイスのリスト。

DaqEnvironment.GetAvailableDeviceFamilies Method

利用可能なデバイス ファミリの辞書を返します。

C++

```
[PublicAPI]
Dictionary<string,DeviceFamily> GetAvailableDeviceFamilies();
```

C#

```
[PublicAPI]
public Dictionary<string,DeviceFamily> GetAvailableDeviceFamilies();
```

Visual Basic

```
<PublicAPI>
public Function GetAvailableDeviceFamilies() As Dictionary<string,DeviceFamily>
```

戻り値

利用可能なデバイス ファミリ。

DaqEnvironment.GetAvailableDeviceFamilyNames Method

利用可能なすべての DeviceFamilyNamesのリストを決定します。

C++

```
[PublicAPI]
List<string>, GetAvailableDeviceFamilyNames();
```

C#

```
[PublicAPI]
public List<string> GetAvailableDeviceFamilyNames();
```

Visual Basic

```
<PublicAPI>
public Function GetAvailableDeviceFamilyNames() As List<string>
```

戻り値

使用可能なすべてのDeviceFamilyNamesのリスト。

DaqEnvironment.GetCalibrationCertificates Method

指定されたデバイスのリストに従って、校正証明ファイル情報のリストを取得します。要求された証明書は、outputDirectoryに保存されます。

C++

```
[PublicAPI]
List<CalibrationCertificateFileInfo> GetCalibrationCertificates(
    List<Device> devices,
    ReleaseType releaseType,
    out Dictionary<Device, List<Problem>> devicesProblems,
    string outputBaseDirectory = null
);
```

C#

```
[PublicAPI]
public List<CalibrationCertificateFileInfo> GetCalibrationCertificates(
    List<Device> devices,
    ReleaseType releaseType,
    out Dictionary<Device, List<Problem>> devicesProblems,
    string outputBaseDirectory
);
```

Visual Basic

```
<PublicAPI>
public Function GetCalibrationCertificates(
    devices As List<Device>,
    releaseType As ReleaseType,
    devicesProblems As out Dictionary<Device, List<Problem>>,
    outputBaseDirectory As string = null
) As List<CalibrationCertificateFileInfo>
```

引数

引数	説明
devices	校正証明を取得(ダウンロード)するデバイス。
releaseType	デバイスから取得する校正証明のリリースタイプ。
devicesProblems	発生した問題。
outputBaseDirectory	指定したデバイスの校正証明を保存する既存のディレクトリ(デバイス毎のサブフォルダ)。

戻り値

校正証明ファイル情報のリスト。

備考

一時的に作成した(本体からダウンロードした)校正証明書は忘れずに削除してください。

例外

例外	説明
UnauthorizedAccessException	現在のユーザに指定されたOutputBaseDirectory にファイルを書き込む権限がない場合にスローされます。

DaqEnvironment.GetInstance Method

これを使用してDaqEnvironment (Singleton)のインスタンスを取得します。

C++

```
[PublicAPI]
DaqEnvironment GetInstance();
```

C#

```
[PublicAPI]
public static DaqEnvironment GetInstance();
```

Visual Basic

```
<PublicAPI>
public static Function GetInstance As DaqEnvironment
```

戻り値

DaqEnvironmentのインスタンス。

DaqEnvironment.GetScanableDeviceFamilyNames Method

スキャン可能なすべてのデバイスファミリのリストを決定します。

C++

```
[PublicAPI]
List<string> GetScanableDeviceFamilyNames();
```

C#

```
[PublicAPI]
public List string GetScanableDeviceFamilyNames();
```

Visual Basic

```
<PublicAPI>
public Function GetScanableDeviceFamilyNames As List<string>
```

戻り値

スキャン可能なすべてのデバイスファミリのリスト。

DaqEnvironment.GetSkillReader Method

指定したデバイスファミリ名に基づいてSkillReaderを取得します。

C++

```
[PublicAPI]
ISkillReader GetSkillReader(
    string familyName
);
```

C#

```
[PublicAPI]
public ISkillReader GetSkillReader(
    string familyName
);
```

Visual Basic

```
<PublicAPI>
public Function GetSkillReader(
    familyName As string
) As ISkillReader
```

引数

引数	説明
familyName	デバイスファミリ名。

戻り値

指定したデバイス ファミリ名に基づく SkillReader。

DaqEnvironment.GetVersionInfos Method

使用しているアセンブリとそのバージョン番号のリストを返します。

C++

```
[PublicAPI]
List<string> GetVersionInfos();
```

C#

```
[PublicAPI]
public List string GetVersionInfos();
```

Visual Basic

```
<PublicAPI>
public Function GetVersionInfos As List<string>
```

Reconnect Method

これは、Reconnect メソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqEnvironment.Reconnect (Device, out List<Problem>)	デバイスを再接続します。指定したデバイスに接続し、物理デバイスの既存の設定を上書きすることで、現在のデバイスオブジェクトの設定を復元します。
	DaqEnvironment.Reconnect (List<Device>, out List<Problem>)	複数のデバイスを一度に再接続します。指定したデバイスに接続し、物理デバイスの既存の設定を上書きすることで、現在のデバイスオブジェクトの設定を復元します。

DaqEnvironment.Reconnect Method (Device, out List<Problem>)

デバイスを再接続します。指定したデバイスに接続し、物理デバイスの既存の設定を上書きすることで、現在のデバイスオブジェクトの設定を復元します。

C++

```
[PublicAPI]
bool Reconnect(
    Device device,
    out List<Problem> problems
);
```

C#

```
[PublicAPI]
public bool Reconnect(
    Device device,
    out List<Problem> problems
);
```

Visual Basic

```
<PublicAPI>
public Function Reconnect(
    device As Device,
    problems As List<Problem>
) As bool
```

引数

引数	説明
device	再接続するデバイス。
problems	再接続プロセス中に発生した警告とエラーのリスト。

戻り値

エラーが発生していない場合はtrue、それ以外の場合はfalse。

例外

例外	説明
System.ArgumentNullException	デバイスがnullです。

DaqEnvironment.Reconnect Method (List<Device>, out List<Problem>)

複数のデバイスを一度に再接続します。指定したデバイスに接続し、物理デバイスの既存の設定を上書きすることで、現在のデバイスオブジェクトの設定を復元します。

C++

```
[PublicAPI]
bool Reconnect(
    List<Device> device,
    out List<Problem> problems
);
```

C#

```
[PublicAPI]
public bool Reconnect(
    List<Device> device,
    out List<Problem> problems
);
```

Visual Basic

```
<PublicAPI>
public Function Reconnect(
    device As List<Device>,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
device	再接続するデバイス。
problems	再接続プロセス中に発生した警告とエラーのリスト。

戻り値

エラーが発生していない場合はtrue、それ以外の場合はfalse。

例外

例外	説明
System.ArgumentNullException	デバイスがnullです。

DaqEnvironment.RequestCollectionFileImageVersion Method

選択したファイルのイメージバージョンを要求します(HBMデバイスのみ)。
このメソッドは将来のバージョンで置き換え、または変更される予定です。使用しないでください!

C++

```
[Obsolete("This method will be replaced or changed in a future version. Please do not use it!", false)]
[PublicAPI]
void RequestCollectionFileImageVersion(
    string fileName,
    List<Device> devices,
    bool allowFirmwareVersionDowngrade,
    out Dictionary<Device,List<Problem>> devicesProblems
);
```

C#

```
[Obsolete("This method will be replaced or changed in a future version. Please do not use it!", false)]
[PublicAPI]
public void RequestCollectionFileImageVersion(
    string fileName,
    List<Device> devices,
    bool allowFirmwareVersionDowngrade,
    out Dictionary<Device, List<Problem>> devicesProblems
);
```

Visual Basic

```
<Obsolete("This method will be replaced or changed in a future version. Please do not use it!", false)>
<PublicAPI>
public Function RequestCollectionFileImageVersion(
    fileName As string,
    device As List<Device>,
    devicesProblems  As out Dictionary<Device, List<Problem>>
) As void
```

引数

引数	説明
fileName	更新するファームウェアが含まれるファイル名。
device	更新するデバイスのリスト。
allowFirmwareVersionDowngrade	ダウングレードを許可するかどうか。
devicesProblems	発生した問題。

Scan Method

これは、Scanメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqEnvironment.Scan ()	使用可能なすべての DeviceFamiliesに対してスキャンを実行し、見つかったすべてのデバイスのリストを返します。注意: DaqEnviromentオブジェクトのインスタンス化とScan()関数の最初の呼び出しの間に、利用可能なデバイスの完全なリストを取得するまで少なくとも6秒待つ必要があります。最後のスキャン以降に追加のデバイスがオンまたはオフになっている可能性があるため、この機能を使用して利用可能なデバイスのリストを更新することもできます。
	DaqEnvironment.Scan (List<string>)	指定したすべてのDeviceFamilyに対してスキャンを実行し、見つかったデバイスのリストを返します。
	DaqEnvironment.Scan (string)	指定されたDeviceFamilyでスキャンを実行し、見つかったデバイスのリストを返します。

DaqEnvironment.Scan Method ()

使用可能なすべての DeviceFamiliesに対してスキャンを実行し、見つかったすべてのデバイスのリストを返します。
注意: DaqEnviromentオブジェクトのインスタンス化とScan()関数の最初の呼び出しの間に、利用可能なデバイスの完全なリストを取得するまで少なくとも6秒待つ必要があります。最後のスキャン以降に追加のデバイスがオンまたはオフになっている可能性があるため、この機能を使用して利用可能なデバイスのリストを更新することもできます。

C++

```
[PublicAPI]
List<Device> Scan();
```

C#

```
[PublicAPI]
public List<Device> Scan();
```

Visual Basic

```
<PublicAPI>
public Function Scan() As List<Device>
```

戻り値

見つかったすべてのデバイスのリスト。

例外

例外	説明
Hbm.Api.Scan.Entities.ScanFailedException	スキャン中にエラーが発生した場合にスローされます。

DaqEnvironment.Scan Method (List<string>)

指定したすべてのDeviceFamilyに対してスキャンを実行し、見つかったデバイスのリストを返します。

C++

```
[PublicAPI]
List<Device> Scan(
    List<string> familyNames
);
```

C#

```
[PublicAPI]
public List<Device> Scan(
    List<string> familyNames
);
```

Visual Basic

```
<PublicAPI>
public Function Scan(
    familyNames As List<string>
) As List<Device>
```

引数

引数	説明
familyNames	スキャンが実行されるDeviceFamilyのリスト。

戻り値

見つかったすべてのデバイスのリスト。

例外

例外	説明
System.ArgumentNullException	familyNamesがnullです。

DaqEnvironment.Scan Method (string)

指定されたDeviceFamilyでスキャンを実行し、見つかったデバイスのリストを返します。

C++

```
[PublicAPI]
List<Device> Scan(
    string familyName
);
```

C#

```
[PublicAPI]
public List<Device> Scan(
    string familyName
);
```

Visual Basic

```
<PublicAPI>
public Function Scan(
    familyName As string
) As List<Device>
```

引数

引数	説明
familyName	スキャンが実行されるDeviceFamilyの名前(例: "MGC"、"PMX"、または "QuantumX")。

戻り値

見つかったすべてのデバイスのリスト。

例外

例外	説明
System.ArgumentNullException	familyNameがnullです。

DaqEnvironment.UpdateFirmware Method

選択したデバイスのファームウェアを更新します(HBMデバイスのみ)。
このメソッドは将来のバージョンで置き換え、または変更される予定です。使用しないでください!

C++

```
[Obsolete("This method will be replaced or changed in a future version. Please do not use
it!", false)]
[PublicAPI]
void UpdateFirmware(
    string fileName,
    List<Device> devices,
    bool allowFirmwareVersionDowngrade,
    out Dictionary<Device, List<Problem>> devicesProblems
);
```

C#

```
[Obsolete("This method will be replaced or changed in a future version. Please do not use
it!", false)]
[PublicAPI]
public void UpdateFirmware(
    string fileName,
    List<Device> devices,
    bool allowFirmwareVersionDowngrade,
    out Dictionary<Device, List<Problem>> devicesProblems
);
```

Visual Basic

```
<Obsolete("This method will be replaced or changed in a future version. Please do not use
it!", false)>
<PublicAPI>
public Function UpdateFirmware(
    fileName As string,
    devices As List<Device>,
    allowFirmwareVersionDowngrade As bool,
    devicesProblems As out Dictionary<Device, List<Problem>>
) As void
```

引数

引数	説明
fileName	ファームウェアのファイル名。
devices	更新するデバイスのリスト。
allowFirmwareVersionDowngrade	ファームウェアのダウングレードを許可するかどうか。
devicesProblems	発生した問題。

DaqEnvironment Properties

DaqEnvironmentクラスのプロパティは次のとおりです。

	名前	説明
	Overflow	オーバーフロー値(デフォルトは1000000)。この値は、通常はデバイスファミリ間で異なる、無効な計測値にスポットライトを当てるために使用します。ステータスが無効な各計測値については、対応する計測値がこのオーバーフロー値に設定されます。

DaqEnvironment.Overflow Property

オーバーフロー値(デフォルトは1000000)。この値は、通常はデバイスファミリ間で異なる、無効な計測値にスポットライトを当てるために使用します。ステータスが無効な各計測値については、対応する計測値がこのオーバーフロー値に設定されます。

C++

```
__property double Overflow;
```

C#

```
public double Overflow;
```

Visual Basic

```
public property Overflow() As double
```

備考

このオーバーフロー値はすべてのデバイスで使用します!

DaqMeasurement Class

データの取得を管理するクラス。

連続計測を実行するにはこのクラスを使用します。したがって、次のワークフローに従う必要があります:
• デバイスのスキャン(Scan)
• デバイスに接続(Connect)
• 計測から信号の追加、削除(AddSignals、RemoveSignals)
• データ収集の準備(PrepareDaq)
• データ収集開始(StartDaq)
• Loop
◦ 計測値の取得(FillMeasurementValues)
◦ これらの値を使用して何かを行う
• Loop end
• データ収集停止(StopDaq)

デバイスの接続後(Device.AssignConnectorなどで計測タスクの準備が整った後)、計測したいデバイスの信号を選択できます。これらの各信号を計測に追加する必要があります。

PrepareDaqを呼び出して、APIに信号の内部循環バッファを割り当て、信号を異なるサンプルレートに応じて異なるグループに分類するなどの内部準備をさらに実行します。

計測を開始するには、StartDaqを呼び出す必要があります。複数のデバイスを使用しており、それらのデバイスが同期している場合(つまりデバイスのシステム時間がほぼ等しいことを意味します)、DataAcquisitionMode.TimestampSynchronizedを使用してデータ収集を開始する必要があります。同期計測の場合、各信号の最初のタイムスタンプは同じである(または少なくとも可能な限り互いに近い)ことが理想的です。

計測値を取得するには、関数FillMeasurementValuesを定期的に呼び出す必要があります。この関数が次に呼び出されるまで、計測に追加した信号のSignal.ContinuousMeasurementValuesの下にある計測値にアクセスできます。計測を同期して開始した場合、同じサンプルレートの各信号(さまざまなデバイスに分配)が同じ数の新しい計測値を取得することが保証されます。

StopDaqは実行中の計測を終了し、以前に追加されたすべての信号を計測から削除します。したがって、新しい計測を準備/開始する前に、信号を再度追加することに注意してください。



C++

```
class DaqMeasurement : public IDisposable;
```

C#

```
public class DaqMeasurement : IDisposable;
```

Visual Basic

```
public class DaqMeasurement
Inherits IDisposable
```

File

DaqMeasurement.cs

例

次の例は、デバイスをスキャンして接続し、連続計測を実行する方法を示しています。

```
...
// Scan for devices and connect to the first device that has been found
DaqEnvironment daqEnvironment = DaqEnvironment.GetInstance ();
List<Device> devices = daqEnvironment.Scan();
List<Problem> lastProblems; // Used to return warnings and errors
daqEnvironment.Connect(devices[0], out lastProblems);

// Get a List of all measurable signals of the connected device
List<Signal> signalList = devices[0].GetAllSignals().Where(sig => sig.IsMeasurable).
ToList();

// Create a DaqMeasurement object
DaqMeasurement daqMeasurement = new DaqMeasurement();

// Add all measurable signals of the first device to the measurement
daqMeasurement.AddSignals(devices[0], signalList);

// Prepare data acquisition
daqMeasurement.PrepareDaq();

// start synchronized data acquisition...
daqMeasurement.StartDaq(DataAcquisitionMode.TimestampSynchronized);

// run data acquisition for 10 seconds
Stopwatch stopWatch = new Stopwatch();
stopWatch.Start();
while (stopWatch.ElapsedMilliseconds<10000)
{
  // Update measurement values of the signals that were added to the measurement
  daqMeasurement.FillMeasurementValues();

  // work with measurement values ..e.g.:
  foreach (Signal sig in signalList)
  {
    Debug.Print("Number of new measurement values for signal {0} = {1}",
      sig.Name, sig.ContinuousMeasurementValues.UpdatedValueCount);
    if (sig.ContinuousMeasurementValues.UpdatedValueCount > 0)
    {
      Debug.Print("First new measurement value={0}", sig.ContinuousMeasurementValues.
Values[0]);
    }
  }

  // wait a while...
  System.Threading.Thread.Sleep(100);
}

// stop data acquisition
daqMeasurement.StopDaq();

// dispose DaqEnvironment
daqEnvironment.Dispose();

// I M P O R T A N T !!!!!:
// To start a new measurement you have to go through the complete cycle:
// daqMeasurement.AddSignals(..);
// daqMeasurement.PrepareDaq(..);
// and daqMeasurement.StartDaq(..)!
// It is NOT sufficient just to call daqMeasurement.StartDaq(..) again!!!
...
```

DaqMeasurement Members

次の表は、DaqMeasurementによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	AddSignals	これは、AddSignalsメソッドの多重定義の概要です。
	DaqMeasurement	これは、DaqMeasurementコンストラクタの多重定義の概要です。
	Dispose	現在のインスタンスを破棄し、クリーンアップ操作を実行します。
	FillMeasurementValues	この関数は、計測に参加するすべての信号のすべての計測値を更新します。データ収集の同期開始の場合、同じサンプルレートの各信号(さまざまなデバイスに分配)が同じ数の新しい計測値を取得することをアサートします。例えばサンプル レート20Hzのdevice1のsignal_Aは、サンプルレート20Hzのdevice2のsignal_Bと同じ数の新しい計測値を取得します! それ以外の場合(StartDaqが非同期計測を開始した場合、信号はこの関数の最後の呼び出し以降に蓄積されたすべての計測値を取得します...詳細。
	GetPossibleDataAcquisitionModes	計測に参加するすべてのデバイスファミリのすべてのデバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。通常、DataAcquisitionMode.Unsynchronizedはいつでも可能です(そうしないと、デバイスは測定値をまったく提供しません)。DataAcquisitionMode.HardwareSynchronizedは、すべてのデバイスが同じファミリに属し、同期ケーブルを介して同期されている場合にのみ使用できます。すべてのデバイス(異なるデバイス ファミリのものも)が同じTimeSourceを使用する場合、DataAcquisitionMode.TimestampSynchronizedが可能です。
S	GroupRegisteredSignals	必要に応じて登録された信号をグループ化します。登録された信号にDigitalCompressedSignalが含まれている場合、圧縮グループ(DigitalCompressedSignals内で指定)毎に1つのDigitalCompressedGroupSignalが作成されます。DigitalCompressedGroupSignalには、同じ圧縮グループ番号を持つすべてのDigitalCompressedSignalが含まれます。DigitalCompressedGroupSignalは、DigitalCompressedSignalsではなく、計測の一部になります。
	PrepareDaq	これは、PrepareDaqメソッドの多重定義の概要です。
	RemoveSignals	これは、RemoveSignalsメソッドの多重定義の概要です。
	StartDaq	計測に追加されたすべての信号の(同期)測定を開始します。
	StopDaq	計測に参加するすべてのデバイスのデータ収集を停止します。

Public Properties

	名前	説明
	DataFetchingInterval	計測に参加するすべてのデバイスから計測データが要求される時間間隔(ミリ秒)。
	IgnoreResync	データ収集の実行中のタイムソースの再同期を無視する必要がある場合はtrueを返します。現時点では、これはストリーミングデバイス(QuantumXファミリ)でのみ可能です。再同期は、計測の実行中にTimeSourceが変更された場合、または2つのデバイス間の時間差が大きすぎると評価された場合に行われます。デフォルト値はfalseです。値はDaqMeasurement.PrepareDaq()関数によってのみ設定できます。
	IsRunning	データ収集が現在実行中の場合はtrueを返します。
	MeasurementStartSystemTime	このコンピュータで実行された最新の計測のローカル開始日時。この時間は、StartDaq()関数を呼び出し時にPCから取得します。
	MeasurementStartTime	最新の計測の開始日時。この時間は、最新の計測の最初のタイムスタンプ(MeasurementStartUTCTime)をこのコンピュータのローカル日付時刻表現に変換することによって生成されます。
	MeasurementStartUTCTime	UTC形式での最新の計測の開始日時。これは、計測に参加するすべての信号の最小タイムスタンプです。形式はUTC UNIX時間(1.1.2000からの秒数、小数点以下の桁は実行中の秒)。
	MeasurementStopSystemTime	このコンピュータで実行された最新の計測のローカル停止日時。
	MeasurementStopTime	最新の計測の停止日時。
	MeasurementStopUTCTime	最新の計測の終了日時をUTC形式で表示します。これは、計測に参加するすべての信号の最新のタイムスタンプです。形式はUTC UNIX時間(1.1.2000からの秒数、小数点以下の桁は実行中の秒)。
	MeasurementTotalTime	最新の計測の合計時間(秒単位)。
	UsedDataAcquisitionMode	最新の計測を開始するために使用されたDataAcquisitionModeを返します。最新の計測がDataAcquisitionMode.Autoで開始されている場合、使用されるデータ収集モードは、選択された実際のモード(DataAcquisitionMode.Unsynchronized、DataAcquisitionMode.HardwareSynchronized、または DataAcquisitionMode.TimestampSynchronized)になります。

DaqMeasurement Constructor

これは、DaqMeasurementコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqMeasurement.DaqMeasurement ()	クラスの新しいインスタンスを作成する。
	DaqMeasurement.DaqMeasurement (int, int)	クラスの新しいインスタンスを作成する。

DaqMeasurement.DaqMeasurement Constructor ()

クラスの新しいインスタンスを作成する。

C++

```
DaqMeasurement;
```

C#

```
public DaqMeasurement;
```

Visual Basic

```
Public Sub New()
```

DaqMeasurement.DaqMeasurement Constructor (int, int)

クラスの新しいインスタンスを作成する。

C++

```
DaqMeasurement(
    int dataFetchingInterval,
    int signalMaxTransmissionTime = 2000
);
```

C#

```
public DaqMeasurement(
    int dataFetchingInterval,
    int signalMaxTransmissionTime
);
```

Visual Basic

```
Public Sub New(
    dataFetchingInterval As Integer,
    signalMaxTransmissionTime As Integer = 2000
)
```

引数

引数	説明
dataFetchingInterval	計測に参加するすべてのデバイスから計測データが要求される時間間隔(ミリ秒)。
signalMaxTransmissionTime	(デバイスから計測値を受信しないため)信号が計測から除外されるまでの、計測データの送信にかかる最大時間(ミリ秒)。

Hbm.Api.Common Namespace

DaqMeasurement Methods

DaqMeasurementクラスのメソッドは次のとおりです。

Public Methods

	名前	説明
	AddSignals	これは、AddSignalsメソッドの多重定義の概要です。
	DaqMeasurement	これは、DaqMeasurementコンストラクタの多重定義の概要です。
	Dispose	現在のインスタンスを破棄し、クリーンアップ操作を実行します。
	FillMeasurementValues	この関数は、計測に参加するすべての信号のすべての計測値を更新します。データ収集の同期開始の場合、同じサンプルレートの各信号(さまざまなデバイスに分配)が同じ数の新しい計測値を取得することをアサートします。例えばサンプル レート20Hzのdevice1のsignal_Aは、サンプルレート20Hzのdevice2のsignal_Bと同じ数の新しい計測値を取得します! それ以外の場合(StartDaqが非同期計測を開始した場合、信号はこの関数の最後の呼び出し以降に蓄積されたすべての計測値を取得します...詳細。
	GetPossibleDataAcquisitionModes	計測に参加するすべてのデバイスファミリのすべてのデバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。通常、DataAcquisitionMode.Unsynchronizedはいつでも可能です(そうしないと、デバイスは測定値をまったく提供しません)。DataAcquisitionMode.HardwareSynchronizedは、すべてのデバイスが同じファミリに属し、同期ケーブルを介して同期されている場合にのみ使用できます。すべてのデバイス(異なるデバイス ファミリのものも)が同じTimeSourceを使用する場合、DataAcquisitionMode.TimestampSynchronizedが可能です。
S	GroupRegisteredSignals	必要に応じて登録された信号をグループ化します。登録された信号にDigitalCompressedSignalが含まれている場合、圧縮グループ(DigitalCompressedSignals内で指定)毎に1つのDigitalCompressedGroupSignalが作成されます。DigitalCompressedGroupSignalには、同じ圧縮グループ番号を持つすべてのDigitalCompressedSignalが含まれます。DigitalCompressedGroupSignalは、DigitalCompressedSignalsではなく、計測の一部になります。
	PrepareDaq	これは、PrepareDaqメソッドの多重定義の概要です。
	RemoveSignals	これは、RemoveSignalsメソッドの多重定義の概要です。
	StartDaq	計測に追加されたすべての信号の(同期)測定を開始します。
	StopDaq	計測に参加するすべてのデバイスのデータ収集を停止します。

AddSignals Method

これは、AddSignalsメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqMeasurement.AddSignals (Device)	指定したデバイスのすべての信号を計測に追加します。
	DaqMeasurement.AddSignals (Device, List<Signal>)	指定したデバイスの信号を計測に追加します。
	DaqMeasurement.AddSignals (Device, Signal)	指定したデバイスの指定した信号を計測に追加します。

DaqMeasurement.AddSignals Method (Device)

指定したデバイスのすべての信号を計測に追加します。

C++

```
void AddSignals(
    Device device
);
```

C#

```
public void AddSignals(
    Device device
);
```

Visual Basic

```
public Function AddSignals(
    device As Device
) As void
```

引数

引数	説明
device	計測に追加する必要がある信号を含むデバイス。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqAlreadyStartedException	データ収集が実行中の場合にスローされます。
System.ArgumentNullException	デバイスがnullです。
System.ArgumentException	device.GetUniqueDeviceID()がnullまたは空の文字列を返した場合にスローされます。

DaqMeasurement.AddSignals Method (Device, List<Signal>)

指定したデバイスの信号を計測に追加します。

C++

```
void AddSignals(
    Device device,
    List<Signal> signals
);
```

C#

```
public void AddSignals(
    Device device,
    List<Signal> signals
);
```

Visual Basic

```
public Function AddSignals(
    device As Device,
    signals As List<Signal>
) As void
```

引数

引数	説明
device	計測に追加する必要がある信号を含むデバイス。
signals	計測に追加する信号。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqAlreadyStartedException	データ収集が実行中の場合にスローされます。
System.ArgumentNullException	デバイスがnullです。信号がnullです。
System.ArgumentException	device.GetUniqueDeviceID()がnullまたは空の文字列を返した場合にスローされます。信号に計測不可能な信号が含まれている場合にスローされます。

DaqMeasurement.AddSignals Method (Device, Signal)

指定したデバイスの指定した信号を計測に追加します。

C++

```
void AddSignals(
    Device device,
    Signal signal
);
```

C#

```
public void AddSignals(
    Device device,
    Signal signal
);
```

Visual Basic

```
public Function AddSignals(
    device As Device,
    signal As Signal
) As void
```

引数

引数	説明
device	計測に追加する必要がある信号を含むデバイス。
signal	計測に追加する信号。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqAlreadyStartedException	データ収集が実行中の場合にスローされます。
System.ArgumentNullException	デバイスがnullです。信号がnullです。
System.ArgumentException	device.GetUniqueDeviceID()がnullまたは空の文字列を返した場合にスローされます。

DaqMeasurement.Dispose Method

現在のインスタンスを破棄し、クリーンアップ操作を実行します。

C++

```
void Dispose();
```

C#

```
public void Dispose();
```

Visual Basic

```
public Function Dispose() As void
```

DaqMeasurement.FillMeasurementValues Method

この関数は、計測に参加するすべての信号のすべての計測値を更新します。データ収集の同期開始の場合、同じサンプルレートの各信号(さまざまなデバイスに分配)が同じ数の新しい計測値を取得することをアサートします。例えばサンプル レート20Hzのdevice1のsignal_Aは、サンプルレート20Hzのdevice2のsignal_Bと同じ数の新しい計測値を取得します! それ以外の場合、StartDaqが非同期計測を開始した場合、信号はこの関数の最後の呼び出し以降に蓄積されたすべての計測値を取得します。この関数は、各信号の計測値に十分なメモリが割り当てられていることもアサートします。したがって、この関数を定期的に呼び出してください。

C++

```
void FillMeasurementValues(
    bool fillUntilSmallestCommonTimestamp = false
);
```

C#

```
public void FillMeasurementValues(
    bool fillUntilSmallestCommonTimestamp
);
```

Visual Basic

```
public Function FillMeasurementValues(
    fillUntilSmallestCommonTimestamp As bool = false
) As void
```

引数

引数	説明
fillUntilSmallestCommonTimestamp	データ収集の同期開始の場合にのみアクティブになります。true(デフォルトはfalse!)に設定すると、データ収集から除外されていないすべての信号の最新の共通タイムスタンプまでのみ計測値が入力されます。同じサンプルレートの信号は、上で説明したように、同じ数の新しい測定値を取得し続けます。

備考

SynchronModeがNonEquidistantに設定された信号の場合、受信したすべての計測値が返されます。計測がどのように開始されたかは関係ありません(DataAcquisitionModeを参照)。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqNotStartedException	データ収集がまだ開始されていない場合にスローされます。

DaqMeasurement.GetPossibleDataAcquisitionModes Method

計測に参加するすべてのデバイスファミリのすべてのデバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。通常、DataAcquisitionMode.Unsynchronizedはいつでも可能です(そうしないと、デバイスは測定値をまったく提供しません)。DataAcquisitionMode.HardwareSynchronizedは、すべてのデバイスが同じファミリに属し、同期ケーブルを介して同期されている場合にのみ使用できます。すべてのデバイス(異なるデバイス ファミリのものも)が同じTimeSourceを使用する場合、DataAcquisitionMode.TimestampSynchronizedが可能です。

C++

```
List<DataAcquisitionMode> GetPossibleDataAcquisitionModes();
```

C#

```
public List<DataAcquisitionMode> GetPossibleDataAcquisitionModes();
```

Visual Basic

```
public Function GetPossibleDataAcquisitionModes() As List<DataAcquisitionMode>
```

戻り値

現在の同期設定に応じて、このデバイスで可能なDataAcquisitionModeのリスト。

備考

この関数を使用する前にPrepareDaqを呼び出す必要があります。そうしないと、計測に参加する信号とデバイスが不明になるためです。

DaqMeasurement.GroupRegisteredSignals Method

必要に応じて登録された信号をグループ化します。登録された信号にDigitalCompressedSignalが含まれている場合、圧縮グループ(DigitalCompressedSignals内で指定)毎に1つのDigitalCompressedGroupSignalが作成されます。DigitalCompressedGroupSignalには、同じ圧縮グループ番号を持つすべてのDigitalCompressedSignalが含まれます。DigitalCompressedGroupSignalは、DigitalCompressedSignalsではなく、計測の一部になります。

C++

```
List<Signal> GroupRegisteredSignals(
    List<Signal> registeredSignals
);
```

C#

```
public static List<Signal> GroupRegisteredSignals(
    List<Signal> registeredSignals
);
```

Visual Basic

```
public static Function GroupRegisteredSignals(
    registeredSignals As List<Signal>
) As List<Signal>
```

引数

引数	説明
registeredSignals	登録した信号のリスト(1台あたり)。

戻り値

登録された信号のグループ化されたリスト(1つのデバイスの)。

備考

このメソッドは、デバイスがデジタル信号の計測に圧縮(複数のデジタル信号に対して1つの計測値) を使用する場合(Hbm.Api.Common.Entities.Signals.DigitalCompressedSignalを使用)、具体的なデバイス実装で使用できます。次に、デバイス固有のメソッドDevice.ReadSingleMeasurementValueおよびDevice.ReadSingleMeasurementValueOfAllSignalsを実装するのに役立ちます。

PrepareDaq Method

これは、PrepareDaqメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqMeasurement.PrepareDaq (bool)	ほとんどの場合にフィッティングパラメータを使用して連続計測を準備します。
	DaqMeasurement.PrepareDaq (int, bool)	連続計測を準備します。
	DaqMeasurement.PrepareDaq (int, bool)	連続計測を準備します。参加デバイスは並行して準備されます。

DaqMeasurement.PrepareDaq Method (bool)

ほとんどの場合にフィッティングパラメータを使用して連続計測を準備します。

C++

```
void PrepareDaq(
    bool ignoreResync = false
);
```

C#

```
public void PrepareDaq(
    bool ignoreResync
);
```

Visual Basic

```
public Function PrepareDaq(
    ignoreResync As bool = false
) As void
```

引数

引数	説明
ignoreResync	データ収集の実行中のタイムソースの再同期を無視するかどうかを決定します。現時点では、これはストリーミングデバイス(QuantumXファミリ)でのみ可能です。再同期が検出された場合にデータ収集を続行する必要がある場合は、trueに設定します。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqAlreadyStartedException	データ収集がすでに実行されている場合にスローされます。
Hbm.Api.Common.Exceptions.DaqNoSignalsAddedException	信号が計測に追加されていない場合にスローされます。

DaqMeasurement.PrepareDaq Method (int, bool)

連続計測を準備します。

C++

```
void PrepareDaq(
    int bufferTimeout,
    bool ignoreResync = false
);
```

C#

```
public void PrepareDaq(
    int bufferTimeout,
    bool ignoreResync
);
```

Visual Basic

```
public Function PrepareDaq(
    bufferTimeout As Integer,
    ignoreResync As bool = false
) As void
```

引数

引数	説明
bufferTimeout	バッファのタイムアウト(ミリ秒単位)。内部循環バッファのサイズを計算するために使用します。信号ごとに少なくとも1000個の値のバッファがあります。通常、バッファサイズは(bufferTimeout/1000)*信号のサンプルレートであり、計測に事前に追加されます(例: Signal.SampleRate=1200Hz、bufferTimeOut=1000ms => 内部循環バッファのサイズは1200エントリになります)。
ignoreResync	データ収集の実行中のタイムソースの再同期を無視するかどうかを決定します。現時点では、これはストリーミングデバイス(QuantumXファミリ)でのみ可能です。再同期が検出された場合にデータ収集を続行する必要がある場合は、trueに設定します。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqAlreadyStartedException	データ収集がすでに実行されている場合にスローされます。
Hbm.Api.Common.Exceptions.DaqNoSignalsAddedException	信号が計測に追加されていない場合にスローされます。

DaqMeasurement.PrepareDaq Method (int, int, int, bool, bool, int, bool)

連続計測を準備します。参加デバイスは並行して準備されます。

C++

```
void PrepareDaq(
    int bufferTimeout,
    int maxNumberOfConcurrentDaqThreads,
    int minNumberOfBufferedValues = 1000,
    bool isFirstTimeStampOnlyRequested = true,
    bool isFirstStatusOnlyRequested = true,
    int maxThreadStackSize = 262144,
    bool ignoreResync = false
);
```

C#

```
public void PrepareDaq(
    int bufferTimeout,
    int maxNumberOfConcurrentDaqThreads,
    int minNumberOfBufferedValues,
    bool isFirstTimeStampOnlyRequested,
    bool isFirstStatusOnlyRequested,
    int maxThreadStackSize,
    bool ignoreResync
);
```

Visual Basic

```
public Function PrepareDaq(
    bufferTimeout As Integer,
    maxNumberOfConcurrentDaqThreads As Integer,
    minNumberOfBufferedValues As Integer = 1000,
    isFirstTimeStampOnlyRequested As bool = true,
    isFirstStatusOnlyRequested As bool = true,
    maxThreadStackSize As Integer = 262144,
    ignoreResync As bool = false
) As void
```

引数

引数	説明
bufferTimeout	バッファのタイムアウト(ミリ秒単位)。内部循環バッファのサイズを計算するために使用します。信号ごとに少なくとも1000個の値のバッファがあります。通常、バッファサイズは(bufferTimeout/1000)*信号のサンプルレートであり、計測に事前に追加されます(例: Signal.SampleRate=1200Hz、bufferTimeOut=1000ms => 内部循環バッファのサイズは1200エントリになります)。
maxNumberOfConcurrentDaqThreads	デバイスからデータをフェッチし、信号の循環バッファにデータを入れる同時実行スレッドの最大数。少なくとも1つのスレッドが必要です。最大数は、オペレーティングシステムと使用するデバイスの数によって異なります。最大100スレッドまでは問題ありません。例: 20台のデバイスを計測する場合は、デバイスごとに1つのスレッドを使用できます。

引数	説明
minNumberOfBufferedValues	計測に参加する信号の循環バッファにバッファリングされる要素の最小数。MinNumberOfBufferedValuesは、minNumberOfBufferedValues>bufferTimeout*signal.Samplerateの場合に使用されます。これは通常、すべての信号がSynchronMode=NonEquidistantであるサンプルレートを持たない信号(またはサンプル レート = 0 Hz)にとって重要です。
isFirstTimeStampOnlyRequested	trueの場合、FillMeasurementValuesの呼び出し中に等間隔信号の最初のタイムスタンプのみが信号計測値にコピーされます。
isFirstStatusOnlyRequested	trueの場合、FillMeasurementValuesの呼び出し中に等間隔信号の最初の状態値のみが信号計測値にコピーされます。
maxThreadStackSize	各デバイスデータフェッチスレッドで使用される最大スタックサイズ(バイト単位)、またはデフォルトの最大スタックサイズを使用する場合は 0。たとえば、Diadem内でAPIを使用する場合は、このパラメータを0に変更します。
ignoreResync	データ収集の実行中のタイムソースの再同期を無視するかどうかを決定します。現時点では、これはストリーミングデバイス(QuantumXファミリ)でのみ可能です。再同期が検出された場合にデータ収集を続行する必要がある場合は、trueに設定します。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqAlreadyStartedException	データ収集がすでに実行されている場合にスローされます。
Hbm.Api.Common.Exceptions.DaqNoSignalsAddedException	信号が計測に追加されていない場合にスローされます。
System.ArgumentException	maxNumberOfConcurrentDaqThreadsが1未満の場合にスローされます。

RemoveSignals Method

これは、RemoveSignalsメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqMeasurement. RemoveSignals (Device)	指定したデバイスのすべての信号を計測から削除します。
	DaqMeasurement. RemoveSignals (Device, List<Signal>)	指定したデバイスのすでに追加されている信号を計測から削除します。
	DaqMeasurement. RemoveSignals (Device, Signal)	指定したデバイスの指定した信号を計測から削除します。

DaqMeasurement.RemoveSignals Method (Device)

指定したデバイスのすべての信号を計測から削除します。

C++

```
void RemoveSignals(
    Device device
);
```

C#

```
public void RemoveSignals(
    Device device
);
```

Visual Basic

```
public Function RemoveSignals(
    device As Device
) As void
```

引数

引数	説明
device	計測から削除する必要がある信号を含むデバイス。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqAlreadyStartedException	データ収集がすでに実行されている場合にスローされます。
System.ArgumentNullException	デバイスがnullです。
System.ArgumentException	device.GetUniqueDeviceID()がnullまたは空の文字列を返した場合にスローされます。

DaqMeasurement.RemoveSignals Method (Device, List<Signal>)

指定したデバイスのすでに追加されている信号を計測から削除します。

C++

```
void RemoveSignals(
    Device device
    List<Signal> signals
);
```

C#

```
public void RemoveSignals(
    Device device
    List<Signal> signals
);
```

Visual Basic

```
public Function RemoveSignals(
    device As Device,
    signals As List<Signal>
) As void
```

引数

引数	説明
device	計測から削除する必要がある信号を含むデバイス。
signals	計測から削除する信号。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqAlreadyStartedException	データ収集がすでに実行されている場合にスローされます。
System.ArgumentNullException	デバイスがnullです。
System.ArgumentException	device.GetUniqueDeviceID()がnullまたは空の文字列を返した場合にスローされます。

DaqMeasurement.RemoveSignals Method (Device, Signal)

指定したデバイスの指定した信号を計測から削除します。

C++

```
void RemoveSignals(
    Device device
    Signal signal
);
```

C#

```
public void RemoveSignals(
    Device device
    Signal signal
);
```

Visual Basic

```
public Function RemoveSignals(
    device As Device,
    signal As Signal
) As void
```

引数

引数	説明
device	計測から削除する必要がある信号を含むデバイス。
signals	計測から削除する信号。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqAlreadyStartedException	データ収集がすでに実行されている場合にスローされます。
System.ArgumentNullException	デバイスがnullです。
System.ArgumentException	device.GetUniqueDeviceID()がnullまたは空の文字列を返した場合にスローされます。

DaqMeasurement.StartDaq Method

計測に追加されたすべての信号の(同期)測定を開始します。

C++

```
double StartDaq(
    DataAcquisitionMode acquisitionMode,
    int syncTimeOut = 5000
);
```

C#

```
public double StartDaq(
    DataAcquisitionMode acquisitionMode,
    int syncTimeOut
);
```

Visual Basic

```
public Function StartDaq(
    acquisitionMode As DataAcquisitionMode,
    syncTimeOut As Integer = 5000
) As double
```

引数

引数	説明
acquisitionMode	収集モードは、計測の開始時にさまざまなデバイスを同期するために使用する手順と、計測値を提供する方法(DaqMeasurement.FillMeasurementValues) を定義します。
syncTimeOut	同期計測の開始に使用する最大時間(ミリ秒)。この時間内に同期計測を開始できない場合は、Hbm.API.Common.Exceptions.StartDaqFailedExceptionがスローされます。

戻り値

最初の計測値のタイムスタンプ間の最大差(秒単位)。0は、各信号の最初のタイムスタンプが同じ値であることを意味します。

備考

計測が開始されると、開始時刻(計測に参加するすべての信号の最小のタイムスタンプ)がStartTimeで確認できます。計測が非同期で開始された場合、関数は0を返します。

例外

例外	説明
Hbm.Api.Common.Exceptions.DaqStartFailedException	データ収集の開始に失敗した場合にスローされます。
Hbm.Api.Common.Exceptions.DaqAlreadyStartedException	データ収集がすでに実行されている場合にスローされます。
Hbm.Api.Common.Exceptions.DaqNoSignalsAddedException	信号が計測に追加されていない場合にスローされます。
Hbm.Api.Common.Exceptions.DaqNoSignalsPreparedException	信号が計測用に準備されていない場合にスローされます。

DaqMeasurement.StopDaq Method

計測に参加するすべてのデバイスのデータ収集を停止します。

C++

```
void StopDaq();
```

C#

```
public void StopDaq();
```

Visual Basic

```
public Function StopDaq() As void
```

DaqMeasurement Properties

DaqMeasurement クラスのプロパティは次のとおりです。

Public Properties

	名前	説明
R	DataFetchingInterval	計測に参加するすべてのデバイスから計測データが要求される時間間隔(ミリ秒)。
R	IgnoreResync	データ収集の実行中のタイムソースの再同期を無視する必要がある場合はtrueを返します。現時点では、これはストリーミングデバイス(QuantumXファミリ)でのみ可能です。再同期は、計測の実行中にTimeSourceが変更された場合、または2つのデバイス間の時間差が大きすぎると評価された場合に行われます。デフォルト値はfalseです。値はDaqMeasurement.PrepareDaq()関数によってのみ設定できます。
R	IsRunning	データ収集が現在実行中の場合はtrueを返します。
	MeasurementStartSystemTime	このコンピュータで実行された最新の計測のローカル開始日時。この時間は、StartDaq()関数を呼び出し時にPCから取得します。
R	MeasurementStartTime	最新の計測の開始日時。この時間は、最新の計測の最初のタイムスタンプ(MeasurementStartUTCTime)をこのコンピュータのローカル日付時刻表現に変換することによって生成されます。
	MeasurementStartUTCTime	UTC形式での最新の計測の開始日時。これは、計測に参加するすべての信号の最小タイムスタンプです。形式はUTC UNIX時間(1.1.2000からの秒数、小数点以下の桁は実行中の秒)。
R	MeasurementStopSystemTime	このコンピュータで実行された最新の計測の現地の停止日時。
R	MeasurementStopTime	最新の計測の停止日時。
R	MeasurementStopUTCTime	最新の計測の終了日時をUTC形式で表示します。これは、計測に参加するすべての信号の最新のタイムスタンプです。形式はUTC UNIX時間(1.1.2000からの秒数、小数点以下の桁は実行中の秒)。
R	MeasurementTotalTime	最新の計測の合計時間(秒単位)。
	UsedDataAcquisitionMode	最新の計測を開始するために使用されたDataAcquisitionModeを返します。最新の計測がDataAcquisitionMode.Autoで開始されている場合、使用されるデータ収集モードは、選択された実際のモード(DataAcquisitionMode.Unsynchronized、DataAcquisitionMode.HardwareSynchronized、または DataAcquisitionMode.TimestampSynchronized)になります。

DaqMeasurement.DataFetchingInterval Property

計測に参加するすべてのデバイスから計測データが要求される時間間隔(ミリ秒)。

C++

```
__property int DataFetchingInterval;
```

C#

```
public int DataFetchingInterval;
```

Visual Basic

```
public property DataFetchingInterval() As Integer
```

DaqMeasurement.IgnoreResync Property

データ収集の実行中のタイムソースの再同期を無視する必要がある場合はtrueを返します。現時点では、これはストリーミングデバイス(QuantumXファミリ)でのみ可能です。再同期は、計測の実行中に TimeSourceが変更された場合、または2つのデバイス間の時間差が大きすぎると評価された場合に行われます。デフォルト値はfalseです。値はDaqMeasurement.PrepareDaq()関数によってのみ設定できます。

C++

```
__property bool IgnoreResync;
```

C#

```
public bool IgnoreResync;
```

Visual Basic

```
public property IgnoreResync() As bool
```

DaqMeasurement.IsRunning Property

データ収集が現在実行中の場合はtrueを返します。

C++

```
__property bool IsRunning;
```

C#

```
public bool IsRunning;
```

Visual Basic

```
public property IsRunning() As bool
```

DaqMeasurement.MeasurementStartSystemTime Property

このコンピュータで実行された最新の計測のローカル開始日時。この時間は、StartDaq()関数を呼び出し時にPCから取得します。

C++

```
__property DateTime MeasurementStartSystemTime;
```

C#

```
public DateTime MeasurementStartSystemTime;
```

Visual Basic

```
public property MeasurementStartSystemTime() As DateTime
```

DaqMeasurement.MeasurementStartTime Property

最新の計測の開始日時。この時間は、最新の計測の最初のタイムスタンプ(MeasurementStartUTCTime)をこのコンピュータのローカル日付時刻表現に変換することによって生成されます。

C++

```
__property DateTime MeasurementStartTime;
```

C#

```
public DateTime MeasurementStartTime;
```

Visual Basic

```
public property MeasurementStartTime() As DateTime
```

備考

これは現地の日付と時間です(地域の夏/冬時間によって異なります)!

DaqMeasurement.MeasurementStartUTCTime Property

UTC形式での最新の計測の開始日時。これは、計測に参加するすべての信号の最小タイムスタンプです。形式はUTC UNIX時間(1.1.2000からの秒数、小数点以下の桁は実行中の秒)。

C++

```
__property double MeasurementStartUTCTime;
```

C#

```
public double MeasurementStartUTCTime;
```

Visual Basic

```
public property MeasurementStartUTCTime() As double
```

備考

これは現地時間ではなく、グリニッジ標準時(GMT)であることに注意してください。

DaqMeasurement.MeasurementStopSystemTime Property

このコンピュータで実行された最新の計測の現地の停止日時。

C++

```
__property DateTime MeasurementStopSystemTime;
```

C#

```
public DateTime MeasurementStopSystemTime;
```

Visual Basic

```
public property MeasurementStopSystemTime() As DateTime
```

DaqMeasurement.MeasurementStopTime Property

最新の計測の停止日時。

C++

```
__property DateTime MeasurementStopTime;
```

C#

```
public DateTime MeasurementStopTime;
```

Visual Basic

```
public property MeasurementStopTime() As DateTime
```

備考

これは現地の日付と時間です(地域の夏/冬時間によって異なります)!

DaqMeasurement.MeasurementStopUTCTime Property

最新の計測の終了日時をUTC形式で表示します。これは、計測に参加するすべての信号の最新のタイムスタンプです。形式はUTC UNIX時間(1.1.2000からの秒数、小数点以下の桁は実行中の秒)。

C++

```
__property double MeasurementStopUTCTime;
```

C#

```
public double MeasurementStopUTCTime;
```

Visual Basic

```
public property MeasurementStopUTCTime() As double
```

備考

これは現地時間ではなく、グリニッジ標準時(GMT)であることに注意してください。

DaqMeasurement.MeasurementTotalTime Property

最新の計測の合計時間(秒単位)。

C++

```
__property double MeasurementTotalTime;
```

C#

```
public double MeasurementTotalTime;
```

Visual Basic

```
public property MeasurementTotalTime() As double
```

DaqMeasurement.UsedDataAcquisitionMode Property

最新の計測を開始するために使用されたDataAcquisitionModeを返します。最新の計測がDataAcquisitionMode.Autoで開始されている場合、使用されるデータ収集モードは、選択された実際のモード(DataAcquisitionMode.Unsynchronized、DataAcquisitionMode.HardwareSynchronized、または DataAcquisitionMode.TimestampSynchronized)になります。

C++

```
__property DataAcquisitionMode UsedDataAcquisitionMode;
```

C#

```
public DataAcquisitionMode UsedDataAcquisitionMode;
```

Visual Basic

```
public property UsedDataAcquisitionMode() As DataAcquisitionMode
```

Hbm.Api.Common.Entities Namespace

これは名前空間Hbm.Api.Common.Entitiesです。

クラス

	名前	説明
	CalibrationCertificateFileInfo	校正証明に関する情報を格納するクラス。
A	Device	単一のデバイスを表します。
	CanFd	CAN FDの構成プロパティ。
A	DeviceFamily	すべてのDeviceFamily実装の抽象基底クラス。
	DetectionRange	検出されたピークが有効となる低波長と高波長(帯域)をnm単位で定義します。
	DeviceErrorStatus	デバイスの1つのエラーステータス情報を表します。
	MeasurementValue	ReSharperはInheritdocConsiderUsageを一度無効にします。
A	DeviceStatus	デバイスの1つのステータス情報を表します。
	MeasurementValues	連続計測の計測値を保持するクラス。データ収集の実行中に計測値を読み取るには、`Hbm.API.Common.DaqMeasurement.FillMeasurementValues`を呼び出す必要があります。これにより、信号の循環バッファから信号のContinuousMeasurementValueに一定数の計測値がコピーされます。
	FirmwareUpdater	デバイスのファームウェア更新を処理するメソッドをカプセル化するクラス(HBMデバイスのみ)。
	Service	デバイスによって提供されるサービス(SCPIなど)を表します。
A	StreamingDevice	データ取得にHBMストリーミングを使用するデバイスの抽象基底クラス。
	Zero	アナログチャネルのゼロオフセット。

名前空間

名前	説明
Hbm.Api.Common.Entities.Channels	これは名前空間Hbm.Api.Common.Entities.Channelsです。
Hbm.Api.Common.Entities.ConnectionInfos	これは名前空間Hbm.Api.Common.Entities.ConnectionInfosです。
Hbm.Api.Common.Entities.Connectors	これは名前空間Hbm.Api.Common.Entities.Connectorsです。
Hbm.Api.Common.Entities.Filters	これは名前空間Hbm.Api.Common.Entities.Filtersです。
Hbm.Api.Common.Entities.Problems	これは名前空間Hbm.Api.Common.Entities.Problemsです。
Hbm.Api.Common.Entities.Scalings	これは名前空間Hbm.Api.Common.Entities.Scalingsです。
Hbm.Api.Common.Entities.Signals	これは名前空間Hbm.Api.Common.Entities.Signalsです。
Hbm.Api.Common.Entities.Statuses	これは名前空間Hbm.Api.Common.Entities.Statusesです。
Hbm.Api.Common.Entities.SpectrumInfos	これは名前空間Hbm.Api.Common.Entities.SpectrumInfosです。
Hbm.Api.Common.Entities.TimeSources	デバイスを同期すると、モジュールのクロックが可能な限り相互に同期して実行されます。Common APIは通常、タイムスタンプを使用して複数のデバイス間で同期計測の開始を実現するため、この場合は同じクロックのデバイスが必須です。 同じファミリの複数のデバイスを計測する場合、通常すべてのファミリがケーブルベースの同期方法(MGCとPMXの場合はそれぞれの同期ケーブル、QuantumXの場合はFireWireケーブル)をサポートしており、クロックが確実に同期されるため、同期に関する問題は発生しません。ただし、使用するデバイスファミリによっては、1つのデバイスのみを計測する場合でも...詳細。
Hbm.Api.Common.Entities.Values	これは名前空間Hbm.Api.Common.Entities.Valuesです。

現在位置:

Hbm.Api.Common.Entities.Channels Namespace

これは名前空間Hbm.Api.Common.Entities.Channelsです。

クラス

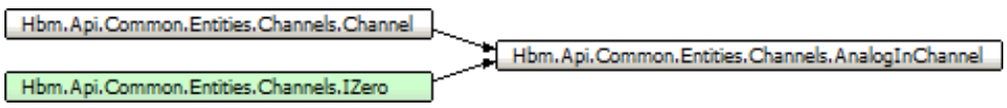
	名前	説明
	AnalogInChannel	センサを接続するチャネル。
	AnalogOutChannel	アナログ出力コネクタのチャネル。
	CanInChannel	Hbm.Api.SensorDB.Entities.Sensors.CanBusSensorによって構成されたCANコネクタの入力チャネル。
	CanRawChannel	CANコネクタ用のCANraw受信チャネル。
	Channel	センサ信号の論理図。
	DigitalChannel	デジタルコネクタ用のチャネル。
	FbgChannel	ファイバコネクタ用のチャネル。
	VirtualChannel	センサが接続されていないチャネル(演算チャネルなど)。

インタフェース

	名前	説明
	IInputSignalReference	入力信号がサポートされているすべての出力チャネルのインタフェース。
	IUnit	ユニットがセンサ/スケーリングの下に配置されていないすべてのチャネルのインタフェース。現在、このインタフェースはVirtualChannel、DigitalChannel、AnalogOutChannel、およびCanOutChannelに使用されています。すべてのチャネルタイプがこのインタフェースを実現するわけではないため、ユニットを取得するためにこのインタフェースにチャネルを直接キャストしないでください。チャネルの単位を直接設定または取得するには、Hbm.Api.Common.Entities.Device.SetUnitおよびHbm.Api.Common.Entities.Device.GetUnitを使用してください。
	IZero	ゼロ調整をサポートするすべてのチャネル タイプのインタフェース。現在、このインタフェースはAnalogInChannelとFbgChannelに使用されています。

AnalogInChannel Class

センサを接続するチャネル。



C++

```
class AnalogInChannel : public Channel, public IZero;
```

C#

```
public abstract class AnalogInChannel : Channel, IZero;
```

Visual Basic

```
public abstract Class AnalogInChannel
Inherits Channel
Inherits IZero
```

ファイル

AnalogInChannel.cs

AnalogInChannel Members

次の表は、AnalogInChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AutoCalibrationMode	このチャネルを自動的に再調整する方法を説明するモード。
	Name	チャネル名。
	Sensor	コネクタを介してデバイスに接続される物理センサ。
	Signals	接続されたセンサから配信される計測信号。
	TedsUsageMode	TEDS使用モードを取得または設定します(デバイスドライバのサポートについては、備考を参照してください)。
	Zero	このアナログ入力チャネルのゼロオフセット。

AnalogInChannel Properties

AnalogInChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	AutoCalibrationMode	このチャネルを自動的に再調整する方法を説明するモード。
	Sensor	コネクタを介してデバイスに接続される物理センサ。
	Zero	このアナログ入力チャネルのゼロオフセット。

AnalogInChannel.AutoCalibrationMode Property

このチャネルを自動的に再調整する方法を説明するモード。

C++

```
__property AutoCalibrationMode AutoCalibrationMode;
```

C#

```
public AutoCalibrationMode AutoCalibrationMode;
```

Visual Basic

```
Public Property AutoCalibrationMode() As AutoCalibrationMode
```

AnalogInChannel.Sensor Property

コネクタを介してデバイスに接続される物理センサ。

C++

```
__property Sensor Sensor;
```

C#

```
public Sensor Sensor;
```

Visual Basic

```
Public Property Sensor() As Sensor
```

AnalogInChannel.Zero Property

このアナログ入力チャネルのゼロオフセット。

C++

```
__property Zero Zero;
```

C#

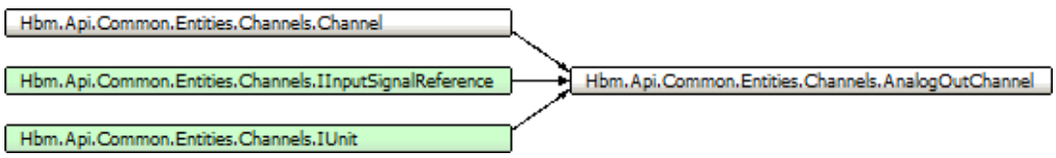
```
public Zero Zero;
```

Visual Basic

```
Public Property Zero() As Zero
```

AnalogOutChannel Class

アナログ出力コネクタのチャネル。



C++

```
class AnalogOutChannel : public Channel, public IInputSignalReference, public IUnit;
```

C#

```
public abstract class AnalogOutChannel : Channel, IInputSignalReference, IUnit;
```

Visual Basic

```
Public abstract Class AnalogOutChannel
Inherits Channel
Inherits IInputSignalReference
Inherits IUnit
```

ファイル

AnalogOutChannel.cs

AnalogOutChannel Members

次の表は、AnalogOutChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	InputModuleReference	入力信号が属するモジュールの固有の(デバイスファミリに依存)リファレンス。入力信号が存在しない場合、または入力信号ソースが同じデバイス上にある場合は、空にする必要があります。
	InputSignalReference	出力チャネルの入力信号への固有の(デバイス ファミリに依存)リファレンス。信号が別のモジュールからのものである場合は、モジュールリファレンス(InputModuleReference)も設定します。入力信号が存在しない場合は空にする必要があります。
	IsActivated	チャネルがアクティブ化されている場合はtrue を返し、そうでない場合はfalseを返します。AnalogOutChannelをアクティブまたは非アクティブにするには、関数Hbm.Api.Common.Entities.Device.SetAnalogOutChannelActivationを使用します。
	Name	チャネル名。
	Signals	接続されたセンサから配信される計測信号。
	TedsUsageMode	TEDS使用モードを取得または設定します(デバイスドライバのサポートについては、備考を参照してください)。
	Unit	チャネルの単位。

AnalogOutChannel Properties

AnalogOutChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	InputModuleReference	入力信号が属するモジュールの固有の(デバイスファミリに依存)リファレンス。入力信号が存在しない場合、または入力信号ソースが同じデバイス上にある場合は、空にする必要があります。
	InputSignalReference	出力チャネルの入力信号への固有の(デバイス ファミリに依存)リファレンス。信号が別のモジュールからのものである場合は、モジュールリファレンス(InputModuleReference)も設定します。入力信号が存在しない場合は空にする必要があります。
	IsActivated	チャネルがアクティブ化されている場合はtrue を返し、そうでない場合はfalseを返します。AnalogOutChannelをアクティブまたは非アクティブにするには、関数Hbm.Api.Common.Entities.Device.SetAnalogOutChannelActivationを使用します。
	Unit	チャネルの単位。

AnalogOutChannel.InputModuleReference Property

入力信号が属するモジュールの固有の(デバイスファミリに依存)リファレンス。入力信号が存在しない場合、または入力信号ソースが同じデバイス上にある場合は、空にする必要があります。

C++

```
__property string InputModuleReference;
```

C#

```
public string InputModuleReference;
```

Visual Basic

```
Public Property InputModuleReference() As string
```

AnalogOutChannel.InputSignalReference Property

出力チャネルの入力信号への固有の(デバイス ファミリに依存)リファレンス。信号が別のモジュールからのものである場合は、モジュールリファレンス(InputModuleReference)も設定します。入力信号が存在しない場合は空にする必要があります。

C++

```
__property string InputSignalReference;
```

C#

```
public string InputSignalReference;
```

Visual Basic

```
Public Property InputSignalReference() As string
```

AnalogOutChannel.IsActivated Property

チャネルがアクティブ化されている場合はtrue を返し、そうでない場合はfalseを返します。AnalogOutChannelをアクティブまたは非アクティブにするには、関数Hbm.Api.Common.Entities.Device.SetAnalogOutChannelActivationを使用します。

C++

```
__property bool IsActivated;
```

C#

```
public bool IsActivated;
```

Visual Basic

```
Public Property IsActivated() As bool
```

AnalogOutChannel.Unit Property

チャネルの単位。

C++

```
__property string Unit;
```

C#

```
public string Unit;
```

Visual Basic

```
Public Property Unit() As string
```

CanInChannel Class

Hbm.Api.SensorDB.Entities.Sensors.CanBusSensorによって構成されたCANコネクタの入力チャネル。



C++

```
class CanInChannel : public Channel;
```

C#

```
public abstract class CanInChannel : Channel;
```

Visual Basic

```
Public abstract Class CanInChannel
Inherits Channel
```

ファイル

CanInChannel.cs

CanInChannel Members

次の表は、CanInChannelによって公開されるメンバの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	IsActivated	チャネルを有効(または無効)にします。
	Name	チャネル名。
	Sensor	CAN bus経由で接続されたセンサのパラメータ化情報。
	Signals	接続されたセンサから配信される計測信号。
	TedsUsageMode	TEDS使用モードを取得または設定します(デバイスドライバのサポートについては、備考を参照してください)。

CanInChannel Properties

CanInChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	IsActivated	チャネルを有効(または無効)にします。
	Sensor	CAN bus経由で接続されたセンサのパラメータ化情報。

CanInChannel.IsActivated Property

チャネルを有効(または無効)にします。

C++

```
__property bool IsActivated;
```

C#

```
public bool IsActivated;
```

Visual Basic

```
Public Property IsActivated() As bool
```

CanInChannel.Sensor Property

CAN bus経由で接続されたセンサのパラメータ化情報。

C++

```
__property CanBusSensor Sensor;
```

C#

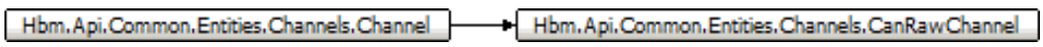
```
public CanBusSensor Sensor;
```

Visual Basic

```
Public Property Sensor() As CanBusSensor
```

CanRawChannel Class

CANコネクタ用のCANraw受信チャネル。



C++

```
class CanRawChannel : public Channel;
```

C#

```
public abstract class CanRawChannel : Channel;
```

Visual Basic

```
Public abstract Class CanRawChannel
Inherits Channel
```

ファイル

CanRawChannel.cs

CanRawChannel Members

次の表は、CanRawChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	IsActivated	チャネルを有効(または無効)にします。
	Name	チャネル名。
	Signals	接続されたセンサから配信される計測信号。
	TedsUsageMode	TEDS使用モードを取得または設定します(デバイスドライバのサポートについては、備考を参照してください)。

CanRawChannel Properties

CanRawChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	IsActivated	チャネルを有効(または無効)にします。

CanRawChannel.IsActivated Property

チャネルを有効(または無効)にします。

C++

```
__property bool IsActivated;
```

C#

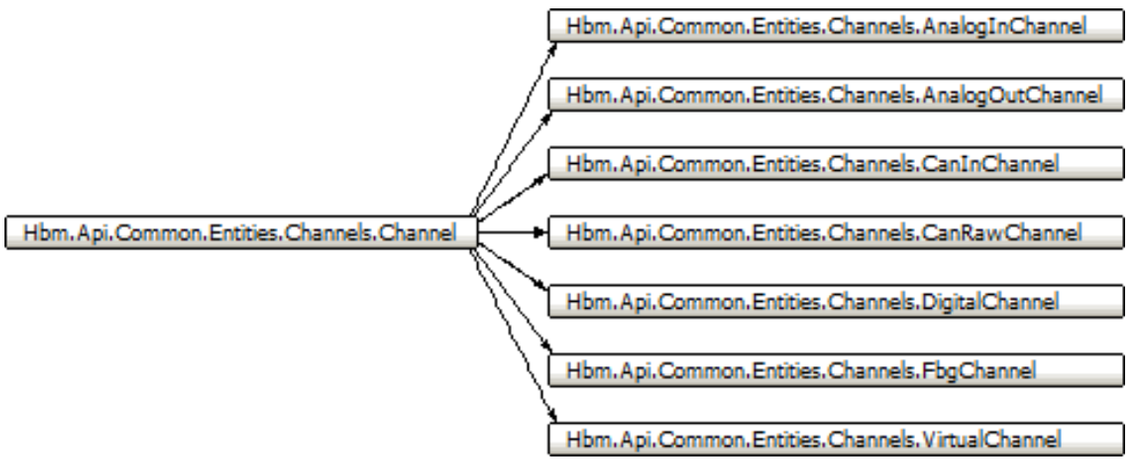
```
public bool IsActivated;
```

Visual Basic

```
Public Property IsActivated() As bool
```

Channel Class

センサ信号の論理図。



C++

```
class Channel;
```

C#

```
public abstract class Channel;
```

Visual Basic

```
Public abstract Class Channel
```

ファイル

Channel.cs

Channel Members

次の表は、Channelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Channel	クラスの新規インスタンスを作成します。
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	Name	チャネル名。
	Signals	接続されたセンサから配信される計測信号。
	TedsUsageMode	TEDS使用モードを取得または設定します(デバイスドライバのサポートについては、備考を参照してください)。

Channel.Channel Constructor

クラスの新規インスタンスを作成します。

C++

```
Channel();
```

C#

```
public Channel();
```

Visual Basic

```
Public Sub New()
```

Channel Methods

Channelクラスのメソッドは次のとおりです。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Channel.GetUniqueID Method

デバイス内のチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public abstract string GetUniqueID();
```

Visual Basic

```
Public abstract Function GetUniqueID() As string
```

戻り値

デバイス内の一意のチャネルID、または識別子が設定されていない場合は `String.Empty`。

備考

この情報は、スキャンまたは接続時に取得する必要があります。このメソッドは例外をスローしてはなりません。

Channel Properties

Channelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Name	チャネル名。
	Signals	接続されたセンサから配信される計測信号。
	TedsUsageMode	TEDS使用モードを取得または設定します(デバイスドライバのサポートについては、備考を参照してください)。

Channel.Name Property

チャネル名。

C++

```
__property string Name;
```

C#

```
public string Name;
```

Visual Basic

```
Public Property Name() As string
```

Channel.Signals Property

接続されたセンサから配信される計測信号。

C++

```
[IgnoreInTms]
__property List<Signal> Signals;
```

C#

```
[IgnoreInTms]
public List<Signal> Signals;
```

Visual Basic

```
Public Property Signals() As List<Signal>
```

Channel.TedsUsageMode Property

TEDS使用モードを取得または設定します(デバイスドライバのサポートについては、備考を参照してください)。

C++

```
[IgnoreInTms]
__property TedsUsageMode TedsUsageMode;
```

C#

```
[IgnoreInTms]
public virtual TedsUsageMode TedsUsageMode ;
```

Visual Basic

```
Public virtual Property TedsUsageMode() As TedsUsageMode
```

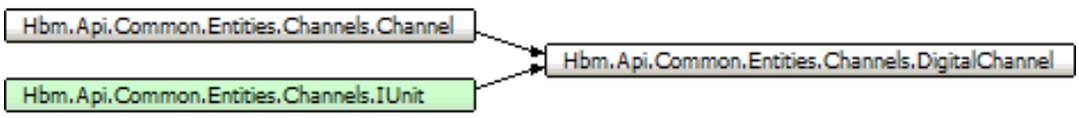
備考

デフォルトはTedsUsageMode.NotSupportedです。チャネルがTEDSをサポートする場合にオーバーライドします (例: QuantumXAnalogInChannelチャネル、MgcAnalogInChannelまたはPmxAnalogInChannel)。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Hbm.Api.Common.Entities.Channels Namespace -> Channel Class -> Channel Properties

DigitalChannel Class

デジタルコネクタ用のチャネル。



C++

```
class DigitalChannel : public Channel, public IUnit;
```

C#

```
public abstract class DigitalChannel : Channel, IUnit;
```

Visual Basic

```
Public abstract Class DigitalChannel
Inherits Channel
Inherits IUnit
```

ファイル

DigitalChannel.cs

DigitalChannel Members

次の表は、DigitalChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	Name	チャネル名。
	Signals	接続されたセンサから配信される計測信号。
	TedsUsageMode	TEDS使用モードを取得または設定します(デバイスドライバのサポートについては、備考を参照してください)。
	Unit	チャネルの単位。

DigitalChannel Properties

DigitalChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Unit	チャネルの単位。

DigitalChannel.Unit Property

チャネルの単位。

C++

```
__property string Unit;
```

C#

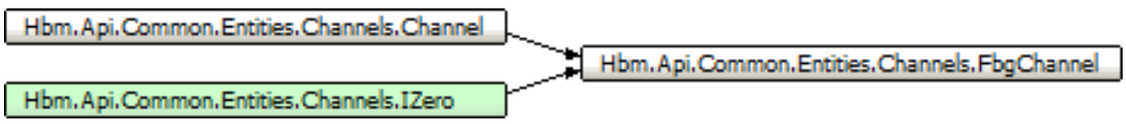
```
public string Unit;
```

Visual Basic

```
Public Property Unit() As string
```

FbgChannel Class

ファイバコネクタ用のチャネル。



C++

```
class FbgChannel : public Channel, public IZero;
```

C#

```
public abstract class FbgChannel  : Channel, IZero;
```

Visual Basic

```
Public abstract Class FbgChannel
Inherits Channel
Inherits IZero
```

ファイル

FbgChannel.cs

FbgChannel Members

次の表は、FbgChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	DetectionRange	検出されたピークが有効となる低波長と高波長(帯域)をnm単位で定義します。
	IsActivated	チャネルを有効(または無効)にします。
	Name	チャネル名。
	Sensor	デバイスに接続されたファイバ上の特定の位置に取り付けられた物理ファイバブラッググレーティングセンサ。
	Signals	接続されたセンサから配信される計測信号。
	TedsUsageMode	TEDS使用モードを取得または設定します(デバイスドライバのサポートについては、備考を参照してください)。
	Zero	このファイバブラッググレーティングチャネルのゼロオフセット。

FbgChannel Properties

FbgChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DetectionRange	検出されたピークが有効となる低波長と高波長(帯域)をnm単位で定義します。
	IsActivated	チャネルを有効(または無効)にします。
	Sensor	デバイスに接続されたファイバ上の特定の位置に取り付けられた物理ファイバブラッググレーティングセンサ。
	Zero	このファイバブラッググレーティングチャネルのゼロオフセット。

FbgChannel.DetectionRange Property

検出されたピークが有効となる低波長と高波長(帯域)をnm単位で定義します。

C++

```
__property DetectionRange DetectionRange;
```

C#

```
public DetectionRange DetectionRange;
```

Visual Basic

```
Public Property DetectionRange() As DetectionRange
```

FbgChannel.IsActivated Property

チャネルを有効(または無効)にします。

C++

```
__property bool IsActivated;
```

C#

```
public bool IsActivated;
```

Visual Basic

```
Public Property IsActivated() As bool
```

FbgChannel.Sensor Property

デバイスに接続されたファイバ上の特定の位置に取り付けられた物理ファイバ ブラッググレーティングセンサ。

C++

```
__property Sensor Sensor;
```

C#

```
public Sensor Sensor;
```

Visual Basic

```
Public Property Sensor() As Sensor
```

FbgChannel.Zero Property

このファイバブラッググレーティングチャネルのゼロオフセット。

C++

```
__property Zero Zero;
```

C#

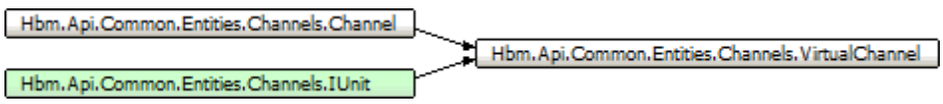
```
public Zero Zero;
```

Visual Basic

```
Public Property Zero() As Zero
```

VirtualChannel Class

センサが接続されていないチャネル(演算チャネルなど)。



C++

```
class VirtualChannel : public Channel, public IUnit;
```

C#

```
public abstract class VirtualChannel : Channel, IUnit;
```

Visual Basic

```
Public abstract Class VirtualChannel
Inherits Channel
Inherits IUnit
```

ファイル

VirtualChannel.cs

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Hbm.Api.Common.Entities.Channels Namespace

VirtualChannel Members

次の表は、VirtualChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	Name	チャネル名。
	Signals	接続されたセンサから配信される計測信号。
	TedsUsageMode	TEDS使用モードを取得または設定します(デバイスドライバのサポートについては、備考を参照してください)。
	Unit	チャネルの単位。

VirtualChannel Properties

VirtualChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Unit	チャネルの単位。

VirtualChannel.Unit Property

チャネルの単位。

C++

```
__property string Unit;
```

C#

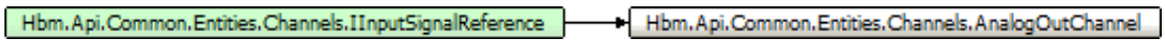
```
public string Unit;
```

Visual Basic

```
Public Property Unit() As string
```

IInputSignalReference Interface

入力信号がサポートされているすべての出力チャネルのインタフェース。



C++

```
__interface IInputSignalReference;
```

C#

```
public interface IInputSignalReference;
```

Visual Basic

```
Public Interface IInputSignalReference
```

ファイル

IInputSignalReference.cs

IInputSignalReference Members

次の表は、IInputSignalReferenceによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	InputModuleReference	入力信号が属するモジュールの固有の(デバイスファミリに依存)リファレンス。入力信号が存在しない場合、または入力信号ソースが同じデバイス上にある場合は、空にする必要があります。
	InputSignalReference	出力チャネルの入力信号への固有の(デバイス ファミリに依存)リファレンス。信号が別のモジュールからのものである場合は、モジュールリファレンス(InputModuleReference)も設定します。入力信号が存在しない場合は空にする必要があります。

IInputSignalReference Properties

IInputSignalReferenceクラスのプロパティの一覧です。

Public Properties

	名前	説明
	InputModuleReference	入力信号が属するモジュールの固有の(デバイスファミリに依存)リファレンス。入力信号が存在しない場合、または入力信号ソースが同じデバイス上にある場合は、空にする必要があります。
	InputSignalReference	出力チャネルの入力信号への固有の(デバイス ファミリに依存)リファレンス。信号が別のモジュールからのものである場合は、モジュールリファレンス(InputModuleReference)も設定します。入力信号が存在しない場合は空にする必要があります。

IInputSignalReference.InputModuleReference Property

入力信号が属するモジュールの固有の(デバイスファミリに依存)リファレンス。入力信号が存在しない場合、または入力信号ソースが同じデバイス上にある場合は、空にする必要があります。

C++

```
__property string InputModuleReference;
```

C#

```
string InputModuleReference;
```

Visual Basic

```
Property InputModuleReference() As string
```

IInputSignalReference.InputSignalReference Property

出力チャネルの入力信号への固有の(デバイス ファミリに依存)リファレンス。信号が別のモジュールからのものである場合は、モジュールリファレンス(InputModuleReference)も設定します。入力信号が存在しない場合は空にする必要があります。

C++

```
__property string InputSignalReference;
```

C#

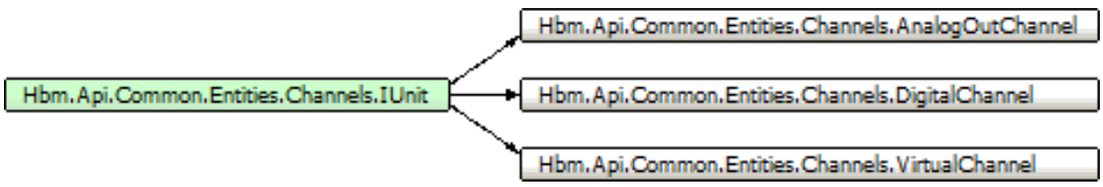
```
string InputSignalReference;
```

Visual Basic

```
Property InputSignalReference() As string
```

IUnit Interface

ユニットがセンサ/スケーリングの下に配置されていないすべてのチャネルのインタフェース。現在、このインタフェースはVirtualChannel、DigitalChannel、AnalogOutChannel、およびCanOutChannelに使用されています。すべてのチャネルタイプがこのインタフェースを実現するわけではないため、ユニットを取得するためにこのインタフェースにチャネルを直接キャストしないでください。チャネルの単位を直接設定または取得するには、Hbm.Api.Common.Entities.Device.SetUnitおよびHbm.Api.Common.Entities.Device.GetUnitを使用してください。



C++

```
__interface IUnit;
```

C#

```
public interface IUnit;
```

Visual Basic

```
Public Interface IUnit
```

ファイル

IUnit.cs

IUnit Members

次の表 は、IUnitによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	Unit	チャネルの単位。

IUnit Properties

IUnitクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Unit	チャネルの単位。

IUnit.Unit Property

チャネルの単位。

C++

```
__property string Unit;
```

C#

```
string Unit;
```

Visual Basic

```
Property Unit() As string
```

IZero Interface

ゼロ調整をサポートするすべてのチャネル タイプのインタフェース。現在、このインタフェースはAnalogInChannelとFbgChannelに使用されています。



C++

```
__interface IZero;
```

C#

```
public interface IZero;
```

Visual Basic

```
Public Interface IZero
```

ファイル

IZero.cs

IZero Members

次の表は、IZeroによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	Zero	ゼロ調整をサポートするチャネルのゼロ設定。

IZero Properties

IZeroクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Zero	ゼロ調整をサポートするチャネルのゼロ設定。

IZero.Zero Property

ゼロ調整をサポートするチャネルのゼロ設定。

C++

```
__property Zero Zero;
```

C#

```
Zero Zero;
```

Visual Basic

```
Property Zero() As Zero
```

Hbm.Api.Common.Entities.ConnectionInfos Namespace

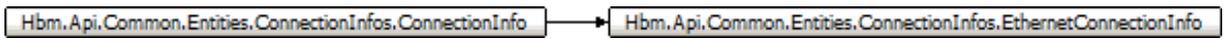
これは名前空間Hbm.Api.Common.Entities.ConnectionInfosです。

クラス

	名前	説明
A	ConnectionInfo	すべての具体的な接続情報実装の抽象基底クラス。
	EthernetConnectionInfo	デバイスのイーサネット接続に関する情報。
	StreamingConnectionInfo	ストリーミング可能なデバイスへの接続に関する情報。デバイスへの接続はイーサネット接続情報を介して実現します。HBMストリーミングを目的とする場合、追加のプロパティが必要です。

ConnectionInfo Class

すべての具体的な接続情報実装の抽象基底クラス。



C++

```
class ConnectionInfo;
```

C#

```
public abstract class ConnectionInfo;
```

Visual Basic

```
Public abstract Class ConnectionInfo
```

ファイル

ConnectionInfo.cs

ConnectionInfo Members

次の表は、ConnectionInfoによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	IsRoutedConnection	接続がCX2xデバイスを経由するかどうかを示す値を取得または設定します。

ConnectionInfo Properties

ConnectionInfoクラスのプロパティの一覧です。

Public Properties

	名前	説明
	IsRoutedConnection	接続がCX2xデバイスを経由するかどうかを示す値を取得または設定します。

ConnectionInfo.IsRoutedConnection Property

接続がCX2xデバイスを経由するかどうかを示す値を取得または設定します。

C++

```
__property bool IsRoutedConnection;
```

C#

```
public bool IsRoutedConnection;
```

Visual Basic

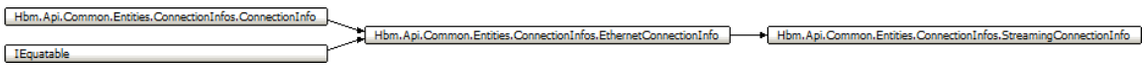
```
Public Property IsRoutedConnection() As bool
```

説明

接続インスタンスがCX2xデバイス経由の場合は`true`。それ以外の場合は`false`。

EthernetConnectionInfo Class

デバイスのイーサネット接続に関する情報。



C++

```
class EthernetConnectionInfo :publicpublic ConnectionInfo, public IEquatable<EthernetConnec
tionInfo>;
```

C#

```
public class EthernetConnectionInfo : ConnectionInfo, IEquatable<EthernetConnectionInfo>;
```

Visual Basic

```
Public Class EthernetConnectionInfo
Inherits ConnectionInfo
Inherits IEquatable
```

ファイル

EthernetConnectionInfo.cs

EthernetConnectionInfo Members

次の表は、EthernetConnectionInfoによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Equals	これは、Equalsメソッドの多重定義の概要です。
	EthernetConnectionInfo	これは、EthernetConnectionInfoコンストラクタの多重定義の概要です。
	GetHashCode	このインスタンスのハッシュ コードを返します。
	ToString	このインスタンスを表すSystem.Stringを返します。

Public Operators

	名前	説明
S	!=	演算子!=を実装します。
S	==	演算子==を実装します。

Public Properties

	名前	説明
	IpAddress	IPアドレス。ip4またはip6バージョン。
	IsRoutedConnection	接続がCX2xデバイスを経由するかどうかを示す値を取得または設定します。
	NetworkConfiguration	ネットワーク構成(dhcpまたは手動)。
	Port	hbmProtocolプロトコル(デバイスに直接接続する場合)が実行されているポート。
	SubnetMask	サブネットマスク。ip4またはip6バージョン

EthernetConnectionInfo Constructor

これは、EthernetConnectionInfoコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	EthernetConnectionInfo.EthernetConnectionInfo ()	新しいインスタンスを作成します。
	EthernetConnectionInfo.EthernetConnectionInfo (string, int)	指定したパラメータで新しいインスタンスを作成します。
	EthernetConnectionInfo.EthernetConnectionInfo (string, int, string)	指定したパラメータで新しいインスタンスを作成します。

EthernetConnectionInfo.EthernetConnectionInfo Constructor ()

新しいインスタンスを作成します。

C++

```
EthernetConnectionInfo();
```

C#

```
public EthernetConnectionInfo();
```

Visual Basic

```
Public Sub New()
```

EthernetConnectionInfo.EthernetConnectionInfo Constructor (string, int)

指定したパラメータで新しいインスタンスを作成します。

C++

```
EthernetConnectionInfo(
    string ipAddress,
    int port
);
```

C#

```
public EthernetConnectionInfo(
    string ipAddress,
    int port
);
```

Visual Basic

```
Public Sub New(
    ipAddress As string,
    port As Integer
)
```

引数

引数	説明
ipAddress	デバイスの IP アドレス。
port	デバイスに接続するポート。

EthernetConnectionInfo.EthernetConnectionInfo Constructor (string, int, string)

指定したパラメータで新しいインスタンスを作成します。

C++

```
EthernetConnectionInfo(
    string ipAddress,
    int port,
    string subnetMask
);
```

C#

```
public EthernetConnectionInfo(
    string ipAddress,
    int port,
    string subnetMask
);
```

Visual Basic

```
Public Sub New(
    ipAddress As string,
    port As Integer,
    subnetMask As string
)
```

引数

引数	説明
ipAddress	デバイスの IP アドレス。
port	デバイスに接続するポート。
subnetMask	対応する サブネットマスク 。

EthernetConnectionInfo Methods

EthernetConnectionInfoクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Equals	これは、Equalsメソッドの多重定義の概要です。
	GetHashCode	このインスタンスのハッシュ コードを返します。
	ToString	このインスタンスを表すSystem.Stringを返します。

Equals Method

これは、Equalsメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	EthernetConnectionInfo.Equals (EthernetConnectionInfo)	現在のオブジェクトが同じ型の別のオブジェクトと等しいかどうかを示します。
	EthernetConnectionInfo.Equals (object)	指定したSystem.Objectがこのインスタンスと等しいかどうかを判断します。

EthernetConnectionInfo.Equals Method (EthernetConnectionInfo)

現在のオブジェクトが同じ型の別のオブジェクトと等しいかどうかを示します。

C++

```
bool Equals(
    EthernetConnectionInfo other
);
```

C#

```
public bool Equals(
    EthernetConnectionInfo other
);
```

Visual Basic

```
public Function Equals(
    other As EthernetConnectionInfo
) As bool
```

引数

引数	説明
other	このオブジェクトと比較するオブジェクト。

戻り値

現在のオブジェクトが他の引数と等しい場合はtrue。それ以外の場合はfalse。

EthernetConnectionInfo.Equals Method (object)

指定したSystem.Objectがこのインスタンスと等しいかどうかを判断します。

C++

```
bool Equals(
    object obj
);
```

C#

```
public override bool Equals(
    object obj
);
```

Visual Basic

```
Public override Function Equals(
    obj As object
) As bool
```

引数

引数	説明
obj	このインスタンスと比較する System.Object。

戻り値

指定したSystem.Object がこのインスタンスと等しい場合は`true`。それ以外の場合は`false`。

EthernetConnectionInfo.GetHashCode Method

このインスタンスのハッシュ コードを返します。

C++

```
int GetHashCode();
```

C#

```
public override int GetHashCode();
```

Visual Basic

```
Public override Function GetHashCode() As Integer
```

戻り値

このインスタンスのハッシュ コード。ハッシュアルゴリズムやハッシュテーブルなどのデータ構造での使用に適しています。

EthernetConnectionInfo.ToString Method

このインスタンスを表すSystem.Stringを返します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

```
Public override Function ToString() As string
```

戻り値

このインスタンスを表すSystem.String。

EthernetConnectionInfo Operators

EthernetConnectionInfoクラスの演算子の一覧です。

Public Operators

	名前	説明
S	!=	演算子!=を実装します。
S	==	演算子==を実装します。

EthernetConnectionInfo.!= Operator

演算子!=を実装します。

C++

```
bool operator !=(
    EthernetConnectionInfo obj1,
    EthernetConnectionInfo obj2
) const;
```

C#

```
public static bool operator !=(
    EthernetConnectionInfo obj1,
    EthernetConnectionInfo obj2
);
```

Visual Basic

```
Public static Operator !=(
    obj1 As EthernetConnectionInfo,
    obj2 As EthernetConnectionInfo
)
```

引数

引数	説明
obj1	obj1。
obj2	obj2。

戻り値

演算子の結果。

EthernetConnectionInfo.== Operator

演算子==を実装します。

C++

```
bool operator ==(
    EthernetConnectionInfo obj1,
    EthernetConnectionInfo obj2
) const;
```

C#

```
public static bool operator ==(
    EthernetConnectionInfo obj1,
    EthernetConnectionInfo obj2
);
```

Visual Basic

```
Public static Operator ==(
    obj1 As EthernetConnectionInfo,
    obj2 As EthernetConnectionInfo
)
```

引数

引数	説明
obj1	obj1。
obj2	obj2。

戻り値

演算子の結果。

EthernetConnectionInfo Properties

EthernetConnectionInfoクラスのプロパティをここに示します。

Public Properties

	名前	説明
	IpAddress	IPアドレス。ip4またはip6バージョン。
	NetworkConfiguration	ネットワーク構成(dhcpまたは手動)。
	Port	hbmProtocolプロトコル(デバイスに直接接続する場合)が実行されているポート。
	SubnetMask	サブネットマスク。ip4またはip6バージョン。

EthernetConnectionInfo.IpAddress Property

IPアドレス。ip4またはip6バージョン。

C++

```
__property IpAddress;
```

C#

```
public string IpAddress;
```

Visual Basic

```
Public Property IpAddress As string
```

EthernetConnectionInfo.NetworkConfiguration Property

ネットワーク構成(dhcpまたは手動)。

C++

```
__property NetworkSettingsConfiguration NetworkConfiguration;
```

C#

```
public NetworkSettingsConfiguration NetworkConfiguration;
```

Visual Basic

```
Public Property NetworkConfiguration() As NetworkSettingsConfiguration
```

EthernetConnectionInfo.Port Property

hbmProtocolプロトコル(デバイスに直接接続する場合)が実行されているポート。

C++

```
__property int Port;
```

C#

```
public int Port;
```

Visual Basic

```
Public Property Port() As Integer
```

EthernetConnectionInfo.SubnetMask Property

サブネットマスク。ip4またはip6バージョン。

C++

```
__property string SubnetMask;
```

C#

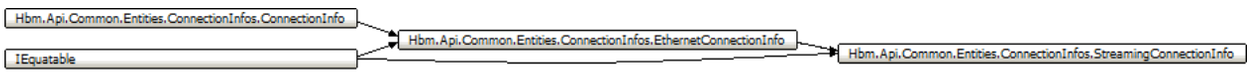
```
public string SubnetMask;
```

Visual Basic

```
Public Property SubnetMask() As string
```

StreamingConnectionInfo Class

ストリーミング可能なデバイスへの接続に関する情報。デバイスへの接続はイーサネット接続情報を介して実現します。
HBMストリーミングを目的とする場合、追加のプロパティが必要です。



C++

```
class StreamingConnectionInfo : public EthernetConnectionInfo, public IEquatable<StreamingC
onnectionInfo>;
```

C#

```
public class StreamingConnectionInfo : EthernetConnectionInfo, IEquatable<StreamingConnecti
onInfo>;
```

Visual Basic

```
Public class StreamingConnectionInfo
Inherits EthernetConnectionInfo
Inherits IEquatable
```

ファイル

StreamingConnectionInfo.cs

StreamingConnectionInfo Members

次の表は、StreamingConnectionInfoによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Equals	これは、Equalsメソッドの多重定義の概要です。
	Equals	これは、Equalsメソッドの多重定義の概要です。
	EthernetConnectionInfo	これは、EthernetConnectionInfoコンストラクタの多重定義の概要です。
	GetHashCode	このインスタンスのハッシュ コードを返します。
	StreamingConnectionInfo	これは、StreamingConnectionInfo Constructorの多重定義の概要です。
	ToString	このインスタンスを表す System.String を返します。

Public Operators

	名前	説明
S	!=	演算子!=を実装します。
S	==	演算子==を実装します。

Public Properties

	名前	説明
	HttpPort	httpプロトコル(ストリーミングで使用する)。で実行しているポート。
	IpAddress	IPアドレス。ip4またはip6バージョン。
	IsRoutedConnection	接続がCX2xデバイスを経由するかどうかを示す値を取得または設定します。
	NetworkConfiguration	ネットワーク構成(dhcpまたは手動)。
	Port	hbmProtocolプロトコル(デバイスに直接接続する場合)が実行されているポート。
	StreamingPort	daqStreamプロトコル(ストリーミングで使用する)で実行しているポート。
	SubnetMask	サブネットマスク。ip4またはip6バージョン。

StreamingConnectionInfo Constructor

これは、StreamingConnectionInfo Constructorの多重定義の概要です。

多重定義リスト

	名前	説明
	StreamingConnectionInfo.StreamingConnectionInfo Constructor ()	新しいインスタンスを作成します。
		指定したパラメータを使用して新しいインスタンスを作成します。
	StreamingConnectionInfo.StreamingConnectionInfo Constructor (string, int, int, int)	指定したパラメータを使用して新しいインスタンスを作成します。
	StreamingConnectionInfo.StreamingConnectionInfo Constructor (string, int, string)	指定したパラメータを使用して新しいインスタンスを作成します。

StreamingConnectionInfo.StreamingConnectionInfo Constructor ()

新しいインスタンスを作成します。

C++

```
StreamingConnectionInfo();
```

C#

```
public StreamingConnectionInfo;
```

Visual Basic

```
Public Sub New()
```

StreamingConnectionInfo.StreamingConnectionInfo Constructor (string, int)

指定したパラメータを使用して新しいインスタンスを作成します。

C++

```
StreamingConnectionInfo(
    string ipAddress,
    int port
);
```

C#

```
public StreamingConnectionInfo(
    string ipAddress,
    int port
);
```

Visual Basic

```
Public Sub New(
    ipAddress As string,
    port As Integer
)
```

引数

引数	説明
ipAddress	デバイスのIPアドレス。
port	デバイスを接続するための[ポート]。

StreamingConnectionInfo.StreamingConnectionInfo Constructor (string, int, int, int)

指定したパラメータを使用して新しいインスタンスを作成します。

C++

```
StreamingConnectionInfo(
    string ipAddress,
    int port,
    int streamingPort,
    int httpPort
);
```

C#

```
public StreamingConnectionInfo(
    string ipAddress,
    int port,
    int streamingPort,
    int httpPort
);
```

Visual Basic

```
Public Sub New(
    ipAddress As string,
    port As Integer,
    streamingPort As Integer,
    httpPort As Integer
)
```

引数

引数	説明
ipAddress	デバイスのIPアドレス。
port	デバイスを接続するためのポート。
streamingPort	ストリーミングポート(ストリーミングに必要)。
httpPort	HTTP ポート(ストリーミングに必要)。

StreamingConnectionInfo.StreamingConnectionInfo Constructor (string, int, string)

指定したパラメータを使用して新しいインスタンスを作成します。

C++

```
StreamingConnectionInfo(
    string ipAddress,
    int port,
    string subnetMask
);
```

C#

```
public StreamingConnectionInfo(
    string ipAddress,
    int port,
    string subnetMask
);
```

Visual Basic

```
Public Sub New(
    ipAddress As string,
    port As Integer,
    subnetMask As string
)
```

引数

引数	説明
ipAddress	デバイスのIPアドレス。
port	デバイスを接続するためのポート。
subnetMask	ネットワークのサブネットマスク。

StreamingConnectionInfo Methods

StreamingConnectionInfoクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Equals	これは、Equalsメソッドの多重定義の概要です。
	GetHashCode	このインスタンスのハッシュ コードを返します。
	ToString	このインスタンスを表す System.String を返します。

Equals Method

これは、Equalsメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	StreamingConnectionInfo.Equals (object)	指定したSystem.Objectがこのインスタンスと等しいかどうかを判断します。
	StreamingConnectionInfo.Equals (StreamingConnectionInfo)	現在のオブジェクトが同じ型の別オブジェクトと等しいかどうかを示します。

StreamingConnectionInfo.Equals Method (object)

指定したSystem.Objectがこのインスタンスと等しいかどうかを判断します。

C++

```
bool Equals(
    object obj
);
```

C#

```
public override bool Equals(
    object obj
);
```

Visual Basic

```
Public override Function Equals(
    obj As object
) As bool
```

引数

引数	説明
obj	このインスタンスと比較するSystem.Object。

StreamingConnectionInfo.Equals Method (StreamingConnectionInfo)

現在のオブジェクトが同じ型の別オブジェクトと等しいかどうかを示します。

C++

```
bool Equals(
    StreamingConnectionInfo other
);
```

C#

```
public bool Equals(
    StreamingConnectionInfo other
);
```

Visual Basic

```
Public Function Equals(
    other As StreamingConnectionInfo
) As bool
```

引数

引数	説明
other	このオブジェクトと比較するオブジェクト。

戻り値

現在のオブジェクトが他のパラメータと等しい場合はtrue。それ以外の場合はfalse。

StreamingConnectionInfo.GetHashCode Method

このインスタンスのハッシュ コードを返します。

C++

```
int GetHashCode();
```

C#

```
public override int GetHashCode();
```

Visual Basic

```
Public override Function GetHashCode() As Integer
```

戻り値

このインスタンスのハッシュコード。ハッシュアルゴリズムやハッシュテーブルなどのデータ構造での使用に適しています。

StreamingConnectionInfo.ToString Method

このインスタンスを表す System.String を返します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

```
Public override Function ToString() As string
```

戻り値

このインスタンスを表すSystem.String。

StreamingConnectionInfo Operators

StreamingConnectionInfoクラスの演算子の一覧です。

Public Operators

	名前	説明
S	!=	演算子!=を実装します。
S	==	演算子==を実装します。

StreamingConnectionInfo.!= Operator

演算子!=を実装します。

C++

```
bool operator !=(
    StreamingConnectionInfo obj1,
    StreamingConnectionInfo obj2
) const;
```

C#

```
public static bool operator !=(
    StreamingConnectionInfo obj1,
    StreamingConnectionInfo obj2
);
```

Visual Basic

```
Public static operator !=(
    obj1 As StreamingConnectionInfo,
    obj2 As StreamingConnectionInfo
)
```

引数

引数	説明
obj1	obj1。
obj2	obj2。

戻り値

演算子の結果。

StreamingConnectionInfo.== Operator

演算子==を実装します。

C++

```
bool operator ==(
    StreamingConnectionInfo obj1,
    StreamingConnectionInfo obj2
) const;
```

C#

```
public static bool operator ==(
    StreamingConnectionInfo obj1,
    StreamingConnectionInfo obj2
);
```

Visual Basic

```
Public static operator ==(
    obj1 As StreamingConnectionInfo,
    obj2 As StreamingConnectionInfo
)
```

引数

引数	説明
obj1	obj1。
obj2	obj2。

戻り値

演算子の結果。

StreamingConnectionInfo Properties

StreamingConnectionInfoクラスのプロパティの一覧です。

Public Properties

	名前	説明
	HttpPort	httpプロトコル(ストリーミングで使用する)。で実行しているポート。
	StreamingPort	daqStreamプロトコル(ストリーミングで使用する)で実行しているポート。

StreamingConnectionInfo.HttpPort Property

httpプロトコル(ストリーミングで使用する)で実行しているポート。

C++

```
__property int HttpPort;
```

C#

```
public int HttpPort;
```

Visual Basic

```
Public property HttpPort() As Integer
```

StreamingConnectionInfo.StreamingPort Property

daqStreamプロトコル(ストリーミングで使用する)で実行しているポート。

C++

```
__property int StreamingPort;
```

C#

```
public int StreamingPort;
```

Visual Basic

```
Public property StreamingPort() As Integer
```

Hbm.Api.Common.Entities.Connectors Namespace

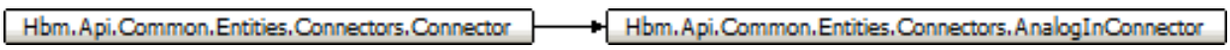
これは名前空間Hbm.Api.Common.Entities.Connectorsです。

クラス

	名前	説明
	AnalogInConnector	センサを接続できるコネクタを表します。
	AnalogOutConnector	アナログ出力コネクタを表します。
	CanConnector	CAN Busを接続できるコネクタを表します。
	Connector	物理デバイスのコネクタを表します。
	DigitalConnector	デジタル入力または出力コネクタを表します(Directionプロパティの値による)。
	FbgConnector	ファイバを接続できるコネクタを表します。
	VirtualConnector	物理センサを接続できないコネクタを表します。

AnalogInConnector Class

センサを接続できるコネクタを表します。



C++

```
class AnalogInConnector : public Connector;
```

C#

```
public abstract class AnalogInConnector : Connector;
```

Visual Basic

```
Public abstract class AnalogInConnector
Inherits Connector
```

ファイル

AnalogInConnector.cs

AnalogOutConnector Class

アナログ出力コネクタを表します。



C++

```
class AnalogOutConnector : public Connector;
```

C#

```
public abstract class AnalogOutConnector : Connector;
```

Visual Basic

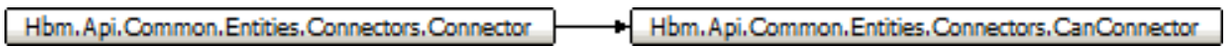
```
Public abstract class AnalogOutConnector
Inherits Connector
```

ファイル

AnalogOutConnector.cs

CanConnector Class

CAN Busを接続できるコネクタを表します。



C++

```
class CanConnector : public Connector;
```

C#

```
public abstract class CanConnector : Connector;
```

Visual Basic

```
Public abstract class CanConnector
Inherits Connector
```

ファイル

CanConnector.cs

CanConnector Members

次の表は、CanConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	BitRate	CAN Busのビットレート[bit/s]。例: 50000、100000、250000。
	CanFd	CAN FD構成プロパティを取得または設定します。CANコネクタがCAN FDをサポートしていない場合はNULL。
	Channels	接続されたセンサ(CANなど)の論理ビュー。
	ErrorResetDelayTime	エラー解決後にエラー状態がリセットされるまでの遅延時間を取得または設定します。
	ErrorResetMode	エラー状態の取得またはリセットを設定します。
	IsListenOnly	CAN Busのリッスン専用モードを有効(または無効)にします。
	IsTerminated	このコネクタのバス終端を有効(または無効)にします。
	LocationHint	人間が判読できるコネクタ位置に関する説明。
	SamplePointRatio	サンプルポイント比率を取得または設定します。
	ShortLocationHint	人間が判読できるコネクタ位置に関する短い説明。
	SortableLocationHint	目的別ののコネクタ位置。
	SupplyVoltage	供給電圧(電力)[V]。
	SynchronizationJumpWidth	同期ジャンプ幅を取得または設定します。

CanConnector Properties

Public Properties

	名前	説明
	BitRate	CAN Busのビットレート[bit/s]。例: 50000、100000、250000。
	CanFd	CAN FD構成プロパティを取得または設定します。CANコネクタがCAN FDをサポートしていない場合はNULL。
	Channels	接続されたセンサ(CANなど)の論理ビュー。
	ErrorResetDelayTime	エラー解決後にエラー状態がリセットされるまでの遅延時間を取得または設定します。
	ErrorResetMode	エラー状態の取得またはリセットを設定します。
	IsListenOnly	CAN Busのリッスン専用モードを有効(または無効)にします。
	IsTerminated	このコネクタのバス終端を有効(または無効)にします。
	SamplePointRatio	サンプルポイント比率を取得または設定します。
	SupplyVoltage	供給電圧(電力)[V]。
	SynchronizationJumpWidth	同期ジャンプ幅を取得または設定します。

CanConnector.BitRate Property

CAN Busのビットレート[bit/s]。例: 50000、100000、250000。

C++

```
__property int BitRate;
```

C#

```
public int BitRate;
```

Visual Basic

```
Public Property BitRate() As Integer
```

CanConnector.CanFd Property

CAN FD構成プロパティを取得または設定します。CANコネクタがCAN FDをサポートしていない場合はNULL。

C++

```
__property CanFd CanFd;
```

C#

```
public CanFd CanFd;
```

Visual Basic

```
Public Property CanFd() As CanFd
```

CanConnector.ErrorResetDelayTime Property

エラー解決後にエラー状態がリセットされるまでの遅延時間を取得または設定します。

C++

```
__property int ErrorResetDelayTime;
```

C#

```
public int ErrorResetDelayTime;
```

Visual Basic

```
Public Property ErrorResetDelayTime() As Integer
```

備考

このプロパティは、ErrorResetModeがHbm.Api.Common.Enums.ErrorResetMode: Delayedに設定されている場合に使用でき、それ以外の場合は無視されます。

CanConnector.ErrorResetMode Property

エラー状態の取得またはリセットを設定します。

C++

```
__property ErrorResetMode ErrorResetMode;
```

C#

```
public ErrorResetMode ErrorResetMode;
```

Visual Basic

```
Public Property ErrorResetMode() As ErrorResetMode
```

CanConnector.IsListenOnly Property

CAN Busのリッスン専用モードを有効(または無効)にします。

C++

```
__property bool IsListenOnly;
```

C#

```
public bool IsListenOnly;
```

Visual Basic

```
Public Property IsListenOnly() As bool
```

CanConnector.IsTerminated Property

このコネクタのバス終端を有効(または無効)にします。

C++

```
__property bool IsTerminated;
```

C#

```
public bool IsTerminated;
```

Visual Basic

```
Public Property IsTerminated() As bool
```

CanConnector.SamplePointRatio Property

サンプルポイント比率を取得または設定します。

C++

```
__property decimal SamplePointRatio;
```

C#

```
public decimal SamplePointRatio;
```

Visual Basic

```
Public Property SamplePointRatio() As decimal
```

説明

サンプルポイント比率。

CanConnector.SupplyVoltage Property

供給電圧(電力)[V]。

C++

```
__property decimal SupplyVoltage;
```

C#

```
public decimal SupplyVoltage;
```

Visual Basic

```
Public Property SupplyVoltage() As decimal
```

CanConnector.SynchronizationJumpWidth Property

同期ジャンプ幅を取得または設定します。

C++

```
__property int SynchronizationJumpWidth;
```

C#

```
public int SynchronizationJumpWidth;
```

Visual Basic

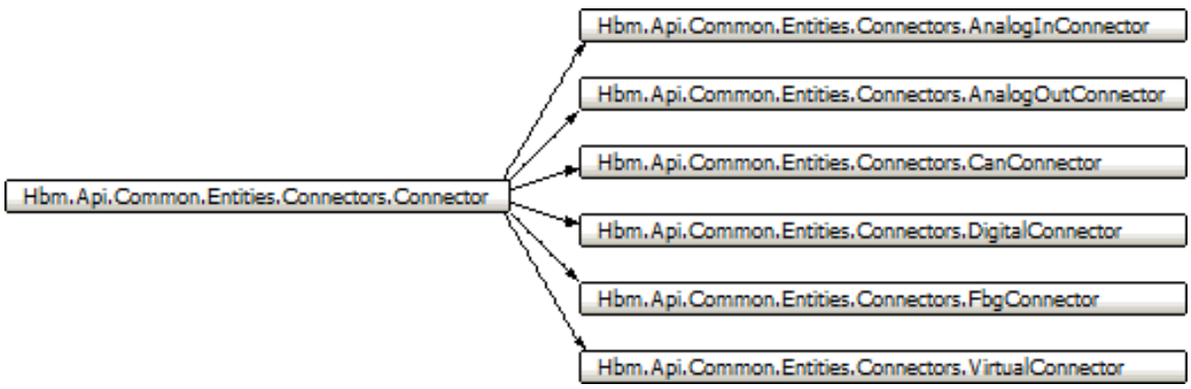
```
Public Property SynchronizationJumpWidth() As Integer
```

説明

同期ジャンプ幅。

Connector Class

物理デバイスのコネクタを表します。



C++

```
class Connector;
```

C#

```
public abstract class Connector;
```

Visual Basic

```
Public abstract Class Connector
```

ファイル

Connector.cs

Connector Members

次の表は、Connectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Connector	クラスの新しいインスタンスを作成し、必要な変数を初期化します。
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	Channels	接続されたセンサ(CANなど)の論理ビュー。
	LocationHint	人間が判読できるコネクタ位置に関する説明。
	ShortLocationHint	人間が判読できるコネクタ位置に関する短い説明。
	SortableLocationHint	目的別ののコネクタ位置。

Connector.Connector Constructor

クラスの新しいインスタンスを作成し、必要な変数を初期化します。

C++

```
Connector();
```

C#

```
public Connector();
```

Visual Basic

```
Public Sub New()
```

Connector Methods

次の表は、Connectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Connector.GetUniqueID Method

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public abstract string GetUniqueID();
```

Visual Basic

```
Public abstract Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタID、またはIDが設定されていない場合は`String.Empty`。

備考

この情報は、スキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはいけません。

Connector Properties

次の表は、Connectorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Channels	接続されたセンサ(CANなど)の論理ビュー。
	LocationHint	人間が判読できるコネクタ位置に関する説明。
	ShortLocationHint	人間が判読できるコネクタ位置に関する短い説明。
	SortableLocationHint	目的別ののコネクタ位置。

Connector.Channels Property

接続されたセンサ(CANなど)の論理ビュー。

C++

```
[IgnoreInTms]
__property List<Channel> Channels;
```

C#

```
[IgnoreInTms]
public List<Channel> Channels;
```

Visual Basic

```
<IgnoreInTms>
Public Property Channels() As List<Channel>
```

Connector.LocationHint Property

人間が判読できるコネクタ位置に関する説明。

C++

```
__property string LocationHint;
```

C#

```
public string LocationHint;
```

Visual Basic

```
Public Property LocationHint() As string
```

Connector.ShortLocationHint Property

人間が判読できるコネクタ位置に関する短い説明。

C++

```
__property string ShortLocationHint;
```

C#

```
public string ShortLocationHint;
```

Visual Basic

```
Public Property ShortLocationHint() As string
```

Connector.SortableLocationHint Property

目的別ののコネクタ位置。

C++

```
__property string SortableLocationHint;
```

C#

```
public string SortableLocationHint;
```

Visual Basic

```
Public Property SortableLocationHint() As string
```

DigitalConnector Class

デジタル入力または出力コネクタを表します(Directionプロパティの値による)。



C++

```
class DigitalConnector : public Connector;
```

C#

```
public abstract class DigitalConnector : Connector;
```

Visual Basic

```
Public abstract class DigitalConnector
Inherits Connector
```

ファイル

DigitalConnector.cs

DigitalConnector Members

次の表は、DigitalConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	Channels	接続されたセンサ(CANなど)の論理ビュー。
	Direction	コネクタの方向。
	LocationHint	人間が判読できるコネクタ位置に関する説明。
	ShortLocationHint	人間が判読できるコネクタ位置に関する短い説明。
	SortableLocationHint	目的別ののコネクタ位置。

DigitalConnector Properties

次の表は、DigitalConnectorによって公開されるプロパティの一覧です。

Public Properties

	名前	説明
	Direction	コネクタの方向。

DigitalConnector.Direction Property

コネクタの方向。

C++

```
__property DirectionType Direction;
```

C#

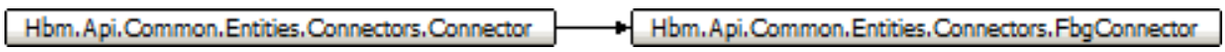
```
public DirectionType Direction;
```

Visual Basic

```
Public Property Direction() As DirectionType
```

FbgConnector Class

ファイバを接続できるコネクタを表します。



C++

```
class FbgConnector : public Connector;
```

C#

```
public abstract class FbgConnector : Connector;
```

Visual Basic

```
Public abstract Class FbgConnector
Inherits Connector
```

ファイル

FbgConnector.cs

FbgConnector Members

次の表は、FbgConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	Channels	接続されたセンサ(CANなど)の論理ビュー。
	DetectionThreshold	このファイバーに沿ったピーク検出のための相対しきい値を取得または設定します。
	LocationHint	人間が判読できるコネクタ位置に関する説明。
	ShortLocationHint	人間が判読できるコネクタ位置に関する短い説明。
	SortableLocationHint	目的別ののコネクタ位置。

FbgConnector Properties

次の表は、FbgConnectorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DetectionThreshold	このファイバーに沿ったピーク検出のための相対しきい値を取得または設定します。

FbgConnector.DetectionThreshold Property

このファイバーに沿ったピーク検出のための相対しきい値を取得または設定します。

C++

```
__property double DetectionThreshold;
```

C#

```
public double DetectionThreshold;
```

Visual Basic

```
Public Property DetectionThreshold() As double
```

説明

検出しきい値。

VirtualConnector Class

物理センサを接続できないコネクタを表します。



C++

```
class VirtualConnector : public Connector;
```

C#

```
public abstract class VirtualConnector : Connector;
```

Visual Basic

```
Public abstract Class VirtualConnector
Inherits Connector
```

ファイル

VirtualConnector.cs

Hbm.Api.Common.Entities.Filters Namespace

これは名前空間Hbm.Api.Common.Entities.Filtersです。

クラス

	名前	説明
	BesselFilter	Bessel IIR(無限インパルス応答)フィルタは、群遅延が最大平坦(線形位相の応答最大値)である線形フィルタの一種です。これらはバンドパス全体にわたってほぼ一定の群遅延を特徴とし、バンドパス内のフィルタリングされた信号の波形を維持します。
	ButterworthFilter	Butterworth IIR(無限インパルス応答)フィルタは信号処理フィルタの一種で、バンドパス内で可能な限り平坦な周波数応答になるように設計されています。これは、最大平坦型フィルタとも呼ばれます。
	ButterworthFirFilter	Butterworth FIR(有限インパルス応答)フィルタは信号処理フィルタの一種で、バンドパス内で可能な限り平坦な周波数応答になるように設計されています。これは、最大平坦型フィルタとも呼ばれます。
A	Filter	フィルタはノイズの多い実験データを平滑化するために使用します。選択した法則に応じて、フィルタリングされた出力の振幅と時間依存の動作が異なります。
	HighpassFilter	高周波信号を通過させますが、カットオフ周波数よりも低い周波数の信号の振幅を減少させるフィルタ。
	LinearPhaseFilter	線形位相FIR(有限インパルス応答)フィルタは、信号処理フィルタの一種です。
	NoFilter	フィルタを無効にするため、フィルタを使用する必要はありません。
	NotSupportedFilter	サポートされていないフィルタは、デバイスに不明なフィルタが設定されていることを示すために使用します。

BesselFilter Class

Bessel IIR(無限インパルス応答)フィルタは、最大限に平坦群遅延(線形位相の応答最大値)を備えた線形フィルタの一種です。これらはバンドパス全体にわたってほぼ一定の群遅延を特徴とし、バンドパス内のフィルタリングされた信号の波形を維持します。



C++

```
class BesselFilter : public Filter;
```

C#

```
public class BesselFilter : Filter;
```

Visual Basic

```
Public Class BesselFilter
Inherits Filter
```

ファイル

BesselFilter.cs

BesselFilter Members

次の表は、BesselFilterによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現(例: Bessel, 10Hz)に変換します。

Public Properties

	名前	説明
	Frequency	フィルタ周波数。

BesselFilter Methods

BesselFilterクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現(例: Bessel, 10Hz)に変換します。

BesselFilter.ToString Method

フィルタの名前と周波数を人間が判読できる文字列表現(例: Bessel, 10Hz)に変換します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

```
Public override Function ToString() As string
```

戻り値

変換および連結されたフィルタ文字列。

ButterworthFilter Class

Butterworth IIR(無限インパルス応答)フィルタは信号処理フィルタの一種で、バンドパス内で可能な限り平坦な周波数応答を持つように設計されています。これは、最大平坦型フィルタとも呼ばれます。



C++

```
[Serializable]
class ButterworthFilter : public Filter;
```

C#

```
[Serializable]
public class ButterworthFilter : Filter;
```

Visual Basic

```
<Serializable>
Public Class ButterworthFilter
Inherits Filter
```

ファイル

ButterworthFilter.cs

ButterworthFilter Members

次の表は、ButterworthFilterによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現(例: Butterworth, 10Hz)に変換します。

Public Properties

	名前	説明
	Frequency	フィルタ周波数。

ButterworthFilter Methods

BesselFilterクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現(例: Butterworth, 10Hz)に変換します。

ButterworthFilter.ToString Method

フィルタの名前と周波数を人間が判読できる文字列表現(例: Butterworth, 10Hz)に変換します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

```
Public override Function ToString() As string
```

戻り値

変換および連結されたフィルタ文字列。

ButterworthFirFilter Class

Butterworth FIR(有限インパルス応答)フィルタは信号処理フィルタの一種で、バンドパス内で可能な限り平坦な周波数応答になるように設計されています。これは、最大平坦型フィルタとも呼ばれます。



C++

```
[Serializable]
class ButterworthFirFilter : public Filter;
```

C#

```
[Serializable]
public class ButterworthFirFilter : Filter;
```

Visual Basic

```
<Serializable>
Public Class ButterworthFirFilter
Inherits Filter
```

ファイル

ButterworthFirFilter.cs

ButterworthFirFilter Members

次の表は、ButterworthFilterによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現(例: Butterworth FIR, 10Hz)に変換します。

Public Properties

	名前	説明
	Frequency	フィルタ周波数。

ButterworthFirFilter Methods

ButterworthFirFilterクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現(例: Butterworth FIR, 10Hz)に変換します。

ButterworthFirFilter.ToString Method

フィルタの名前と周波数を人間が判読できる文字列表現(例: Butterworth FIR, 10Hz)に変換します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

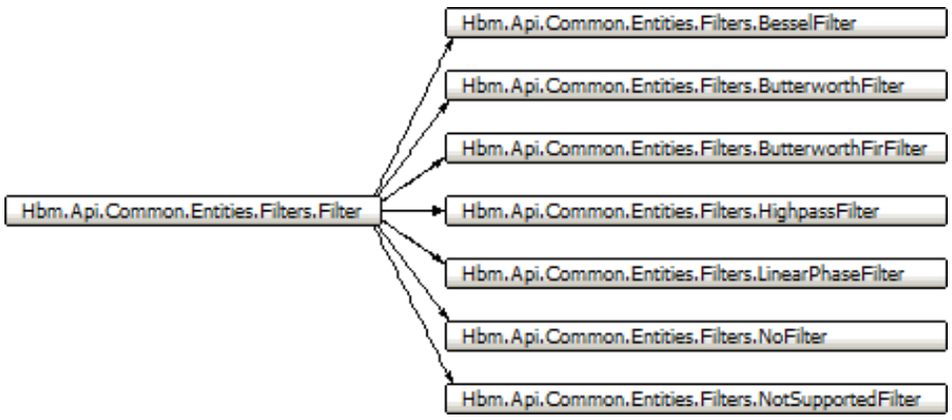
```
Public override Function ToString() As string
```

戻り値

変換および連結されたフィルタ文字列。

Filter Class

フィルタはノイズの多い実験データを平滑化するために使用します。選択した法則に応じて、フィルタリングされた出力の振幅と時間依存の動作が異なります。



C++

```
[Serializable]
class Filter;
```

C#

```
[Serializable]
public abstract class Filter;
```

Visual Basic

```
<Serializable>
Public abstract Class Filter
```

ファイル

Filter.cs

Filter Members

次の表は、Filterによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	Frequency	フィルタ周波数。

Filter Properties

Filterクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Frequency	フィルタ周波数。

Filter.Frequency Property

フィルタ周波数。

C++

```
__property decimal Frequency;
```

C#

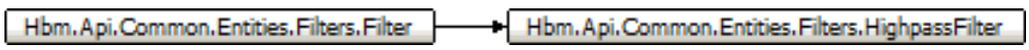
```
public decimal Frequency;
```

Visual Basic

```
Public Property Frequency() As decimal
```

HighpassFilter Class

高周波信号を通過させますが、カットオフ周波数よりも低い周波数の信号の振幅を減少させるフィルタ。



C++

```
[Serializable]
class HighpassFilter : public Filter;
```

C#

```
[Serializable]
public class HighpassFilter : Filter;
```

Visual Basic

```
<Serializable>
Public Class HighpassFilter
Inherits Filter
```

ファイル

HighpassFilter.cs

HighpassFilter Members

次の表は、HighpassFilterによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現(例: Highpass, 5 Hz)に変換します。

Public Properties

	名前	説明
	Frequency	フィルタ周波数。

HighpassFilter Methods

HighpassFilterクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現(例: Highpass, 5 Hz)に変換します。

HighpassFilter.ToString Method

フィルタの名前と周波数を人間が判読できる文字列表現(例: Highpass, 5 Hz)に変換します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

```
Public override Function ToString() As string
```

戻り値

変換および連結されたフィルタ文字列。

LinearPhaseFilter Class

線形位相FIR(有限インパルス応答)フィルタは、信号処理フィルタの一種です。



C++

```
[Serializable]
class LinearPhaseFilter : public Filter;
```

C#

```
[Serializable]
public class LinearPhaseFilter : Filter;
```

Visual Basic

```
<Serializable>
Public Class LinearPhaseFilter
Inherits Filter
```

ファイル

LinearPhaseFilter.cs

LinearPhaseFilter Members

次の表は、LinearPhaseFilterによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現(例: LinearPhase, 50Hz)に変換します。

Public Properties

	名前	説明
	Frequency	フィルタ周波数。

LinearPhaseFilter Methods

HighpassFilterクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現(例: LinearPhase, 50Hz)に変換します。

LinearPhaseFilter.ToString Method

フィルタの名前と周波数を人間が判読できる文字列表現(例: LinearPhase, 50Hz)に変換します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

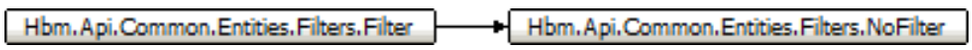
```
Public override Function ToString() As string
```

戻り値

変換および連結されたフィルタ文字列。

NoFilter Class

フィルタを無効にするため、フィルタを使用する必要はありません。



C++

```
[Serializable]
class NoFilter : public Filter;
```

C#

```
[Serializable]
public class NoFilter : Filter;
```

Visual Basic

```
<Serializable>
Public Class NoFilter
Inherits Filter
```

ファイル

NoFilter.cs

NoFilter Members

次の表は、NoFilterによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	NoFilter	新しいインスタンスを作成します。
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現に変換します。

Public Properties

	名前	説明
	Frequency	フィルタ周波数。

NoFilter.NoFilter Constructor

新しいインスタンスを作成します。

C++

```
NoFilter();
```

C#

```
public NoFilter();
```

Visual Basic

```
Public Sub New()
```

NoFilter Methods

NoFilterクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現に変換します。

NoFilter.ToString Method

フィルタの名前と周波数を人間が判読できる文字列表現に変換します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

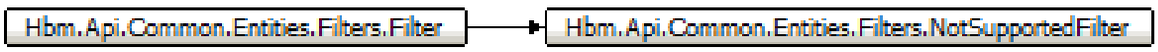
```
Public override Function ToString() As string
```

戻り値

フィルタ文字列。

NotSupportedFilter Class

サポートされていないフィルタは、デバイスに不明なフィルタが設定されていることを示すために使用します。



C++

```
[Serializable]
class NotSupportedFilter : public Filter;
```

C#

```
[Serializable]
public class NotSupportedFilter : Filter;
```

Visual Basic

```
<Serializable>
Public Class NotSupportedFilter
Inherits Filter
```

ファイル

NotSupportedFilter.cs

NotSupportedFilter Members

次の表は、NotSupportedFilterによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	NotSupportedFilter	新しいインスタンスを作成します。
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現に変換します。

Public Properties

	名前	説明
	Frequency	フィルタ周波数。

NotSupportedFilter.NotSupportedFilter Constructor

新しいインスタンスを作成します。

C++

```
NotSupportedFilter();
```

C#

```
public NotSupportedFilter();
```

Visual Basic

```
Public Sub New()
```

NotSupportedFilter Methods

NotSupportedFilterクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ToString	フィルタの名前と周波数を人間が判読できる文字列表現に変換します。

NotSupportedFilter.ToString Method

フィルタの名前と周波数を人間が判読できる文字列表現に変換します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

```
Public override Function ToString() As string
```

戻り値

フィルタ文字列。

Hbm.Api.Common.Entities.Problems Namespace

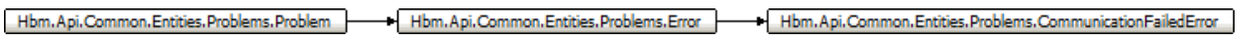
これは名前空間Hbm.Api.Common.Entities.Problemsです。

クラス

	名前	説明
	CommunicationFailedError	デバイスの使用中に発生した通信障害エラーを記述するクラス。
	ConnectionNotSupportedError	デバイス上でDaqEnvironment.Find(string, List{ConnectionInfo}, out List{Problem})中に発生した接続がサポートされていないエラーを記述するクラス。
	DeviceAlreadyConnectedWarning	デバイスの使用中に発生した警告を記述するクラス。
	Error	デバイスの使用中に発生したエラーを記述するクラス。
	ExtendedSampleRateModeError	デバイスの使用中に発生した拡張サンプルレートモードエラーを記述するクラス。
	FirmwareUpdateFailedError	ファームウェア更新時のエラーを記述するクラス。
	InvalidPropertyOfDeviceError	デバイスに間違ったプロパティが設定されている場合に、デバイスエラーを記述するクラス。
	OperationFailedError	デバイスの使用中に発生した操作失敗エラーを記述するクラス。
A	Problem	デバイスの使用中に発生した問題を記述するクラス。
S	ProblemExtension	Hbm.Api.Common.Entities.Shoulder.Problem型のオブジェクトに追加のメソッドを提供する拡張クラス。
S	ProblemListExtension	List<Problem>型の問題リストに対するヘルパとしての拡張メソッド。
	TedsActivationWarning	TEDSセンサの起動中に発生した警告を記述するクラス。
	TedsNotAvailableWarning	TEDS がコネクタに接続されていないという警告を記述するクラス。
	TedsNotSupportedWarning	コネクタがTEDSをサポートしていないという警告を記述するクラス。
	TedsReadingWarning	TEDSの読み取り中に発生した警告を記述するクラス。
	TedsWritingWarning	TEDSの書き込み中に発生した警告を記述するクラス。
	TypeNotSupportedError	デバイスの使用中に発生した型がサポートされていないエラーを記述するクラス。
	UnexpectedError	デバイスの使用中に発生した予期しないエラーを記述するクラス。
	ValueAdaptedWarning	特定の値の適応を記述するクラス。
	ValueNotAssignedWarning	割り当てが完了していないことを示すために使用されるクラス。現時点では、これはNotSupportedセンサの割り当て中にのみ使用されます。この場合、代入関数は呼び出されていますが、代入は行われません。
	ValueNotSetError	デバイスの使用中に発生した、値が設定されていないエラーを記述するクラス。
	ValueOutOfRangeError	デバイスの使用中に発生した範囲外エラーを記述するクラス。
	Warning	デバイスの使用中に発生した警告を記述するクラス。

CommunicationFailedError Class

デバイスの使用中に発生した通信障害エラーを記述するクラス。



C++

```
class CommunicationFailedError : public Error;
```

C#

```
public class CommunicationFailedError : Error;
```

Visual Basic

```
Public Class CommunicationFailedError
Inherits Error
```

ファイル

CommunicationFailedError.cs

CommunicationFailedError Members

次の表は、CommunicationFailedErrorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommunicationFailedError	これは、CommunicationFailedErrorコンストラクタの多重定義の概要です。
	Error	これは、Errorコンストラクタの多重定義の概要です。
	Problem	これは、Problemコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	問題を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

CommunicationFailedError Constructor

これは、CommunicationFailedErrorコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	CommunicationFailedError.CommunicationFailedError (Device, Channel, SettingType, Exception, string)	コンストラクタ。
	CommunicationFailedError.CommunicationFailedError (Device, Connector, SettingType, Exception, string)	コンストラクタ。
	CommunicationFailedError.CommunicationFailedError (Device, Sensor, SettingType, Exception, string)	コンストラクタ。
	CommunicationFailedError.CommunicationFailedError (Device, SettingType, Exception, string)	コンストラクタ。
	CommunicationFailedError.CommunicationFailedError (Device, SettingType, string)	コンストラクタ。
	CommunicationFailedError.CommunicationFailedError (Device, Signal, SettingType, Exception, string)	コンストラクタ。

CommunicationFailedError.CommunicationFailedError Constructor (Device, Channel, SettingType, Exception, string)

コンストラクタ。

C++

```
CommunicationFailedError(
    Device device,
    Channel channel,
    SettingType settingType,
    Exception exception = null,
    string propertyName = null
);
```

C#

```
public CommunicationFailedError(
    Device device,
    Channel channel,
    SettingType settingType,
    Exception exception,
    string propertyName
);
```

Visual Basic

```
Public Sub  New(
    device As Device,
    channel As Channel,
    settingType As SettingType,
    exception As Exception = null,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。
exception	エラーの原因となった例外。
propertyName	エラーの原因となったプロパティの名前。

CommunicationFailedError.CommunicationFailedError Constructor (Device, Connector, SettingType, Exception, string)

コンストラクタ。

C++

```
CommunicationFailedError(
    Device device,
    Connector connector,
    SettingType settingType,
    Exception exception = null,
    string propertyName = null
);
```

C#

```
public CommunicationFailedError(
    Device device,
    Connector connector,
    SettingType settingType,
    Exception exception,
    string propertyName
);
```

Visual Basic

```
Public Sub  New(
    device As Device,
    Connector As Connector,
    settingType As SettingType,
    exception As Exception = null,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。
exception	エラーの原因となった例外。
propertyName	エラーの原因となったプロパティの名前。

CommunicationFailedError.CommunicationFailedError Constructor (Device, Sensor, SettingType, Exception, string)

コンストラクタ。

C++

```
CommunicationFailedError(
    Device device,
    Sensor sensor,
    SettingType settingType,
    Exception exception = null,
    string propertyName = null
);
```

C#

```
public CommunicationFailedError(
    Device device,
    Sensor sensor,
    SettingType settingType,
    Exception exception,
    string propertyName
);
```

Visual Basic

```
Public Sub  New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType,
    exception As Exception = null,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。
exception	エラーの原因となった例外。
propertyName	エラーの原因となったプロパティの名前。

CommunicationFailedError.CommunicationFailedError Constructor (Device, SettingType, Exception, string)

コンストラクタ。

C++

```
CommunicationFailedError(
    Device device,
    SettingType settingType,
    Exception exception,
    string propertyName = null
);
```

C#

```
public CommunicationFailedError(
    Device device,
    SettingType settingType,
    Exception exception,
    string propertyName
);
```

Visual Basic

```
Public Sub  New(
    device As Device,
    settingType As SettingType,
    exception As Exception,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。
exception	エラーの原因となった例外。
propertyName	エラーの原因となったプロパティの名前。

CommunicationFailedError.CommunicationFailedError Constructor (Device, SettingType, string)

コンストラクタ。

C++

```
CommunicationFailedError(
    Device device,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public CommunicationFailedError(
    Device device,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub  New(
    device As Device,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

CommunicationFailedError.CommunicationFailedError Constructor (Device, Signal, SettingType, Exception, string)

コンストラクタ。

C++

```
CommunicationFailedError(
    Device device,
    signal signal,
    SettingType settingType,
    Exception exception = null,
    string propertyName = null
);
```

C#

```
public CommunicationFailedError(
    Device device,
    signal signal,
    SettingType settingType,
    Exception exception,
    string propertyName
);
```

Visual Basic

```
Public Sub  New(
    device As Device,
    signal As signal,
    settingType As SettingType,
    exception As Exception = null,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。
exception	エラーの原因となった例外。
propertyName	エラーの原因となったプロパティの名前。

CommunicationFailedError Properties

CommunicationFailedErrorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	エラーを記述するメッセージ。

CommunicationFailedError.Message Property

問題を記述するメッセージ。

C++

```
__property string Message;
```

C#

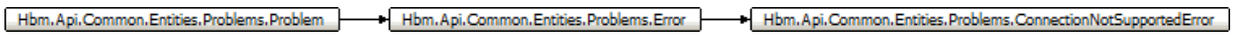
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

ConnectionNotSupportedError Class

デバイス上でDaqEnvironment.Find(string, List{ConnectionInfo}, out List{Problem})中に発生した接続がサポートされていないエラーを記述するクラス。



C++

```
class ConnectionNotSupportedError  : public Error;
```

C#

```
public class ConnectionNotSupportedError : Error;
```

Visual Basic

```
Public Class ConnectionNotSupportedError
Inherits Error
```

ファイル

ConnectionNotSupportedError.cs

関連するトピック

Hbm.Api.Common.Entities.Problems.Error

ConnectionNotSupportedError Members

次の表は、ConnectionNotSupportedErrorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ConnectionNotSupportedError	ConnectionNotSupportedErrorクラスの新しいインスタンスを初期化します。
	Error	これは、Errorコンストラクタの多重定義の概要です。
	Problem	これは、Problemコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectionInfo	接続を確立できなかったConnectionInfoを取得または設定します。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	エラーを記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

ConnectionNotSupportedError.ConnectionNotSupportedError Constructor

ConnectionNotSupportedErrorクラスの新しいインスタンスを初期化します。

C++

```
ConnectionNotSupportedError(
    Device device,
    SettingType settingType,
    ConnectionInfo connectionInfo
);
```

C#

```
public ConnectionNotSupportedError(
    Device device,
    SettingType settingType,
    ConnectionInfo connectionInfo
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType,
    connectionInfo As ConnectionInfo
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。
connectionInfo	接続情報。

ConnectionNotSupportedError Properties

CommunicationFailedErrorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ConnectionInfo	接続を確立できなかったConnectionInfoを取得または設定します。
R	Message	エラーを記述するメッセージ。

ConnectionNotSupportedError.ConnectionInfo Property

接続を確立できなかったConnectionInfoを取得または設定します。

C++

```
__property ConnectionInfo ConnectionInfo;
```

C#

```
public ConnectionInfo ConnectionInfo;
```

Visual Basic

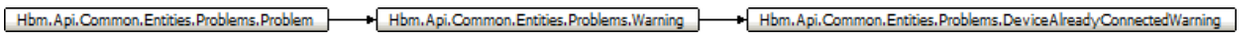
```
Public Property ConnectionInfo() As ConnectionInfo
```

説明

接続情報。

DeviceAlreadyConnectedWarning Class

デバイスの使用中に発生した警告を記述するクラス。



C++

```
class DeviceAlreadyConnectedWarning : public Warning;
```

C#

```
public class DeviceAlreadyConnectedWarning : Warning;
```

Visual Basic

```
Public Class DeviceAlreadyConnectedWarning
Inherits Warning
```

ファイル

DeviceAlreadyConnectedWarning.cs

DeviceAlreadyConnectedWarning Members

次の表は、DeviceAlreadyConnectedWarningによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DeviceAlreadyConnectedWarning	コンストラクタ。
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	Warning	これは、Warningコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectionInfo	接続を確立できなかったConnectionInfoを取得または設定します。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
R V	Message	問題を記述するメッセージ。
R	Message	エラーを記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

DeviceAlreadyConnectedWarning.DeviceAlreadyConnectedWarning Constructor

コンストラクタ。

C++

```
DeviceAlreadyConnectedWarning(
    Device device
);
```

C#

```
public DeviceAlreadyConnectedWarning(
    Device device
);
```

Visual Basic

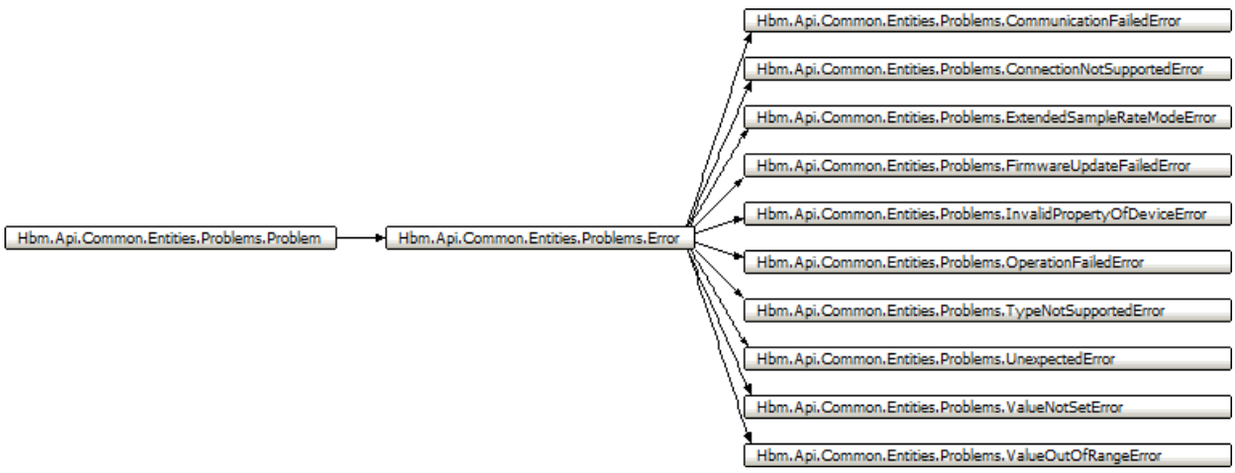
```
Public Sub New(
    device As Device
)
```

引数

引数	説明
device	エラーの原因となったデバイス。

Error Class

デバイスの使用中に発生したエラーを記述するクラス。



C++

```
class Error : public Problem;
```

C#

```
public class Error : Problem;
```

Visual Basic

```
Public Class Error
Inherits Problem
```

ファイル

Error.cs

Error Members

次の表は、Errorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Error	これは、Errorコンストラクタの多重定義の概要です。
	Problem	これは、Problemコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	エラーを記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

Error Constructor

これは、Errorコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	Error.Error (Device, Channel, SettingType, string)	コンストラクタ。
	Error.Error (Device, Connector, SettingType, string)	コンストラクタ。
	Error.Error (Device, Sensor, SettingType, string)	コンストラクタ。
	Error.Error (Device, SettingType, string)	コンストラクタ。
	Error.Error (Device, Signal, SettingType, string)	コンストラクタ。

Error.Error Constructor (Device, Channel, SettingType, string)

コンストラクタ。

C++

```
Error(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public Error(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

Error.Error Constructor (Device, Connector, SettingType, string)

コンストラクタ。

C++

```
Error(
    Device device,
    connector connector,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public Error(
    Device device,
    connector connector,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As connector,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

Error.Error Constructor (Device, Sensor, SettingType, string)

コンストラクタ。

C++

```
Error(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public Error(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

Error.Error Constructor (Device, SettingType, string)

コンストラクタ。

C++

```
Error(
    Device device,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public Error(
    Device device,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

Error.Error Constructor (Device, Signal, SettingType, string)

コンストラクタ。

C++

```
Error(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public Error(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

Error Properties

Errorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	エラーを記述するメッセージ。

Error.Exception Property

デバイスとの対話中に発生した予期しない例外。

C++

```
__property Exception Exception;
```

C#

```
public Exception Exception;
```

Visual Basic

```
Public Property Exception() As Exception
```

Error.Message Property

エラーを記述するメッセージ。

C++

```
__property string Message;
```

C#

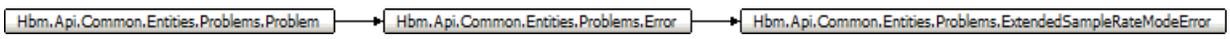
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

ExtendedSampleRateModeError Class

デバイスの使用中に発生した拡張サンプルレートモードエラーを記述するクラス。



C++

```
class ExtendedSampleRateModeError : public Error;
```

C#

```
public class ExtendedSampleRateModeError : Error;
```

Visual Basic

```
Public Class ExtendedSampleRateModeError
Inherits Error
```

ファイル

ExtendedSampleRateModeError.cs

ExtendedSampleRateModeError Members

次の表は、ExtendedSampleRateModeErrorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Error	これは、Errorコンストラクタの多重定義の概要です。
	ExtendedSampleRateModeError	これは、ExtendedSampleRateModeErrorコンストラクタの多重定義の概要です。
	Problem	これは、Problemコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	ErrorCodeKey	問題のエラー コード キーを取得します。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	エラーを記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

ExtendedSampleRateModeError Constructor

これは、ExtendedSampleRateModeErrorコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	ExtendedSampleRateModeError.ExtendedSampleRateModeError (Device, Connector, string)	コンストラクタ。
	ExtendedSampleRateModeError.ExtendedSampleRateModeError (Device, string)	コンストラクタ。

ExtendedSampleRateModeError.ExtendedSampleRateModeError Constructor (Device, Connector, string)

コンストラクタ。

C++

```
ExtendedSampleRateModeError(
    Device device,
    Connector connector,
    string errorCodeKey
);
```

C#

```
public ExtendedSampleRateModeError(
    Device device,
    Connector connector,
    string errorCodeKey
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    errorCodeKey As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
errorCodeKey	エラーコードキー。

ExtendedSampleRateModeError.ExtendedSampleRateModeError Constructor (Device, string)

コンストラクタ。

C++

```
ExtendedSampleRateModeError(
    Device device,
    string errorCodeKey
);
```

C#

```
public ExtendedSampleRateModeError(
    Device device,
    string errorCodeKey
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    errorCodeKey As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
errorCodeKey	エラーコードキー。

ExtendedSampleRateModeError Properties

ExtendedSampleRateModeErrorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ErrorCodeKey	問題のエラー コード キーを取得します。
R	Message	エラーを記述するメッセージ。

ExtendedSampleRateModeError.ErrorCodeKey Property

問題のエラー コード キーを取得します。

C++

```
__property string ErrorCodeKey;
```

C#

```
public string ErrorCodeKey;
```

Visual Basic

```
Public Property ErrorCodeKey() As string
```

ExtendedSampleRateModeError.Message Property

エラーを記述するメッセージ。

C++

```
__property string Message;
```

C#

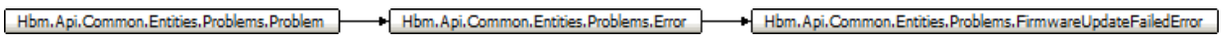
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

FirmwareUpdateFailedError Class

ファームウェア更新時のエラーを記述するクラス。



C++

```
class FirmwareUpdateFailedError : public Error;
```

C#

```
public class FirmwareUpdateFailedError : Error;
```

Visual Basic

```
Public Class FirmwareUpdateFailedError
Inherits Error
```

ファイル

FirmwareUpdateFailedError.cs

FirmwareUpdateFailedError Members

次の表は、FirmwareUpdateFailedErrorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Error	これは、Errorコンストラクタの多重定義の概要です。
	FirmwareUpdateFailedError	コンストラクタ。
	Problem	これは、Problemコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	エラーを記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

FirmwareUpdateFailedError.FirmwareUpdateFailedError Constructor

コンストラクタ。

C++

```
FirmwareUpdateFailedError(
    Device device,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public FirmwareUpdateFailedError(
    Device device,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

FirmwareUpdateFailedError Properties

FirmwareUpdateFailedErrorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	エラーを記述するメッセージ。

FirmwareUpdateFailedError.Message Property

エラーを記述するメッセージ。

C++

```
__property string Message;
```

C#

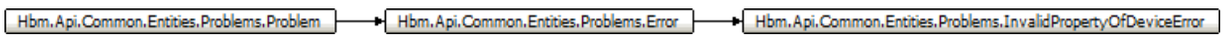
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

InvalidPropertyOfDeviceError Class

デバイスに間違ったプロパティが設定されている場合に、デバイスエラーを記述するクラス。



C++

```
class InvalidPropertyOfDeviceError : public Error;
```

C#

```
public class InvalidPropertyOfDeviceError : Error;
```

Visual Basic

```
Public Class InvalidPropertyOfDeviceError
Inherits Error
```

ファイル

InvalidPropertyOfDeviceError.cs

InvalidPropertyOfDeviceError Members

次の表は、InvalidPropertyOfDeviceErrorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Error	これは、Errorコンストラクタの多重定義の概要です。
	InvalidPropertyOfDeviceError	コンストラクタ。
	Problem	これは、Problemコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	エラーを記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

InvalidPropertyOfDeviceError.InvalidPropertyOfDeviceError Constructor

コンストラクタ。

C++

```
InvalidPropertyOfDeviceError(
    Device device,
    SettingType settingType
);
```

C#

```
public InvalidPropertyOfDeviceError(
    Device device,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。

InvalidPropertyOfDeviceError Properties

FirmwareUpdateFailedErrorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	エラーを記述するメッセージ。

InvalidPropertyOfDeviceError.Message Property

エラーを記述するメッセージ。

C++

```
__property string Message;
```

C#

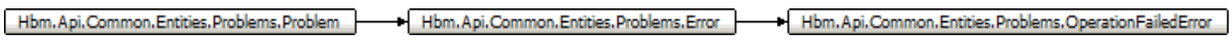
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

OperationFailedError Class

デバイスの使用中に発生した操作失敗エラーを記述するクラス。



C++

```
class OperationFailedError : public Error;
```

C#

```
public class OperationFailedError : Error;
```

Visual Basic

```
Public Class OperationFailedError
Inherits Error
```

ファイル

OperationFailedError.cs

OperationFailedError Members

次の表は、OperationFailedErrorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Error	これは、Errorコンストラクタの多重定義の概要です。
	OperationFailedError	これは、OperationFailedErrorコンストラクタの多重定義の概要です。
	Problem	これは、Problemコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	エラーを記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

OperationFailedError Constructor

これは、OperationFailedErrorコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	OperationFailedError.OperationFailedError (Device, Channel, SettingType, string)	コンストラクタ。
	OperationFailedError.OperationFailedError (Device, Connector, SettingType, string)	コンストラクタ。
	OperationFailedError.OperationFailedError (Device, Sensor, SettingType, string)	コンストラクタ。
	OperationFailedError.OperationFailedError (Device, SettingType, string)	コンストラクタ。
	OperationFailedError.OperationFailedError (Device, Signal, SettingType, string)	コンストラクタ。

OperationFailedError.OperationFailedError Constructor (Device, Channel, SettingType, string)

コンストラクタ。

C++

```
OperationFailedError(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public OperationFailedError(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

OperationFailedError.OperationFailedError Constructor (Device, Connector, SettingType, string)

コンストラクタ。

C++

```
OperationFailedError(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public OperationFailedError(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

OperationFailedError.OperationFailedError Constructor (Device, Sensor, SettingType, string)

コンストラクタ。

C++

```
OperationFailedError(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public OperationFailedError(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

OperationFailedError.OperationFailedError Constructor (Device, SettingType, string)

コンストラクタ。

C++

```
OperationFailedError(
    Device device,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public OperationFailedError(
    Device device,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

OperationFailedError.OperationFailedError Constructor (Device, Signal, SettingType, string)

コンストラクタ。

C++

```
OperationFailedError(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public OperationFailedError(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
Signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

OperationFailedError Properties

OperationFailedErrorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	エラーを記述するメッセージ。

OperationFailedError.Message Property
エラーを記述するメッセージ。

C++

```
__property string Message;
```

C#

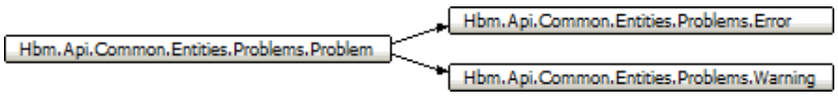
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

Problem Class

デバイスの使用中に発生した問題を記述するクラス。



C++

```
class Problem;
```

C#

```
public abstract class Problem;
```

Visual Basic

```
Public abstract Class Problem
```

ファイル

Problem.cs

Problem Members

次の表は、Problemによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Problem	これは、Problemコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
R V	Message	問題を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

Problem Constructor

これは、Problemコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	Problem.Problem (Device, Channel, SettingType, string)	コンストラクタ。
	Problem.Problem (Device, Connector, SettingType, string)	コンストラクタ。
	Problem.Problem (Device, Sensor, SettingType, string)	コンストラクタ。
	Problem.Problem (Device, SettingType, string)	コンストラクタ。
	Problem.Problem (Device, Signal, SettingType, string)	コンストラクタ。

Problem.Problem Constructor (Device, Channel, SettingType, string)

コンストラクタ。

C++

```
Problem(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName
);
```

C#

```
public Problem(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType,
    propertyName As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

Problem.Problem Constructor (Device, Connector, SettingType, string)

コンストラクタ。

C++

```
Problem(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName
);
```

C#

```
public Problem(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType,
    propertyName As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

Problem.Problem Constructor (Device, Sensor, SettingType, string)

コンストラクタ。

C++

```
Problem(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName
);
```

C#

```
public Problem(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType,
    propertyName As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

Problem.Problem Constructor (Device, SettingType, string)

コンストラクタ。

C++

```
Problem(
    Device device,
    SettingType settingType,
    string propertyName
);
```

C#

```
public Problem(
    Device device,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType,
    propertyName As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

Problem.Problem Constructor (Device, Signal, SettingType, string)

コンストラクタ。

C++

```
Problem(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName
);
```

C#

```
public Problem(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType,
    propertyName As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

Problem Properties

Problemクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
R V	Message	問題を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

Problem.ChannelUniqueID Property

問題の原因となったチャネルの一意の識別子。

C++

```
__property string ChannelUniqueID;
```

C#

```
public string ChannelUniqueID;
```

Visual Basic

```
Public Property ChannelUniqueID() As string
```

Problem.ConnectorUniqueID Property

問題の原因となったコネクタの一意の識別子。

C++

```
__property string ConnectorUniqueID;
```

C#

```
public string ConnectorUniqueID;
```

Visual Basic

```
Public Property ConnectorUniqueID() As string
```

Problem.DemandedValue Property

問題の原因となった値。

C++

```
__property string DemandedValue;
```

C#

```
public string DemandedValue;
```

Visual Basic

```
Public Property DemandedValue() As string
```

Problem.DeviceUniqueID Property

問題の原因となったデバイスの一意の識別子。

C++

```
__property string DeviceUniqueID;
```

C#

```
public string DeviceUniqueID;
```

Visual Basic

```
Public Property DeviceUniqueID() As string
```

Problem.Message Property

問題を記述するメッセージ。

C++

```
__property string Message;
```

C#

```
public virtual string Message;
```

Visual Basic

```
Public virtual Property Message() As string
```

Problem.OriginalContext Property

デバイスによって生成された問題の前後関係。

C++

```
__property string OriginalContext;
```

C#

```
public string OriginalContext;
```

Visual Basic

```
Public Property OriginalContext() As string
```

Problem.OriginalErrorCode Property

デバイスが生成した元のエラー。

C++

```
__property string OriginalErrorCode;
```

C#

```
public string OriginalErrorCode;
```

Visual Basic

```
Public Property OriginalErrorCode() As string
```

Problem.OriginalMessage Property

デバイスが生成した元の問題(エラー)メッセージ。

C++

```
__property string OriginalMessage;
```

C#

```
public string OriginalMessage;
```

Visual Basic

```
Public Property OriginalMessage() As string
```

Problem.Position Property

問題が発生した場所。

C++

```
__property string Position;
```

C#

```
public string Position;
```

Visual Basic

```
Public Property Position() As string
```

Problem.PropertyName Property

問題の原因となったプロパティの名前。

C++

```
__property string PropertyName;
```

C#

```
public string PropertyName;
```

Visual Basic

```
Public Property PropertyName() As string
```

Problem.SensorUniqueID Property

問題の原因となったセンサの一意の識別子。

C++

```
__property string SensorUniqueID;
```

C#

```
public string SensorUniqueID;
```

Visual Basic

```
Public Property SensorUniqueID() As string
```

Problem.SettingType Property

問題の原因となるオブジェクトの型。

C++

```
__property SettingType SettingType;
```

C#

```
public SettingType SettingType;
```

Visual Basic

```
Public Property SettingType() As SettingType
```

Problem.SignalUniqueID Property
問題の原因となった信号の一意の識別子。

C++

```
__property string SignalUniqueID;
```

C#

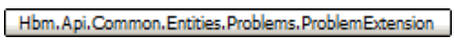
```
public string SignalUniqueID;
```

Visual Basic

```
Public Property SignalUniqueID() As string
```

ProblemExtension Class

Hbm.Api.Common.Entities.Shoulder.Problem型のオブジェクトに追加のメソッドを提供する拡張クラス。



C++

```
class ProblemExtension;
```

C#

```
public static class ProblemExtension;
```

Visual Basic

```
Public static Class ProblemExtension
```

ファイル

ProblemExtension.cs

ProblemExtension Members

次の表は、ProblemExtensionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
S	BuildReadableString	問題を判読可能な文字列を作成します。
S	Log	問題を指定したロガーに記録します。問題にエラーが含まれている場合、問題はLogLevel.Errorレベルに記録され、それ以外の場合はLogLevel.Warningに記録されます。

ProblemExtension Methods

ProblemExtensionクラスのメソッドの一覧です。

Public Methods

	名前	説明
S	BuildReadableString	問題を判読可能な文字列を作成します。
S	Log	問題を指定したロガーに記録します。問題にエラーが含まれている場合、問題はLogLevel.Errorレベルに記録され、それ以外の場合はLogLevel.Warningに記録されます。

ProblemExtension.BuildReadableString Method

問題を判読可能な文字列を作成します。

C++

```
string BuildReadableString(
    this Problem problem
);
```

C#

```
public static string BuildReadableString(
    this Problem problem
);
```

Visual Basic

```
Public static Function BuildReadableString(
    problem As this Problem
) As string
```

引数

引数	説明
problem	問題。

戻り値

問題をフォーマットした文字列。

ProblemExtension.Log Method

問題を指定したロガーに記録します。問題にエラーが含まれている場合、問題はLogLevel.Errorレベルに記録され、それ以外の場合はろおお l .Warningに記録されます。

C++

```
void Log(
    this Problem problem,
    ILogger logger,
    LogContext logContext = null,
    string problemSummary = null
);
```

C#

```
public static void Log(
    this Problem problem,
    ILogger logger,
    LogContext logContext,
    string problemSummary
);
```

Visual Basic

```
Public static Function Log(
    problem As this Problem,
    logger As ILogger,
    logContext As LogContext = null,
    problemSummary As string = null
) As void
```

引数

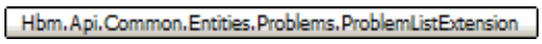
引数	説明
problem	メソッドを呼び出す対象となる問題。
logger	ログオンするためのロガー。
logContext	オプション: コンテキストをログに記録します。
problemSummary	オプション: 問題の概要。

例外

引数	説明
ArgumentNullException	ロガーがnullの場合にスローされます。

ProblemListExtension Class

List<Problem>型の問題リストに対するヘルパとしての拡張メソッド。



C++

```
class ProblemListExtension;
```

C#

```
public static class ProblemListExtension;
```

Visual Basic

```
Public static Class ProblemListExtension
```

ファイル

ProblemListExtension.cs

備考

使用するリストがnullの場合、拡張メソッドは自動的にSystem.NullReferenceExceptionをスローするため、引数のチェックは必要ありません。

ProblemListExtension Members

次の表は、ProblemListExtensionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
S	BuildReadableString	問題を判読可能な文字列を作成します。
S	BuildXmlString	問題からXML文字列を作成します。
S	HasError	問題リストに1つ以上のエラーが含まれているかどうかを確認します。
S	HasWarning	問題リストに1つ以上の警告が含まれているかどうかを確認します。
S	IsEmpty	問題リストが空かどうかを確認します。
S	Log	問題を指定したロガーに記録します。問題に1つ以上のエラーが含まれている場合、問題はLogLevel.Errorレベルに記録され、それ以外の場合はLogLevel.Warningに記録されます。

ProblemListExtension Methods

ProblemListExtension クラスのメソッドの一覧です。

Public Methods

	名前	説明
S	BuildReadableString	問題を判読可能な文字列を作成します。
S	BuildXmlString	問題からXML文字列を作成します。
S	HasError	問題リストに1つ以上のエラーが含まれているかどうかを確認します。
S	HasWarning	問題リストに1つ以上の警告が含まれているかどうかを確認します。
S	IsEmpty	問題リストが空かどうかを確認します。
S	Log	問題を指定したロガーに記録します。問題に1つ以上のエラーが含まれている場合、問題はLogLevel.Errorレベルに記録され、それ以外の場合はLogLevel.Warningに記録されます。

ProblemListExtension.BuildReadableString Method

問題を判読可能な文字列を作成します。

C++

```
string BuildReadableString(
    this List<Problem> problems
);
```

C#

```
public static string BuildReadableString(
    this List<Problem> problems
);
```

Visual Basic

```
Public static Function BuildReadableString(
    problems As this List<Problem>
) As string
```

引数

引数	説明
problems	メソッドが呼び出される問題のリストを指定します。

戻り値

問題をフォーマットした文字列。

ProblemListExtension.BuildXmlString Method

問題からXML文字列を作成します。

C++

```
string BuildXmlString(
    this List<Problem> problems
);
```

C#

```
public static string BuildXmlString(
    this List<Problem> problems
);
```

Visual Basic

```
Public static Function BuildXmlString(
    problems As this List<Problem>
) As string
```

引数

引数	説明
problems	メソッドが呼び出される問題のリストを指定します。

戻り値

XML形式の問題のリスト。

ProblemListExtension.HasError Method

問題リストに1つ以上のエラーが含まれているかどうかを確認します。

C++

```
bool HasError(
    this List<Problem> problemlist
);
```

C#

```
public static bool HasError(
    this List<Problem> problemlist
);
```

Visual Basic

```
Public static Function HasError(
    problemlist As this List<Problem>
) As bool
```

引数

引数	説明
problemlist	メソッドが呼び出される問題のリストを指定します。

戻り値

問題のリストに1つ以上のエラーが含まれている場合は`true`、それ以外の場合は`false`。

ProblemListExtension.HasWarning Method

問題リストに1つ以上の警告が含まれているかどうかを確認します。

C++

```
bool HasWarning(
    this List<Problem> problemlist
);
```

C#

```
public static bool HasWarning(
    this List<Problem> problemlist
);
```

Visual Basic

```
Public static Function HasWarning(
    problemlist As this List<Problem>
) As bool
```

引数

引数	説明
problemlist	メソッドが呼び出される問題のリストを指定します。

戻り値

問題のリストに1つ以上の警告が含まれている場合は`true`、それ以外の場合は`false`。

ProblemListExtension.IsEmpty Method

問題リストが空かどうかを確認します。

C++

```
bool IsEmpty(
    this List<Problem> problemlist
);
```

C#

```
public static bool IsEmpty(
    this List<Problem> problemlist
);
```

Visual Basic

```
Public static Function IsEmpty(
    problemlist As this List<Problem>
) As bool
```

引数

引数	説明
problemlist	メソッドが呼び出される問題のリストを指定します。

戻り値

問題のリストが空の場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	problemlistが`null`です。

ProblemListExtension.Log Method

問題を指定したロガーに記録します。問題に1つ以上のエラーが含まれている場合、問題はLogLevel.Errorレベルに記録され、それ以外の場合はLogLevel.Warningに記録されます。

C++

```
void Log(
    this List<Problem> problems,
    ILogger logger,
    LogContext logContext = null,
    string problemSummary = null
);
```

C#

```
public static void Log(
    this List<Problem> problems,
    ILogger logger,
    LogContext logContext,
    string problemsSummary
);
```

Visual Basic

```
Public static Function Log(
    problems As this List<Problem>,
    logger As ILogger,
    logContext As LogContext = null,
    problemsSummary As string = null
) As void
```

引数

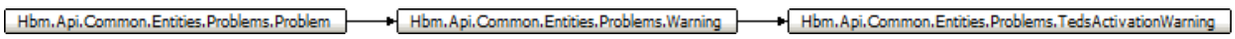
引数	説明
problems	メソッドが呼び出される問題のリストを指定します。
logger	ログオンするためのロガー。
logContext	オプション: コンテキストをログに記録します。
problemsSummary	オプション: 問題の概要。

例外

引数	説明
ArgumentNullException	ロガーがnullの場合にスローされます。

TedsActivationWarning Class

TEDSセンサの起動中に発生した警告を記述するクラス。



C++

```
class TedsActivationWarning : public Warning;
```

C#

```
public class TedsActivationWarning : Warning;
```

Visual Basic

```
Public Class TedsActivationWarning
Inherits Warning
```

ファイル

TedsActivationWarning.cs

TedsActivationWarning Members

次の表は、TedsActivationWarningによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	TedsActivationWarning	これは、TedsActivationWarningコンストラクタの多重定義の概要です。
	Warning	これは、Warningコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
R	Message	警告を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

TedsActivationWarning Constructor

これは、TedsActivationWarningコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	TedsActivationWarning.TedsActivationWarning (Device, Channel, SettingType)	コンストラクタ。
	TedsActivationWarning.TedsActivationWarning (Device, Connector, SettingType)	コンストラクタ。
	TedsActivationWarning.TedsActivationWarning (Device, Sensor, SettingType)	コンストラクタ。
	TedsActivationWarning.TedsActivationWarning (Device, SettingType)	コンストラクタ。
	TedsActivationWarning.TedsActivationWarning (Device, Signal, SettingType)	コンストラクタ。

TedsActivationWarning.TedsActivationWarning Constructor (Device, Channel, SettingType)

コンストラクタ。

C++

```
TedsActivationWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

C#

```
public TedsActivationWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。

TedsActivationWarning.TedsActivationWarning Constructor (Device, Connector, SettingType)

コンストラクタ。

C++

```
TedsActivationWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

C#

```
public TedsActivationWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。

TedsActivationWarning.TedsActivationWarning Constructor (Device, Sensor, SettingType)

コンストラクタ。

C++

```
TedsActivationWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

C#

```
public TedsActivationWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。

TedsActivationWarning.TedsActivationWarning Constructor (Device, SettingType)

コンストラクタ。

C++

```
TedsActivationWarning(
    Device device,
    SettingType settingType
);
```

C#

```
public TedsActivationWarning(
    Device device,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。

TedsActivationWarning.TedsActivationWarning Constructor (Device, Signal, SettingType)

コンストラクタ。

C++

```
TedsActivationWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

C#

```
public TedsActivationWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。

TedsActivationWarning Properties

ProblemListExtensionクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	警告を記述するメッセージ。

TedsActivationWarning.Message Property

警告を記述するメッセージ。

C++

```
__property string Message;
```

C#

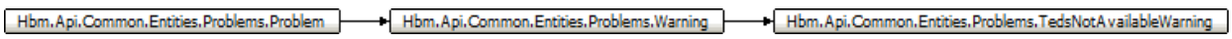
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

TedsNotAvailableWarning Class

TEDSがコネクタに接続されていないという警告を記述するクラス。



C++

```
class TedsNotAvailableWarning : public Warning;
```

C#

```
public class TedsNotAvailableWarning : Warning;
```

Visual Basic

```
Public Class TedsNotAvailableWarning
Inherits Warning
```

ファイル

TedsNotAvailableWarning.cs

TedsNotAvailableWarning Members

次の表は、TedsNotAvailableWarningによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	TedsNotAvailableWarning	これは、TedsNotAvailableWarningコンストラクタの多重定義の概要です。
	Warning	これは、Warningコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
R	Message	警告を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

TedsNotAvailableWarning Constructor

これは、TedsNotAvailableWarningコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	TedsNotAvailableWarning.TedsNotAvailableWarning (Device, Channel, SettingType)	コンストラクタ。
	TedsNotAvailableWarning.TedsNotAvailableWarning (Device, Connector, SettingType)	コンストラクタ。
	TedsNotAvailableWarning.TedsNotAvailableWarning (Device, Sensor, SettingType)	コンストラクタ。
	TedsNotAvailableWarning.TedsNotAvailableWarning (Device, SettingType)	コンストラクタ。
	TedsNotAvailableWarning.TedsNotAvailableWarning (Device, Signal, SettingType)	コンストラクタ。

TedsNotAvailableWarning.TedsNotAvailableWarning Constructor (Device, Channel, SettingType)

コンストラクタ。

C++

```
TedsNotAvailableWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

C#

```
public TedsNotAvailableWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。

TedsNotAvailableWarning.TedsNotAvailableWarning Constructor (Device, Connector, SettingType)

コンストラクタ。

C++

```
TedsNotAvailableWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

C#

```
public TedsNotAvailableWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。

TedsNotAvailableWarning.TedsNotAvailableWarning Constructor (Device, Sensor, SettingType)

コンストラクタ。

C++

```
TedsNotAvailableWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

C#

```
public TedsNotAvailableWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。

TedsNotAvailableWarning.TedsNotAvailableWarning Constructor (Device, SettingType)

コンストラクタ。

C++

```
TedsNotAvailableWarning(
    Device device,
    SettingType settingType
);
```

C#

```
public TedsNotAvailableWarning(
    Device device,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。

TedsNotAvailableWarning.TedsNotAvailableWarning Constructor (Device, Signal, SettingType)

コンストラクタ。

C++

```
TedsNotAvailableWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

C#

```
public TedsNotAvailableWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。

TedsNotAvailableWarning Properties

TedsNotAvailableWarningクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	警告を記述するメッセージ。

TedsNotAvailableWarning.Message Property

警告を記述するメッセージ。

C++

```
__property string Message;
```

C#

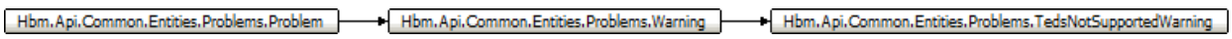
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

TedsNotSupportedWarning Class

コネクタがTEDSをサポートしていないという警告を記述するクラス。



C++

```
class TedsNotSupportedWarning : public Warning;
```

C#

```
public class TedsNotSupportedWarning : Warning;
```

Visual Basic

```
Public Class TedsNotSupportedWarning
Inherits Warning
```

ファイル

TedsNotSupportedWarning.cs

TedsNotSupportedWarning Members

次の表は、TedsNotSupportedWarningによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	TedsNotSupportedWarning	これは、TedsNotSupportedWarningコンストラクタの多重定義の概要です。
	Warning	これは、Warningコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
R	Message	警告を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

TedsNotSupportedWarning Constructor

これは、TedsNotSupportedWarningコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	TedsNotSupportedWarning.TedsNotSupportedWarning (Device, Channel, SettingType)	コンストラクタ。
	TedsNotSupportedWarning.TedsNotSupportedWarning (Device, Connector, SettingType)	コンストラクタ。
	TedsNotSupportedWarning.TedsNotSupportedWarning (Device, Sensor, SettingType)	コンストラクタ。
	TedsNotSupportedWarning.TedsNotSupportedWarning (Device, SettingType)	コンストラクタ。
	TedsNotSupportedWarning.TedsNotSupportedWarning (Device, Signal, SettingType)	コンストラクタ。

TedsNotSupportedWarning.TedsNotSupportedWarning Constructor (Device, Channel, SettingType)

コンストラクタ。

C++

```
TedsNotSupportedWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

C#

```
public TedsNotSupportedWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。

TedsNotSupportedWarning.TedsNotSupportedWarning Constructor (Device, Connector, SettingType)

コンストラクタ。

C++

```
TedsNotSupportedWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

C#

```
public TedsNotSupportedWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。

TedsNotSupportedWarning.TedsNotSupportedWarning Constructor (Device, Sensor, SettingType)

コンストラクタ。

C++

```
TedsNotAvailableWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

C#

```
public TedsNotAvailableWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。

TedsNotSupportedWarning.TedsNotSupportedWarning Constructor (Device, SettingType)

コンストラクタ。

C++

```
TedsNotAvailableWarning(
    Device device,
    SettingType settingType
);
```

C#

```
public TedsNotAvailableWarning(
    Device device,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。

TedsNotSupportedWarning.TedsNotSupportedWarning Constructor (Device, Signal, SettingType)

コンストラクタ。

C++

```
TedsNotAvailableWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

C#

```
public TedsNotAvailableWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。

TedsNotSupportedWarning Properties

TedsNotSupportedWarningクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	警告を記述するメッセージ。

TedsNotSupportedWarning.Message Property

警告を記述するメッセージ。

C++

```
__property string Message;
```

C#

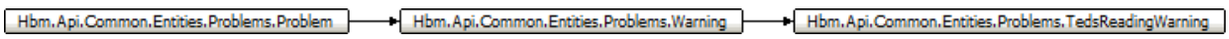
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

TedsReadingWarning Class

TEDSの読み取り中に発生した警告を記述するクラス。



C++

```
class TedsReadingWarning : public Warning;
```

C#

```
public class TedsReadingWarning : Warning;
```

Visual Basic

```
Public Class TedsReadingWarning
Inherits Warning
```

ファイル

TedsReadingWarning.cs

TedsReadingWarning Members

次の表は、TedsReadingWarningによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	TedsReadingWarning	これは、TedsReadingWarningコンストラクタの多重定義の概要です。
	Warning	これは、Warningコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
R	Message	警告を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

TedsReadingWarning Constructor

これは、TedsNotSupportedWarningコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	TedsReadingWarning.TedsReadingWarning (Device, Channel, SettingType)	コンストラクタ。
	TedsReadingWarning.TedsReadingWarning (Device, Connector, SettingType)	コンストラクタ。
	TedsReadingWarning.TedsReadingWarning (Device, Sensor, SettingType)	コンストラクタ。
	TedsReadingWarning.TedsReadingWarning (Device, SettingType)	コンストラクタ。
	TedsReadingWarning.TedsReadingWarning (Device, Signal, SettingType)	コンストラクタ。

TedsReadingWarning.TedsReadingWarning Constructor (Device, Channel, SettingType)

コンストラクタ。

C++

```
TedsReadingWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

C#

```
public TedsReadingWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。

TedsReadingWarning.TedsReadingWarning Constructor (Device, Connector, SettingType)

コンストラクタ。

C++

```
TedsReadingWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

C#

```
public TedsReadingWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。

TedsReadingWarning.TedsReadingWarning Constructor (Device, Sensor, SettingType)

コンストラクタ。

C++

```
TedsReadingWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

C#

```
public TedsReadingWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。

TedsReadingWarning.TedsReadingWarning Constructor (Device, SettingType)

コンストラクタ。

C++

```
TedsReadingWarning(
    Device device,
    SettingType settingType
);
```

C#

```
public TedsReadingWarning(
    Device device,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。

TedsReadingWarning.TedsReadingWarning Constructor (Device, Signal, SettingType)

コンストラクタ。

C++

```
TedsReadingWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

C#

```
public TedsReadingWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。

TedsReadingWarning Properties

TedsReadingWarningクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	警告を記述するメッセージ。

TedsReadingWarning.Message Property
警告を記述するメッセージ。

C++

```
__property string Message;
```

C#

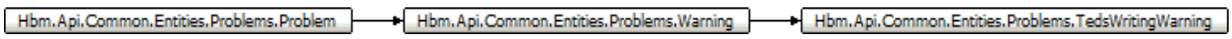
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

TedsWritingWarning Class

TEDSの書き込み中に発生した警告を記述するクラス。



C++

```
class TedsWritingWarning : public Warning;
```

C#

```
public class TedsWritingWarning : Warning;
```

Visual Basic

```
Public Class TedsWritingWarning
Inherits Warning
```

ファイル

TedsWritingWarning.cs

TedsWritingWarning Members

次の表は、TedsWritingWarningによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	TedsWritingWarning	これは、TedsReadingWarningコンストラクタの多重定義の概要です。
	Warning	これは、Warningコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
R	Message	警告を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

TedsWritingWarning Constructor

これは、TedsWritingWarningコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	TedsWritingWarning.TedsWritingWarning (Device, Channel, SettingType)	コンストラクタ。
	TedsWritingWarning.TedsWritingWarning (Device, Connector, SettingType)	コンストラクタ。
	TedsWritingWarning.TedsWritingWarning (Device, Sensor, SettingType)	コンストラクタ。
	TedsWritingWarning.TedsWritingWarning (Device, SettingType)	コンストラクタ。
	TedsWritingWarning.TedsWritingWarning (Device, Signal, SettingType)	コンストラクタ。

TedsWritingWarning.TedsWritingWarning Constructor (Device, Channel, SettingType)

コンストラクタ。

C++

```
TedsWritingWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

C#

```
public TedsWritingWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。

TedsWritingWarning.TedsWritingWarning Constructor (Device, Connector, SettingType)

コンストラクタ。

C++

```
TedsWritingWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

C#

```
public TedsWritingWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。

TedsWritingWarning.TedsWritingWarning Constructor (Device, Sensor, SettingType)

コンストラクタ。

C++

```
TedsWritingWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

C#

```
public TedsWritingWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。

TedsWritingWarning.TedsWritingWarning Constructor (Device, SettingType)

コンストラクタ。

C++

```
TedsWritingWarning(
    Device device,
    SettingType settingType
);
```

C#

```
public TedsWritingWarning(
    Device device,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。

TedsWritingWarning.TedsWritingWarning Constructor (Device, Signal, SettingType)

コンストラクタ。

C++

```
TedsWritingWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

C#

```
public TedsWritingWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。

TedsWritingWarning Properties

TedsWritingWarningクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	警告を記述するメッセージ。

TedsWritingWarning.Message Property

警告を記述するメッセージ。

C++

```
__property string Message;
```

C#

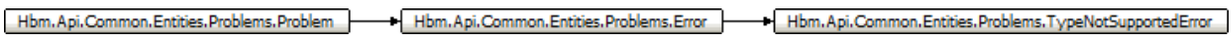
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

TypeNotSupportedError Class

デバイスの使用中に発生した型がサポートされていないエラーを記述するクラス。



C++

```
class TypeNotSupportedError : public Error;
```

C#

```
public class TypeNotSupportedError : Error;
```

Visual Basic

```
Public Class TypeNotSupportedError
Inherits Error
```

ファイル

TypeNotSupportedError.cs

TypeNotSupportedError Members

次の表は、TedsWritingWarningによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Error	これは、Errorコンストラクタの多重定義の概要です。
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	TypeNotSupportedError	これは、TypeNotSupportedErrorコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	警告を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

TypeNotSupportedError Constructor

これは、TypeNotSupportedErrorコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	TypeNotSupportedError.TypeNotSupportedError (Device, Channel, SettingType, string)	コンストラクタ。
	TypeNotSupportedError.TypeNotSupportedError (Device, Connector, SettingType, string)	コンストラクタ。
	TypeNotSupportedError.TypeNotSupportedError (Device, Sensor, SettingType, string)	コンストラクタ。
	TypeNotSupportedError.TypeNotSupportedError (Device, SettingType, string)	コンストラクタ。
	TypeNotSupportedError.TypeNotSupportedError (Device, Signal, SettingType, string)	コンストラクタ。

TypeNotSupportedError.TypeNotSupportedError Constructor (Device, Channel, SettingType, string)

コンストラクタ。

C++

```
TypeNotSupportedError(
    Device device,
    Channel channel,
    SettingType settingType
    string demandedValue
);
```

C#

```
public TypeNotSupportedError(
    Device device,
    Channel channel,
    SettingType settingType
    string demandedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType
    demandedValue As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。
demandedValue	エラーの原因となった要求された型。

TypeNotSupportedError.TypeNotSupportedError Constructor (Device, Connector, SettingType, string)

コンストラクタ。

C++

```
TypeNotSupportedError(
    Device device,
    Connector connector,
    SettingType settingType
    string demandedValue
);
```

C#

```
public TypeNotSupportedError(
    Device device,
    Connector connector,
    SettingType settingType
    string demandedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType
    demandedValue As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。
demandedValue	エラーの原因となった要求された型。

TypeNotSupportedError.TypeNotSupportedError Constructor (Device, Sensor, SettingType, string)

コンストラクタ。

C++

```
TypeNotSupportedError(
    Device device,
    Sensor sensor,
    SettingType settingType
    string demandedValue
);
```

C#

```
public TypeNotSupportedError(
    Device device,
    Sensor sensor,
    SettingType settingType
    string demandedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType
    demandedValue As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。
demandedValue	エラーの原因となった要求された型。

TypeNotSupportedError.TypeNotSupportedError Constructor (Device, SettingType, string)

コンストラクタ。

C++

```
TypeNotSupportedError(
    Device device,
    SettingType settingType
    string demandedValue
);
```

C#

```
public TypeNotSupportedError(
    Device device,
    SettingType settingType
    string demandedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType
    demandedValue As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。
demandedValue	エラーの原因となった要求された型。

TypeNotSupportedError.TypeNotSupportedError Constructor (Device, Signal, SettingType, string)

コンストラクタ。

C++

```
TypeNotSupportedError(
    Device device,
    Signal signal,
    SettingType settingType
    string demandedValue
);
```

C#

```
public TypeNotSupportedError(
    Device device,
    Signal signal,
    SettingType settingType
    string demandedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType
    demandedValue As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。
demandedValue	エラーの原因となった要求された型。

TypeNotSupportedError Properties

TypeNotSupportedErrorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	警告を記述するメッセージ。

TypeNotSupportedError.Message Property

警告を記述するメッセージ。

C++

```
__property string Message;
```

C#

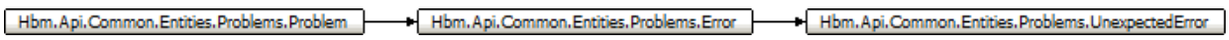
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

UnexpectedError Class

デバイスの使用中に発生した予期しないエラーを記述するクラス。



C++

```
class UnexpectedError : public Error;
```

C#

```
public class UnexpectedError : Error;
```

Visual Basic

```
Public Class UnexpectedError
Inherits Error
```

ファイル

UnexpectedError.cs

UnexpectedError Members

次の表は、UnexpectedErrorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Error	これは、Errorコンストラクタの多重定義の概要です。
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	UnexpectedError	これは、UnexpectedErrorコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	エラーを記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

UnexpectedError Constructor
これは、UnexpectedErrorコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	UnexpectedError.UnexpectedError (Device, Channel, Exception, SettingType, string)	コンストラクタ。
	UnexpectedError.UnexpectedError (Device, Connector, Exception, SettingType, string)	コンストラクタ。
	UnexpectedError.UnexpectedError (Device, Exception, SettingType, string)	コンストラクタ。
	UnexpectedError.UnexpectedError (Device, Sensor, Exception, SettingType, string)	コンストラクタ。
	UnexpectedError.UnexpectedError (Device, Signal, Exception, SettingType, string)	コンストラクタ。

UnexpectedError.UnexpectedError Constructor (Device, Channel, Exception, SettingType, string)

コンストラクタ。

C++

```
UnexpectedError(
    Device device,
    Channel channel,
    Exception exception,
    SettingType settingType
    string propertyName = null
);
```

C#

```
public UnexpectedError(
    Device device,
    Channel channel,
    Exception exception,
    SettingType settingType
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    exception As Exception,
    settingType As SettingType
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
exception	エラーの原因となった例外。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

UnexpectedError.UnexpectedError Constructor (Device, Connector, Exception, SettingType, string)

コンストラクタ。

C++

```
UnexpectedError(
    Device device,
    Connector connector,
    Exception exception,
    SettingType settingType
    string propertyName = null
);
```

C#

```
public UnexpectedError(
    Device device,
    Connector connector,
    Exception exception,
    SettingType settingType
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    exception As Exception,
    settingType As SettingType
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
exception	エラーの原因となった例外。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

UnexpectedError.UnexpectedError Constructor (Device, Exception, SettingType, string)

コンストラクタ。

C++

```
UnexpectedError(
    Device device,
    Exception exception,
    SettingType settingType
    string propertyName = null
);
```

C#

```
public UnexpectedError(
    Device device,
    Exception exception,
    SettingType settingType
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    exception As Exception,
    settingType As SettingType
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
exception	エラーの原因となった例外。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

UnexpectedError.UnexpectedError Constructor (Device, Sensor, Exception, SettingType, string)

コンストラクタ。

C++

```
UnexpectedError(
    Device device,
    Sensor sensor,
    Exception exception,
    SettingType settingType
    string propertyName = null
);
```

C#

```
public UnexpectedError(
    Device device,
    Sensor sensor,
    Exception exception,
    SettingType settingType
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    exception As Exception,
    settingType As SettingType
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
exception	エラーの原因となった例外。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

UnexpectedError.UnexpectedError Constructor (Device, Signal, Exception, SettingType, string)

コンストラクタ。

C++

```
UnexpectedError(
    Device device,
    Signal signal,
    Exception exception,
    SettingType settingType
    string propertyName = null
);
```

C#

```
public UnexpectedError(
    Device device,
    Signal signal,
    Exception exception,
    SettingType settingType
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    exception As Exception,
    settingType As SettingType
    propertyName As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
exception	エラーの原因となった例外。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。

UnexpectedError Properties

UnexpectedErrorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	エラーを記述するメッセージ。

UnexpectedError.Message Property

エラーを記述するメッセージ。

C++

```
__property string Message;
```

C#

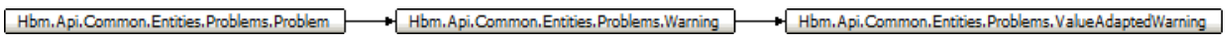
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

ValueAdaptedWarning Class

特定の値の適応を記述するクラス。



C++

```
class ValueAdaptedWarning : public Warning;
```

C#

```
public class ValueAdaptedWarning : Warning;
```

Visual Basic

```
Public Class ValueAdaptedWarning
Inherits Warning
```

ファイル

ValueAdaptedWarning.cs

ValueAdaptedWarning Members

次の表は、ValueAdaptedWarningによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	ValueAdaptedWarning	これは、ValueAdaptedWarningコンストラクタの多重定義の概要です。
	Warning	これは、Warningコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	警告を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	RealizedValue	Problem.DemandedValueの代わりに設定された値
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

ValueAdaptedWarning Constructor

これは、ValueAdaptedWarningコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	ValueAdaptedWarning.ValueAdaptedWarning (Device, Channel, SettingType, string, string, string)	コンストラクタ。
	ValueAdaptedWarning.ValueAdaptedWarning (Device, Connector, SettingType, string, string, string)	コンストラクタ。
	ValueAdaptedWarning.ValueAdaptedWarning (Device, Sensor, SettingType, string, string, string)	コンストラクタ。
	ValueAdaptedWarning.ValueAdaptedWarning (Device, SettingType, string, string, string)	コンストラクタ。
	ValueAdaptedWarning.ValueAdaptedWarning (Device, Signal, SettingType, string, string, string)	コンストラクタ。

ValueAdaptedWarning.ValueAdaptedWarning Constructor (Device, Channel, SettingType, string, string, string)

コンストラクタ。

C++

```
ValueAdaptedWarning(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string realizedValue
);
```

C#

```
public UnexpectedError(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string realizedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string,
    realizedValue As string
)
```

引数

引数	説明
device	警告の原因となったデバイス。
channel	警告の原因となったチャネル。
settingType	警告の原因となったオブジェクトの型。
propertyName	警告の原因となったプロパティの名前。
demandedValue	問題の原因となった要求された値。
realizedValue	Problem.DemandedValueの代わりに設定された値。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Hbm.Api.Common.Entities.Problems Namespace -> ValueAdaptedWarning Class -> ValueAdaptedWarning Members

ValueAdaptedWarning.ValueAdaptedWarning Constructor (Device, Connector, SettingType, string, string, string)

コンストラクタ。

C++

```
ValueAdaptedWarning(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string realizedValue
);
```

C#

```
public UnexpectedError(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string realizedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string,
    realizedValue As string
)
```

引数

引数	説明
device	警告の原因となったデバイス。
connector	警告の原因となったコネクタ。
settingType	警告の原因となったオブジェクトの型。
propertyName	警告の原因となったプロパティの名前。
demandedValue	問題の原因となった要求された値。
realizedValue	Problem.DemandedValueの代わりに設定された値。

ValueAdaptedWarning.ValueAdaptedWarning Constructor (Device, Sensor, SettingType, string, string, string)

コンストラクタ。

C++

```
ValueAdaptedWarning(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string realizedValue
);
```

C#

```
public UnexpectedError(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string realizedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string,
    realizedValue As string
)
```

引数

引数	説明
device	警告の原因となったデバイス。
sensor	警告の原因となったセンサ。
settingType	警告の原因となったオブジェクトの型。
propertyName	警告の原因となったプロパティの名前。
demandedValue	問題の原因となった要求された値。
realizedValue	Problem.DemandedValueの代わりに設定された値。

ValueAdaptedWarning.ValueAdaptedWarning Constructor (Device, SettingType, string, string, string)

コンストラクタ。

C++

```
ValueAdaptedWarning(
    Device device,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string realizedValue
);
```

C#

```
public UnexpectedError(
    Device device,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string realizedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string,
    realizedValue As string
)
```

引数

引数	説明
device	警告の原因となったデバイス。
settingType	警告の原因となったオブジェクトの型。
propertyName	警告の原因となったプロパティの名前。
demandedValue	問題の原因となった要求された値。
realizedValue	Problem.DemandedValueの代わりに設定された値。

ValueAdaptedWarning.ValueAdaptedWarning Constructor (Device, Signal, SettingType, string, string, string)

コンストラクタ。

C++

```
ValueAdaptedWarning(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string realizedValue
);
```

C#

```
public UnexpectedError(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string realizedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string,
    realizedValue As string
)
```

引数

引数	説明
device	警告の原因となったデバイス。
signal	警告の原因となった信号。
settingType	警告の原因となったオブジェクトの型。
propertyName	警告の原因となったプロパティの名前。
demandedValue	問題の原因となった要求された値。
realizedValue	Problem.DemandedValueの代わりに設定された値。

ValueAdaptedWarning Properties

ValueAdaptedWarningクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	警告を記述するメッセージ。
	RealizedValue	Problem.DemandedValueの代わりに設定された値

ValueAdaptedWarning.Message Property

警告を記述するメッセージ。

C++

```
__property string Message;
```

C#

```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

ValueAdaptedWarning.RealizedValue Property

Problem.DemandedValueの代わりに設定された値

C++

```
__property string RealizedValue;
```

C#

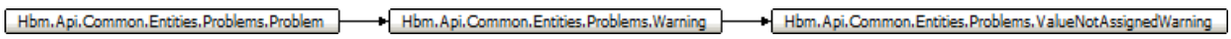
```
public string RealizedValue;
```

Visual Basic

```
Public Property RealizedValue() As string
```

ValueNotAssignedWarning Class

割り当てが完了していないことを示すために使用されるクラス。現時点では、これはNotSupportedセンサの割り当て中にのみ使用されます。この場合、代入関数は呼び出されていますが、代入は行われません。



C++

```
class ValueNotAssignedWarning : public Warning;
```

C#

```
public class ValueNotAssignedWarning : Warning;
```

Visual Basic

```
Public Class ValueNotAssignedWarning
Inherits Warning
```

ファイル

ValueNotAssignedWarning.cs

ValueNotAssignedWarning Members

次の表は、ValueNotAssignedWarningによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	ValueNotAssignedWarning	これは、ValueNotAssignedWarningコンストラクタの多重定義の概要です。
	Warning	これは、Warningコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	警告を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

ValueNotAssignedWarning Constructor

これは、ValueNotAssignedWarningコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	ValueNotAssignedWarning.ValueNotAssignedWarning (Device, Channel, SettingType)	コンストラクタ。
	ValueNotAssignedWarning.ValueNotAssignedWarning (Device, Connector, SettingType)	コンストラクタ。
	ValueNotAssignedWarning.ValueNotAssignedWarning (Device, Sensor, SettingType)	コンストラクタ。
	ValueNotAssignedWarning.ValueNotAssignedWarning (Device, SettingType)	コンストラクタ。
	ValueNotAssignedWarning.ValueNotAssignedWarning (Device, Signal, SettingType)	コンストラクタ。

ValueNotAssignedWarning.ValueNotAssignedWarning Constructor (Device, Channel, SettingType)

コンストラクタ。

C++

```
ValueNotAssignedWarning(
    Device device,
    Channel channel,
    SettingType settingType

);
```

C#

```
public ValueNotAssignedWarning(
    Device device,
    Channel channel,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType
)
```

引数

引数	説明
device	警告の原因となったデバイス。
channel	警告の原因となったチャネル。
settingType	警告の原因となったオブジェクトの型。

ValueNotAssignedWarning.ValueNotAssignedWarning Constructor (Device, Connector, SettingType)

コンストラクタ。

C++

```
ValueNotAssignedWarning(
    Device device,
    Connector connector,
    SettingType settingType

);
```

C#

```
public ValueNotAssignedWarning(
    Device device,
    Connector connector,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType
)
```

引数

引数	説明
device	警告の原因となったデバイス。
connector	警告の原因となったコネクタ。
settingType	警告の原因となったオブジェクトの型。

ValueNotAssignedWarning.ValueNotAssignedWarning Constructor (Device, Sensor, SettingType)

コンストラクタ。

C++

```
ValueNotAssignedWarning(
    Device device,
    Sensor sensor,
    SettingType settingType

);
```

C#

```
public ValueNotAssignedWarning(
    Device device,
    Sensor sensor,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType
)
```

引数

引数	説明
device	警告の原因となったデバイス。
sensor	警告の原因となったセンサ。
settingType	警告の原因となったオブジェクトの型。

ValueNotAssignedWarning.ValueNotAssignedWarning Constructor (Device, SettingType)

コンストラクタ。

C++

```
ValueNotAssignedWarning(
    Device device,
    SettingType settingType

);
```

C#

```
public ValueNotAssignedWarning(
    Device device,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType
)
```

引数

引数	説明
device	警告の原因となったデバイス。
settingType	警告の原因となったオブジェクトの型。

ValueNotAssignedWarning.ValueNotAssignedWarning Constructor (Device, Signal, SettingType)

コンストラクタ。

C++

```
ValueNotAssignedWarning(
    Device device,
    Signal signal,
    SettingType settingType

);
```

C#

```
public ValueNotAssignedWarning(
    Device device,
    Signal signal,
    SettingType settingType
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType
)
```

引数

引数	説明
device	警告の原因となったデバイス。
signal	警告の原因となった信号。
settingType	警告の原因となったオブジェクトの型。

ValueNotAssignedWarning Properties

ValueNotAssignedWarningクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	警告を記述するメッセージ。

ValueNotAssignedWarning.Message Property

C++

```
__property string Message;
```

C#

```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

ValueNotSetError Class

デバイスの使用中に発生した、値が設定されていないエラーを記述するクラス。



C++

```
class ValueNotSetError : public Error;
```

C#

```
public class ValueNotSetError : Error;
```

Visual Basic

```
Public Class ValueNotSetError
Inherits Error
```

ファイル

ValueNotSetError.cs

ValueNotSetError Members

次の表は、ValueNotSetErrorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Error	これは、Errorコンストラクタの多重定義の概要です。
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	ValueNotSetError	これは、ValueNotSetErrorコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
R	Message	エラーを記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

ValueNotSetError Constructor

これは、ValueNotSetErrorコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	ValueNotSetError.ValueNotSetError (Device, Channel, SettingType, string, string)	コンストラクタ。
	ValueNotSetError.ValueNotSetError (Device, Connector, SettingType, string, string)	コンストラクタ。
	ValueNotSetError.ValueNotSetError (Device, Sensor, SettingType, string, string)	コンストラクタ。
	ValueNotSetError.ValueNotSetError (Device, SettingType, string, string)	コンストラクタ。
	ValueNotSetError.ValueNotSetError (Device, Signal, SettingType, string, string)	コンストラクタ。

ValueNotSetError.ValueNotSetError Constructor (Device, Channel, SettingType, string, string)

コンストラクタ。

C++

```
ValueNotSetError(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName,
    string demandedValue
);
```

C#

```
public ValueNotSetError(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName,
    string demandedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。
demandedValue	設定できなかった値。

ValueNotSetError.ValueNotSetError Constructor (Device, Connector, SettingType, string, string)

コンストラクタ。

C++

```
ValueNotSetError(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName,
    string demandedValue
);
```

C#

```
public ValueNotSetError(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName,
    string demandedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。
demandedValue	設定できなかった値。

ValueNotSetError.ValueNotSetError Constructor (Device, Sensor, SettingType, string, string)

コンストラクタ。

C++

```
ValueNotSetError(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName,
    string demandedValue
);
```

C#

```
public ValueNotSetError(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName,
    string demandedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。
demandedValue	設定できなかった値。

ValueNotSetError.ValueNotSetError Constructor (Device, SettingType, string, string)

コンストラクタ。

C++

```
ValueNotSetError(
    Device device,
    SettingType settingType,
    string propertyName,
    string demandedValue
);
```

C#

```
public ValueNotSetError(
    Device device,
    SettingType settingType,
    string propertyName,
    string demandedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。
demandedValue	設定できなかった値。

ValueNotSetError.ValueNotSetError Constructor (Device, Signal, SettingType, string, string)

コンストラクタ。

C++

```
ValueNotSetError(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName,
    string demandedValue
);
```

C#

```
public ValueNotSetError(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName,
    string demandedValue
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。
demandedValue	設定できなかった値。

ValueNotSetError Properties

ValueNotSetErrorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	エラーを記述するメッセージ。

ValueNotSetError.Message Property

エラーを記述するメッセージ。

C++

```
__property string Message;
```

C#

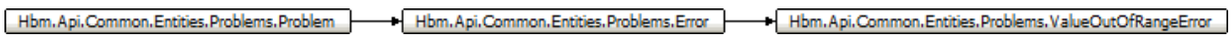
```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

ValueOutOfRangeError Class

デバイスの使用中に発生した範囲外エラーを記述するクラス。



C++

```
class ValueOutOfRangeError : public Error;
```

C#

```
public class ValueOutOfRangeError : Error;
```

Visual Basic

```
Public Class ValueOutOfRangeError
Inherits Error
```

ファイル

ValueOutOfRangeError.cs

ValueOutOfRangeError Members

次の表は、ValueOutOfRangeErrorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Error	これは、Errorコンストラクタの多重定義の概要です。
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	ValueOutOfRangeError	これは、ValueOutOfRangeErrorコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
	Exception	デバイスとの対話中に発生した予期しない例外。
	MaxPossibleValue	使用できる値の最大値。
R	Message	エラーを記述するメッセージ。
	MinPossibleValue	使用できる値の最小値。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

ValueOutOfRangeError Constructor

これは、ValueOutOfRangeErrorコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	ValueOutOfRangeError.ValueOutOfRangeError (Device, Channel, SettingType, string, string, string, string)	コンストラクタ。
	ValueOutOfRangeError.ValueOutOfRangeError (Device, Connector, SettingType, string, string, string, string)	コンストラクタ。
	ValueOutOfRangeError.ValueOutOfRangeError (Device, Sensor, SettingType, string, string, string, string)	コンストラクタ。
	ValueOutOfRangeError.ValueOutOfRangeError (Device, SettingType, string, string, string, string)	コンストラクタ。
	ValueOutOfRangeError.ValueOutOfRangeError (Device, Signal, SettingType, string, string, string, string)	コンストラクタ。

ValueOutOfRangeError.ValueOutOfRangeError Constructor (Device, Channel, SettingType, string, string, string, string)

コンストラクタ。

C++

```
ValueOutOfRangeError(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string minPossibleValue = null,
    string maxPossibleValue = null
);
```

C#

```
public ValueOutOfRangeError(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string minPossibleValue = null,
    string maxPossibleValue = null
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string,
    minPossibleValue As string = null,
    maxPossibleValue As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
channel	エラーの原因となったチャネル。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。
demandedValue	設定できなかった値。
minPossibleValue	使用できる値の最小値。
maxPossibleValue	使用できる値の最大値。

ValueOutOfRangeError.ValueOutOfRangeError Constructor (Device, Connector, SettingType, string, string, string, string)

コンストラクタ。

C++

```
ValueOutOfRangeError(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string minPossibleValue = null,
    string maxPossibleValue = null
);
```

C#

```
public ValueOutOfRangeError(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string minPossibleValue = null,
    string maxPossibleValue = null
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string,
    minPossibleValue As string = null,
    maxPossibleValue As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
connector	エラーの原因となったコネクタ。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。
demandedValue	設定できなかった値。
minPossibleValue	使用できる値の最小値。
maxPossibleValue	使用できる値の最大値。

ValueOutOfRangeError.ValueOutOfRangeError Constructor (Device, Sensor, SettingType, string, string, string, string)

コンストラクタ。

C++

```
ValueOutOfRangeError(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string minPossibleValue = null,
    string maxPossibleValue = null
);
```

C#

```
public ValueOutOfRangeError(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string minPossibleValue = null,
    string maxPossibleValue = null
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string,
    minPossibleValue As string = null,
    maxPossibleValue As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
sensor	エラーの原因となったセンサ。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。
demandedValue	設定できなかった値。
minPossibleValue	使用できる値の最小値。
maxPossibleValue	使用できる値の最大値。

ValueOutOfRangeError.ValueOutOfRangeError Constructor (Device, SettingType, string, string, string, string)

コンストラクタ。

C++

```
ValueOutOfRangeError(
    Device device,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string minPossibleValue = null,
    string maxPossibleValue = null
);
```

C#

```
public ValueOutOfRangeError(
    Device device,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string minPossibleValue = null,
    string maxPossibleValue = null
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string,
    minPossibleValue As string = null,
    maxPossibleValue As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。
demandedValue	設定できなかった値。
minPossibleValue	使用できる値の最小値。
maxPossibleValue	使用できる値の最大値。

ValueOutOfRangeError.ValueOutOfRangeError Constructor (Device, Signal, SettingType, string, string, string, string)

コンストラクタ。

C++

```
ValueOutOfRangeError(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string minPossibleValue = null,
    string maxPossibleValue = null
);
```

C#

```
public ValueOutOfRangeError(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName,
    string demandedValue,
    string minPossibleValue = null,
    string maxPossibleValue = null
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType,
    propertyName As string,
    demandedValue As string,
    minPossibleValue As string = null,
    maxPossibleValue As string = null
)
```

引数

引数	説明
device	エラーの原因となったデバイス。
signal	エラーの原因となった信号。
settingType	エラーの原因となったオブジェクトの型。
propertyName	エラーの原因となったプロパティの名前。
demandedValue	設定できなかった値。
minPossibleValue	使用できる値の最小値。
maxPossibleValue	使用できる値の最大値。

ValueOutOfRangeError Properties

ValueOutOfRangeErrorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	MaxPossibleValue	使用できる値の最大値。
R	Message	エラーを記述するメッセージ。
	MinPossibleValue	使用できる値の最小値。

ValueOutOfRangeError.MaxPossibleValue Property

使用できる値の最大値。

C++

```
__property string MaxPossibleValue;
```

C#

```
public string MaxPossibleValue;
```

Visual Basic

```
Public Property MaxPossibleValue() As string
```

ValueOutOfRangeError.Message Property

エラーを記述するメッセージ。

C++

```
__property string Message;
```

C#

```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

ValueOutOfRangeError.MinPossibleValue Property

使用できる値の最小値。

C++

```
__property string MinPossibleValue;
```

C#

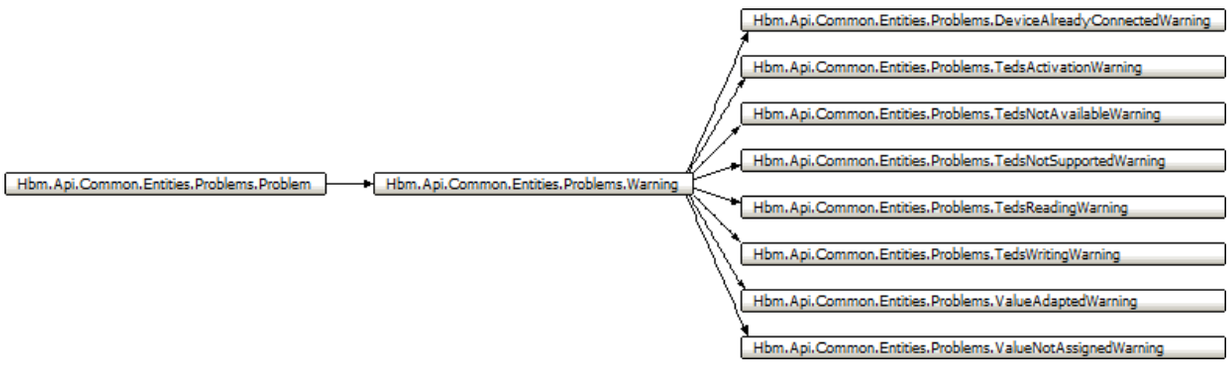
```
public string MinPossibleValue;
```

Visual Basic

```
Public Property MinPossibleValue() As string
```

Warning Class

デバイスの使用中に発生した警告を記述するクラス。



C++

```
class DeviceAlreadyConnectedWarning : public Warning;
```

C#

```
public class DeviceAlreadyConnectedWarning : Warning;
```

Visual Basic

```
Public Class DeviceAlreadyConnectedWarning
Inherits Warning
```

ファイル

DeviceAlreadyConnectedWarning.cs

Warning Members

次の表は、ValueNotSetErrorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Problem	これは、Problemコンストラクタの多重定義の概要です。
	Warning	これは、Warningコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ChannelUniqueID	問題の原因となったチャネルの一意の識別子。
	ConnectorUniqueID	問題の原因となったコネクタの一意の識別子。
	DemandedValue	問題の原因となった値。
	DeviceUniqueID	問題の原因となったデバイスの一意の識別子。
R	Message	警告を記述するメッセージ。
	OriginalContext	デバイスによって生成された問題の前後関係。
	OriginalErrorCode	デバイスが生成した元のエラー。
	OriginalMessage	デバイスが生成した元の問題(エラー)メッセージ。
R	Position	問題が発生した場所。
	PropertyName	問題の原因となったプロパティの名前。
	SensorUniqueID	問題の原因となったセンサの一意の識別子。
	SettingType	問題の原因となるオブジェクトの型。
	SignalUniqueID	問題の原因となった信号の一意の識別子。

Warning Constructor

これは、Warningコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	Warning.Warning (Device, Channel, SettingType, string)	コンストラクタ。
	Warning.Warning (Device, Connector, SettingType, string)	コンストラクタ。
	Warning.Warning (Device, Sensor, SettingType, string)	コンストラクタ。
	Warning.Warning (Device, SettingType, string)	コンストラクタ。
	Warning.Warning (Device, Signal, SettingType, string)	コンストラクタ。

Warning.Warning Constructor (Device, Channel, SettingType, string)

コンストラクタ。

C++

```
Warning(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public Warning(
    Device device,
    Channel channel,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    channel As Channel,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	警告の原因となったデバイス。
channel	警告の原因となったチャネル。
settingType	警告の原因となったオブジェクトの型。
propertyName	警告の原因となったプロパティの名前。

Warning.Warning Constructor (Device, Connector, SettingType, string)

コンストラクタ。

C++

```
Warning(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public Warning(
    Device device,
    Connector connector,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    connector As Connector,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	警告の原因となったデバイス。
connector	警告の原因となったコネクタ。
settingType	警告の原因となったオブジェクトの型。
propertyName	警告の原因となったプロパティの名前。

Warning.Warning Constructor (Device, Sensor, SettingType, string)

コンストラクタ。

C++

```
Warning(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public Warning(
    Device device,
    Sensor sensor,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    sensor As Sensor,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	警告の原因となったデバイス。
sensor	警告の原因となったセンサ。
settingType	警告の原因となったオブジェクトの型。
propertyName	警告の原因となったプロパティの名前。

Warning.Warning Constructor (Device, SettingType, string)

コンストラクタ。

C++

```
Warning(
    Device device,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public Warning(
    Device device,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	警告の原因となったデバイス。
settingType	警告の原因となったオブジェクトの型。
propertyName	警告の原因となったプロパティの名前。

Warning.Warning Constructor (Device, Signal, SettingType, string)

コンストラクタ。

C++

```
Warning(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName = null
);
```

C#

```
public Warning(
    Device device,
    Signal signal,
    SettingType settingType,
    string propertyName
);
```

Visual Basic

```
Public Sub New(
    device As Device,
    signal As Signal,
    settingType As SettingType,
    propertyName As string = null
)
```

引数

引数	説明
device	警告の原因となったデバイス。
signal	警告の原因となった信号。
settingType	警告の原因となったオブジェクトの型。
propertyName	警告の原因となったプロパティの名前。

Warning Properties

Warningクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Message	警告を記述するメッセージ。

Warning.Message Property

警告を記述するメッセージ。

C++

```
__property string Message;
```

C#

```
public override string Message;
```

Visual Basic

```
Public override Property Message() As string
```

Hbm.Api.Common.Entities.Scalings Namespace

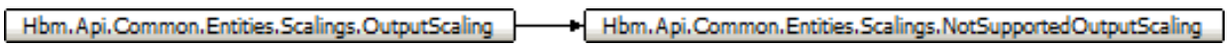
これは名前空間Hbm.Api.Common.Entities.Scalingsです。

クラス

	名前	説明
	NotSupportedOutputScaling	サポートされていない出力スケールを表します。
A	OutputScaling	アナログ出力信号のスケールを表します。
	OutputScalingPoint	単一の出力スケーリングポイントを表します。
	OutputTableScaling	アナログ出力信号のテーブルスケールを表します。

NotSupportedOutputScaling Class

サポートされていない出力スケールを表します。



C++

```
class NotSupportedOutputScaling : public OutputScaling;
```

C#

```
public class NotSupportedOutputScaling : OutputScaling;
```

Visual Basic

```
Public Class NotSupportedOutputScaling
Inherits OutputScaling
```

ファイル

NotSupportedOutputScaling.cs

NotSupportedOutputScaling Members

次の表は、NotSupportedOutputScalingによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	OutUnit	電気単位(例: "V")。
R	ScalingType	出力スケーリングの型(読み取り専用)。
	SourceUnit	電気単位(例: "V")。

NotSupportedOutputScaling Properties

NotSupportedOutputScalingクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ScalingType	出力スケーリングの型(読み取り専用)。

NotSupportedOutputScaling.ScalingType Property

出力スケーリングの型(読み取り専用)。

C++

```
__property OutputScalingType ScalingType;
```

C#

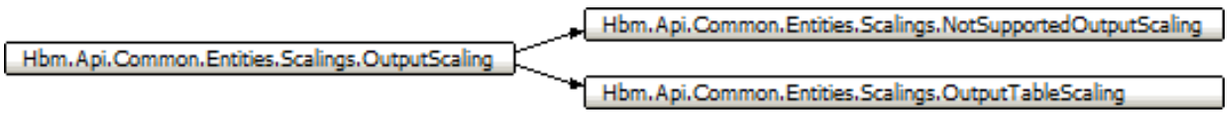
```
public override OutputScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As OutputScalingType
```

OutputScaling Class

アナログ出力信号のスケールを表します。



C++

```
class OutputScaling;
```

C#

```
public abstract class OutputScaling;
```

Visual Basic

```
Public abstract Class OutputScaling
```

ファイル

OutputScaling.cs

OutputScaling Members

次の表は、OutputScalingによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	OutUnit	電気単位(例: "V")。
R	ScalingType	スケーリングの型(読み取り専用)。
	SourceUnit	電気単位(例: "V")。

OutputScaling Properties

NotSupportedOutputScalingクラスのプロパティの一覧です。

Public Properties

	名前	説明
	OutUnit	電気単位(例: "V")。
R	ScalingType	スケーリングの型(読み取り専用)。
	SourceUnit	電気単位(例: "V")。

OutputScaling.OutUnit Property

電気単位(例: "V")。

C++

```
__property string OutUnit;
```

C#

```
public string OutUnit;
```

Visual Basic

```
Public Property OutUnit() As string
```

OutputScaling.ScalingType Property

スケーリングの型(読み取り専用)。

C++

```
__property OutputScalingType ScalingType;
```

C#

```
public abstract OutputScalingType ScalingType;
```

Visual Basic

```
Public abstract Property ScalingType() As OutputScalingType
```

備考

派生クラスで実装する必要があります。

OutputScaling.SourceUnit Property

電気単位(例: "V")。

C++

```
__property string SourceUnit;
```

C#

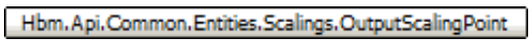
```
public string SourceUnit;
```

Visual Basic

```
Public Property SourceUnit() As string
```

OutputScalingPoint Class

単一の出力スケールポイントを表します。



C++

```
class OutputScalingPoint;
```

C#

```
public class OutputScalingPoint;
```

Visual Basic

```
Public Class OutputScalingPoint
```

ファイル

OutputScalingPoint.cs

OutputScalingPoint Members

次の表は、OutputScalingによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	OutputScalingPoint	これは、OutputScalingPointコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	Out	ポイントのY値。
	Source	ポイントのX値。

OutputScalingPoint Constructor

これは、OutputScalingPointコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	OutputScalingPoint.OutputScalingPoint ()	新しいインスタンスを作成します。
	OutputScalingPoint.OutputScalingPoint (decimal, decimal)	新しいインスタンスを作成し、値を初期化します。

OutputScalingPoint.OutputScalingPoint Constructor ()

新しいインスタンスを作成します。

C++

```
OutputScalingPoint();
```

C#

```
public OutputScalingPoint();
```

Visual Basic

```
Public Sub New()
```

OutputScalingPoint.OutputScalingPoint Constructor (decimal, decimal)

新しいインスタンスを作成し、値を初期化します。

C++

```
OutputScalingPoint(
    decimal x,
    decimal y
);
```

C#

```
public OutputScalingPoint(
    decimal x,
    decimal y
);
```

Visual Basic

```
Public Sub New(
    x As decimal,
    y As decimal
)
```

引数

引数	説明
x	x値。
y	y値。

OutputScalingPoint Properties

OutputScalingPointクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Out	ポイントのY値。
	Source	ポイントのX値。

OutputScalingPoint.Out Property

ポイントのY値。

C++

```
__property decimal Out;
```

C#

```
public decimal Out;
```

Visual Basic

```
Public Property Out() As decimal
```

OutputScalingPoint.Source Property

ポイントのX値。

C++

```
__property decimal Source;
```

C#

```
public decimal Source;
```

Visual Basic

```
Public Property Source() As decimal
```

OutputTableScaling Class

アナログ出力信号のテーブルスケールを表します。



C++

```
class OutputTableScaling : public OutputScaling;
```

C#

```
public class OutputTableScaling : OutputScaling;
```

Visual Basic

```
Public Class OutputTableScaling
Inherits OutputScaling
```

ファイル

OutputTableScaling.cs

OutputTableScaling Members

次の表は、OutputTableScalingによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	OutputTableScaling	これは、OutputTableScalingコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	OutUnit	電気単位(例: "V")。
	ScalingPoints	スケーリングテーブルを表すスケーリングポイントのリスト。
R	ScalingType	スケーリングの型(読み取り専用)。
	SourceUnit	電気単位(例: "V")。

OutputTableScaling Constructor

これは、OutputTableScalingコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	OutputTableScaling.OutputTableScaling ()	新しいインスタンスを作成し、スケーリングポイントのリストを初期化します。
	OutputTableScaling.OutputTableScaling (TableScaling)	新しいインスタンスを作成し、指定したセンサテーブルのスケーリングを新しいテーブルの出力スケーリングに変換します。

OutputTableScaling.OutputTableScaling Constructor ()

新しいインスタンスを作成し、スケーリングポイントのリストを初期化します。

C++

```
OutputTableScaling();
```

C#

```
public OutputTableScaling();
```

Visual Basic

```
Public Sub New()
```

OutputTableScaling.OutputTableScaling Constructor (TableScaling)

新しいインスタンスを作成し、指定したセンサテーブルのスケーリングを新しいテーブルの出力スケーリングに変換します。

C++

```
OutputTableScaling(
    TableScaling tableScaling
);
```

C#

```
public OutputTableScaling(
    TableScaling tableScaling
);
```

Visual Basic

```
Public Sub New(
tableScaling As TableScaling
)
```

OutputTableScaling Properties

OutputScalingPointクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ScalingPoints	スケーリングテーブルを表すスケーリングポイントのリスト。
R	ScalingType	スケーリングの型(読み取り専用)。

OutputTableScaling.ScalingPoints Property

スケーリングテーブルを表すスケーリングポイントのリスト。

C++

```
__property List<OutputScalingPoint> ScalingPoints;
```

C#

```
public List<OutputScalingPoint> ScalingPoints;
```

Visual Basic

```
Public ScalingPoints() As List<OutputScalingPoint>
```

OutputTableScaling.ScalingType Property

スケーリングの型(読み取り専用)。

Hbm.Api.Common.Enums.OutputScalingType Enumeration

C++

```
__property OutputScalingType ScalingType;
```

C#

```
public override OutputScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As OutputScalingType
```

Hbm.Api.Common.Entities.Signals Namespace

これは名前空間Hbm.Api.Common.Entities.Signalsです。

クラス

	名前	説明
	AnalogInSignal	物理センサから送られる測定信号を表します。
	AnalogOutSignal	アナログ出力から送られる信号を表します。
	CanInSignal	CANセンサから送られる測定信号を表します。
	CanRawSignal	CAN Rawレシーバによって送られる信号を表します。この信号の測定値はより複雑であり、値を1つのdouble値だけで表すことはできません(AnalogInSignalなどの他のすべての信号タイプと同様)。実際、CanRawSignalの測定値はHbm.Api.Common.Entities.Values.CanRawMessagesであり、次の2つの方法でアクセスできます。 **新しい方法(新たに導入されました - これまではCAN Rawデータのみ)**: IGenericSignal Interfaceを使用して値にアクセスし、CanRawSignalのIGenericSignal.GenericMeasurementValuesを取得できます。GenericMeasurementValuesの値であるCanRawMessagesにアクセスするには、続き...
	DigitalCompressedGroupSignal	すべてに同じ測定値が供給されたn個のデジタル圧縮信号を保持するデジタル信号を表します。これにより、デジタル圧縮信号のビットマスクは、それ自身の測定値を表す測定値のビットを定義します。
	DigitalCompressedSignal	デジタル圧縮信号は、n個のデジタル信号に対して1つの測定値を提供するDigitalCompressedGroupSignalの一部です。例: 16個のデジタル入力ピンに1つの16ビット値を送信するデバイスは、16個のデジタル圧縮入力信号を保持し、それぞれが測定値の1ビット(マスクされたビットに従って)にマッピングされます。
	DigitalSignal	単一のデジタル入力または出力を表します。
	FbgSignal	ファイバブラッググレーティングセンサによって送られる測定信号を表します。
	GenericSignal	ReSharperはInheritdocConsiderUsageを一度無効にします。
	Signal	測定信号を表します。
	SyncSignal	物理センサ(アナログまたはファイバブラッググレーティングセンサ)によって送られる測定信号を表します。
	VirtualSignal	物理センサによって送られたものではない測定信号を表します(演算信号など)。

インタフェース

	名前	説明
	IFilterable	すべてのフィルタリング可能な信号用のインタフェース。一部の信号には関連付けられたフィルタがないため、これらの信号はフィルタリングできません。
	IGenericSignal	一般的な信号用のインタフェース。つまり、一般的な測定結果タイプ(測定値として CanRawMessagesを配信するCanRawSignalなど)を持つ信号を意味します。

AnalogInSignal Class

物理センサから送られる測定信号を表します。



C++

```
class AnalogInSignal : public SyncSignal;
```

C#

```
public abstract class AnalogInSignal : SyncSignal
```

Visual Basic

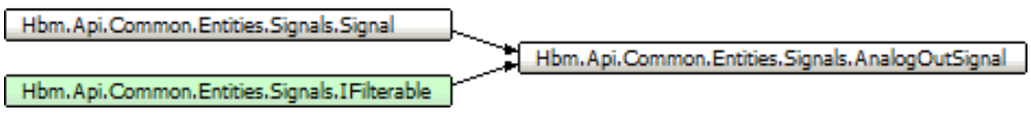
```
Public  abstract Class AnalogInSignal
Inherits SyncSignal
```

ファイル

AnalogInSignal.cs

AnalogOutSignal Class

アナログ出力によって送られる信号を表します。



C++

```
class AnalogOutSignal : public Signal, public IFilterable;
```

C#

```
public abstract class AnalogOutSignal : Signal, IFilterable;
```

Visual Basic

```
Public  abstract Class AnalogOutSignal
Inherits Signal
Inherits IFilterable
```

ファイル

AnalogOutSignal.cs

AnalogOutSignal Members

次の表は、AnalogOutSignalによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	CreateCircularBuffer	内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。!
	FreeCircularBuffer	内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Methods

	名前	説明
	AnalogOutSignal	クラスの新しいインスタンスを作成し、変数を初期化します。
	GetSingleMeasurementValue	単一の測定値を返します。この値を更新するには、Hbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用する必要があります。
A	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。
	SetMeasurementValues	これは、SetMeasurementValuesメソッドの多重定義の概要です。
	SetSingleMeasurementValue	単一の測定値を設定します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Properties

	名前	説明
	ContinuousMeasurementValues	ストリーミングなどの実行中の測定によって取得された複数の測定値を返します。
	ExcludeFromDaqTimeout	連続測定の実行中にこの信号が測定値を送信する必要があるまでのタイムアウト(msec)。この間に新しい測定値がない場合、現在実行中の測定中に信号は無視されます。
R	HasSampleRate	信号がサンプルレートを使用する場合はtrueを返します。
	IdleTime	連続測定の実行中に測定値を最後に受信してからの時間(msec)。IdleTime >= ExcludeFromDaqTimeoutで、それでも測定値が受信されない場合、IsExcludedFromDaqはtrueに設定され、現在実行中の測定中に信号は無視されます。
	IsExcludedFromDaq	この信号がタイムアウトになり、実行中の連続測定内で使用できなくなった場合、true。
	IsMeasurable	信号が測定の一部である可能性があるかどうかを示します(Hbm.Api.Common.DaqMeasurementまたはHbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用するなど)。
	Name	信号の名前。
	SampleRate	連続測定中に測定値を収集するために使用するサンプルレート。
	Scaling	出力信号のスケーリングに使用するスケール。
	SelectedFilter	出力信号をフィルタリングに使用するフィルタ。
	SynchronousMode	信号が測定値を送信する方法を定義します。これは、等距離方式(1秒内の測定値の数 = サンプルレート)または非等距離方式(測定値は予測できない時間に配信される)のいずれかです。

AnalogOutSignal.AnalogOutSignal Constructor

クラスの新しいインスタンスを作成し、変数を初期化します。

C++

```
AnalogOutSignal();
```

C#

```
public AnalogOutSignal();
```

Visual Basic

```
Public Sub New()
```

AnalogOutSignal Properties

AnalogOutSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Scaling	出力信号のスケーリングに使用するスケール。
	SelectedFilter	出力信号をフィルタリングに使用するフィルタ。

AnalogOutSignal.Scaling Property

出力信号のスケーリングに使用するスケール。

C++

```
__property OutputScaling Scaling;
```

C#

```
public OutputScaling Scaling;
```

Visual Basic

```
Public Property Scaling() As OutputScaling
```

AnalogOutSignal.SelectedFilter Property

出力信号をフィルタリングに使用するフィルタ。

C++

```
__property Filter SelectedFilter;
```

C#

```
public Filter SelectedFilter;
```

Visual Basic

```
Public Property SelectedFilter() As Filter
```

CanInSignal Class

CANセンサから送られる測定信号を表します。



C++

```
class CanInSignal : public Signal;
```

C#

```
public abstract class CanInSignal : Signal;
```

Visual Basic

```
Public  abstract Class CanInSignal
Inherits Signal
```

ファイル

CanInSignal.cs

CanRawSignal Class

CAN Rawレシーバによって送られる信号を表します。この信号の測定値はより複雑であり、値を1つのdouble値だけで表すことはできません(AnalogInSignalなどの他のすべての信号タイプと同様)。実際、CanRawSignalの測定値はHbm.Api.Common.Entities.Values.CanRawMessagesであり、次の2つの方法でアクセスできます。

新しい方法(新たに導入されました - これまではCAN Rawデータのみ):
IGenericSignal Interfaceを使用して値にアクセスし、CanRawSignalのIGenericSignal.GenericMeasurementValuesを取得できます。GenericMeasurementValuesの値であるCanRawMessagesにアクセスするには、さらにMeasurementValuesBase<CanRawMessage>へのキャストを行う必要があります。次のコード スニペットは、CanRawSignalのCanRawMessagesへのアクセスを示します:

```
if (signal is IGenericSignal genericSignal)
{
   if (genericSignal.GenericMeasurementValues is MeasurementValuesBase <CanRawMessage>
canRawMeasurementValues)
   {
      // now you have access to
      //  canRawMeasurementValues.UpdatedValueCount
      //  canRawMeasurementValues.Timestamps[0..canRawMeasurementValues.
UpdatedValueCount-1]       // relative (to measurement start) timestamps (starting around
0.0)
      //  canRawMeasurementValues.Values[0..canRawMeasurementValues.UpdatedValueCount-1].
Timestamp // absolute! timestamp here (the one of the message itself)
      //  canRawMeasurementValues.Values[0..canRawMeasurementValues.UpdatedValueCount-1].
Header    // header of the CanRawMessage
      //  canRawMeasurementValues.Values[0..canRawMeasurementValues.UpdatedValueCount-1].
Payload[0...63] // 64 Bytes Payload because of CAN FD data
   }
}
```

2番目の方法(他のすべての信号タイプにすでに使用されている"通常の"方法):
他のすべての信号に対して既に行っているように、Signal.ContinuousMeasurementValuesにアクセスすることで値にアクセスできます。そうするとCanRawSignalの場合、常に10の倍数のSignal.ContinuousMeasurementValues.Timestamps[n...n+9]が存在し、これらはまったく同じ時間情報を持ち、"これにより"対応する Signal.ContinuousMeasurementValues.Values[n...n+9]を一緒に属するものとしてマークし、1つのCanRawMessageを表します。Signal.ContinuousMeasurementValues.Valuesを調べると、長さが10個の値[n...n+9]である特定のパターンにも気づくでしょう。1つのCanRawMessageを表すこれら10個のdouble値(Values[n...n+9])の意味は次のとおりです:

1. Values[n] = CAN Rawメッセージの絶対タイムスタンプ
2. Values[n+1] = CAN Rawメッセージのヘッダ(double!としてコード化されているため、実際の値を取得するには、doubleの8バイトをULongの8バイトに変換する必要があります)
3. Values[n+2] = 1.8バイトのペイロード(double!としてコード化されているため、doubleの8バイトを8バイトに変換する必要があります)
4. Values[n+3] = 2.8バイトのペイロード(double!としてコード化されているため、doubleの 8 バイトを8バイトに変換する必要があります)

5 ....

6. Values[n+9] = 8. 8バイトのペイロード(double!としてコード化されているため、doubleの8バイトを8バイトに変換する必要があります)

したがって、Values[n+2] ... Values[n+9]には、実際のデータを取得するために8つのdoubleから64バイトの配列に変換する必要があるCANメッセージのペイロードが保持されます。測定値を取得する方法を変更したくない場合(たとえば、測定値をすでに保護しており、それらを保護する方法を変更したくない場合)、CANメッセージを取得するこの方法を使用する必要があります。したがって、CAN Raw測定値を保護するためにコードを変更する必要はありません。ただし、測定データを理解するには(ファイルから再度ロードした後など)、データを理解するためにそれらをCanRawMessagesに変換する必要があります。

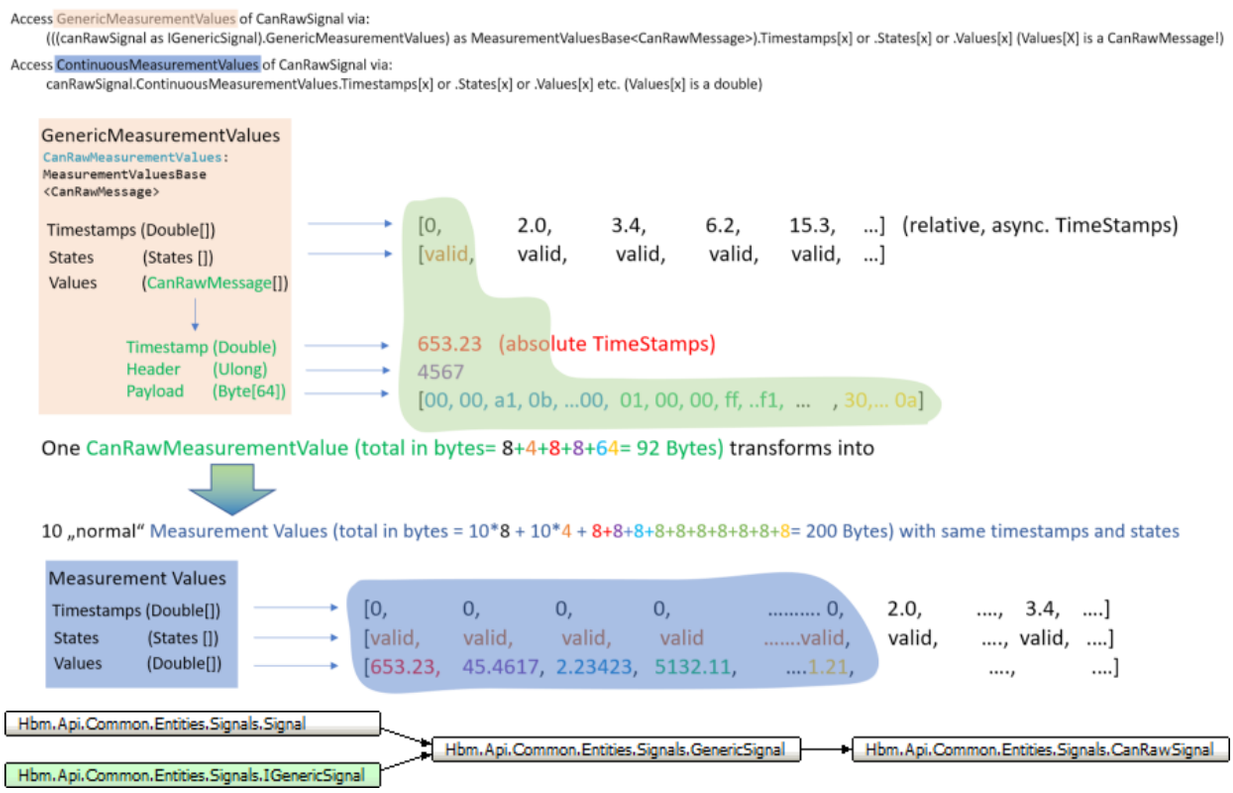
10個のValuesを1個のCanRawMessageに変換するには、Hbm.Api.Common.Entities.Values.CanRawMessageのコンストラクタを使用するだけです。

```
// suppose you have the Values (here just containing ONE CanRawMessage) available
// e.g. after loading from disk or getting directly during a measurement..
int startIndexOfCanMessage = 0; //10,20,30... always a multiple of 10 because 10 "normal"
measurement values represent one CanRawMessage
CanRawMessage canRawMessage = new CanRawMessage(signal.ContinuousMeasurementValues.
Values[startIndexOfCanMessage],
                                                signal.ContinuousMeasurementValues.
Values[startIndexOfCanMessage + 1],
                                                new double[]
                                                {
                                                    signal.ContinuousMeasurementValues.
Values[startIndexOfCanMessage + 2],
                                                    signal.ContinuousMeasurementValues.
Values[startIndexOfCanMessage + 3],
                                                    signal.ContinuousMeasurementValues.
Values[startIndexOfCanMessage + 4],
                                                    signal.ContinuousMeasurementValues.
Values[startIndexOfCanMessage + 5],
                                                    signal.ContinuousMeasurementValues.
Values[startIndexOfCanMessage + 6],
                                                    signal.ContinuousMeasurementValues.
Values[startIndexOfCanMessage + 7],
                                                    signal.ContinuousMeasurementValues.
Values[startIndexOfCanMessage + 8],
                                                    signal.ContinuousMeasurementValues.
Values[startIndexOfCanMessage + 9]
                                                });
// now you can access the CanRawMessage properties
// canRawMessage.Timestamp;          // (double)    absolute! timestamp here (the one of
the message itself)
// canRawMessage.Header;             // (ulong)     header of the CanRawMessage
// canRawMessage.Payload[0...63];    // (byte[64])  64 Bytes Payload because of CAN FD data
```

CanRawMeasurementValuesの型はCanRawMessageです。1つのCanRawMessage(CanRawMeasurementValues.Value)は、上で説明したように10個の"通常の"ContinuousMeasurementValueにマッピングされます。

これが、1つのCanRawMeasurementValuesに対して常に10個の"通常の"ContinuousMeasurementValueを取得する理由です。APIは、CanRawMessagesにアクセスするための両方の方法を提供することに重点を置いています。

次の図は、CanRawSignalのGenericMeasurementValuesとContinuousMeasurementValuesの関係を示しています:



C++

```
class CanRawSignal : public GenericSignal<CanRawMessage>;
```

C#

```
public abstract class CanRawSignal : GenericSignal<CanRawMessage>;
```

Visual Basic

```
Public  abstract Class CanRawSignal
Inherits GenericSignal
```

ファイル

CanRawSignal.cs

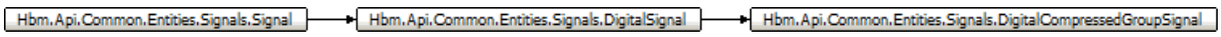
備考

データ取得の準備方法に関係なくCanRawSignalは常に、CanRawMessage型の最大100000 ContinuousMeasurementValuesと10000 GenericMeasurementValuesを割り当てます。メモリの問題が発生した場合(BufferOverFlowOccurredがtrueに設定されている場合)、DaqMeasurement.FillMeasurementValuesをより頻繁に呼び出して、より頻繁にデータをリクエストしてください。

重要: これはおそらく少し理解しにくいため、CommonAPI Demo Project内にCAN Rawの処理を示すセクションも提供しています。

DigitalCompressedGroupSignal Class

すべてに同じ測定値が供給されたn個のデジタル圧縮信号を保持するデジタル信号を表します。これにより、デジタル圧縮信号のビットマスクは、それ自身の測定値を表す測定値のビットを定義します。



C++

```
class DigitalCompressedGroupSignal : public DigitalSignal;
```

C#

```
public class DigitalCompressedGroupSignal : DigitalSignal;
```

Visual Basic

```
Public  Class DigitalCompressedGroupSignal
Inherits DigitalSignal
```

ファイル

DigitalCompressedGroupSignal.cs

DigitalCompressedGroupSignal Members

次の表は、DigitalCompressedGroupSignalによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	CreateCircularBuffer	内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。!
	FreeCircularBuffer	内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Methods

	名前	説明
	DigitalCompressedGroupSignal	新しいインスタンスを作成し、この圧縮グループのデジタル信号のリストを初期化します。
	GetSingleMeasurementValue	単一の測定値を返します。この値を更新するには、Hbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用する必要があります。
	GetUniqueID	デジタル圧縮信号のリストの最初の信号を識別する一意の文字列を返します。
	SetMeasurementValues	これは、SetMeasurementValuesメソッドの多重定義の概要です。
	SetSingleMeasurementValue	単一の測定値を設定します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Properties

	名前	説明
	ContinuousMeasurementValues	ストリーミングなどの実行中の測定によって取得された複数の測定値を返します。
	DigitalCompressedSignals	すべてこの信号の測定値を使用して独自の測定値を計算するデジタル圧縮信号のリスト。通常、この信号の測定値の1つのビット位置は、デジタル圧縮信号の1つの測定値にマッピングされます。
	ExcludeFromDaqTimeout	連続測定の実行中にこの信号が測定値を送信する必要があるまでのタイムアウト(msec)。この間に新しい測定値がない場合、現在実行中の測定中に信号は無視されます。
	Group	DigitalCompressedSignalsで収集されたすべての信号が属するグループ。
R	HasSampleRate	信号がサンプルレートを使用する場合はtrueを返します。
	IdleTime	連続測定の実行中に測定値を最後に受信してからの時間(msec)。IdleTime >= ExcludeFromDaqTimeoutで、それでも測定値が受信されない場合、IsExcludedFromDaqはtrueに設定され、現在実行中の測定中に信号は無視されます。
	IsExcludedFromDaq	この信号がタイムアウトになり、実行中の連続測定内で使用できなくなった場合、true。
	IsMeasurable	信号が測定の一部である可能性があるかどうかを示します(Hbm.Api.Common.DaqMeasurementまたはHbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用するなど)。
	Name	信号の名前。
	SampleRate	連続測定中に測定値を収集するために使用するサンプルレート。
	SelectedFilter	出力信号をフィルタリングに使用するフィルタ。
	SynchronousMode	信号が測定値を送信する方法を定義します。これは、等距離方式(1秒内の測定値の数 = サンプルレート)または非等距離方式(測定値は予測できない時間に配信される)のいずれかです。

DigitalCompressedGroupSignal.DigitalCompressedGroupSignal Constructor

新しいインスタンスを作成し、この圧縮グループのデジタル信号のリストを初期化します。

C++

```
DigitalCompressedGroupSignal(
    int groupNo,
    List<DigitalCompressedSignal> compressedDigitalSignals
);
```

C#

```
public DigitalCompressedGroupSignal(
    int groupNo,
    List<DigitalCompressedSignal> compressedDigitalSignals
);
```

Visual Basic

```
Public Sub New(
    groupNo As Integer,
    compressedDigitalSignals As List<DigitalCompressedSignal>
)
```

引数

引数	説明
groupNo	圧縮グループのグループ番号。
compressedDigitalSignals	1 つの圧縮グループに対するデジタル信号のリスト。

DigitalCompressedGroupSignal Methods

DigitalCompressedGroupSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デジタル圧縮信号のリストの最初の信号を識別する一意の文字列を返します。

DigitalCompressedGroupSignal.GetUniqueID Method

デジタル圧縮信号のリストの最初の信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェアモジュール上の信号を識別するためにストリーミングで使用される情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。

DigitalCompressedGroupSignal Properties

DigitalCompressedGroupSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DigitalCompressedSignals	すべてこの信号の測定値を使用して独自の測定値を計算するデジタル圧縮信号のリスト。通常、この信号の測定値の1つのビット位置は、デジタル圧縮信号の1つの測定値にマッピングされます。
	Group	DigitalCompressedSignalsで収集されたすべての信号が属するグループ。

DigitalCompressedGroupSignal.DigitalCompressedSignals Property

すべてこの信号の測定値を使用して独自の測定値を計算するデジタル圧縮信号のリスト。通常、この信号の測定値の1つのビット位置は、デジタル圧縮信号の1つの測定値にマッピングされます。

C++

```
__property List<DigitalCompressedSignal> DigitalCompressedSignals;
```

C#

```
public List<DigitalCompressedSignal> DigitalCompressedSignals;
```

Visual Basic

```
Public Property DigitalCompressedSignals() As List<DigitalCompressedSignal>
```

DigitalCompressedGroupSignal.Group Property

DigitalCompressedSignalsで収集されたすべての信号が属するグループ。

C++

```
__property int Group;
```

C#

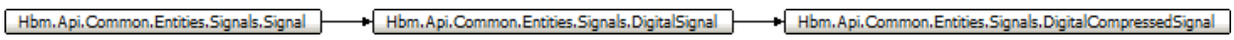
```
public int Group;
```

Visual Basic

```
Public Property Group() As Integer
```

DigitalCompressedSignal Class

デジタル圧縮信号は、n個のデジタル信号に対して1つの測定値を提供するDigitalCompressedGroupSignalの一部です。
例: 16個のデジタル入力ピンに1つの16ビット値を送信するデバイスは、16個のデジタル圧縮入力信号を保持し、それぞれが測定値の1ビット(マスクされたビットに従って)にマッピングされます。



C++

```
class DigitalCompressedSignal : public DigitalSignal;
```

C#

```
public abstract class DigitalCompressedSignal : DigitalSignal;
```

Visual Basic

```
Public abstract Class DigitalCompressedSignal
Inherits DigitalSignal
```

ファイル

DigitalCompressedSignal.cs

DigitalCompressedSignal Members

次の表は、DigitalCompressedSignalによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	CreateCircularBuffer	内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。!
	FreeCircularBuffer	内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Methods

	名前	説明
	DecompressMeasurementValues	マスクされたビット値とデジタル圧縮グループ信号の測定値に従ってこのデジタル圧縮信号の測定値を計算することにより、digitalCompressedGROUPSignalのContinuousMeasurementValuesに従ってこの信号のContinuousMeasurementValuesを更新します。
	GetSingleMeasurementValue	単一の測定値を返します。この値を更新するには、Hbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用する必要があります。
	GetUniqueID	デジタル圧縮信号のリストの最初の信号を識別する一意の文字列を返します。
	SetMeasurementValues	これは、SetMeasurementValuesメソッドの多重定義の概要です。
	SetSingleMeasurementValue	単一の測定値を設定します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Properties

	名前	説明
	ContinuousMeasurementValues	ストリーミングなどの実行中の測定によって取得された複数の測定値を返します。
	ExcludeFromDaqTimeout	連続測定の実行中にこの信号が測定値を送信する必要があるまでのタイムアウト(msec)。この間に新しい測定値がない場合、現在実行中の測定中に信号は無視されます。
	Group	この信号が属するグループ。グループは、実際には1つの物理信号にすぎないデジタル信号の論理的複合体を定義します。たとえば、1つの信号にマッピングされたn個のピンを持つデジタル入力ポートです。この物理信号の各ビットは、同じグループの別のデジタル信号に属します。
R	HasSampleRate	信号がサンプルレートを使用する場合はtrueを返します。
	IdleTime	連続測定の実行中に測定値を最後に受信してからの時間(msec)。IdleTime >= ExcludeFromDaqTimeoutで、それでも測定値が受信されない場合、IsExcludedFromDaqはtrueに設定され、現在実行中の測定中に信号は無視されます。
	IsExcludedFromDaq	この信号がタイムアウトになり、実行中の連続測定内で使用できなくなった場合、true。
	IsMeasurable	信号が測定の一部である可能性があるかどうかを示します(Hbm.Api.Common.DaqMeasurementまたはHbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用するなど)。
V	MaskedBit	このデジタル信号にマッピングされる測定値のビットを定義します。可能な値は0から31です。
V	MaskedBitValue	マスクされたビットに対応する値を返します。(例: MaskedBitis 3の場合、対応する値は2^(3)=8です)。
	Name	信号の名前。
	SampleRate	連続測定中に測定値を収集するために使用するサンプルレート。
	SynchronousMode	信号が測定値を送信する方法を定義します。これは、等距離方式(1秒内の測定値の数 = サンプルレート)または非等距離方式(測定値は予測できない時間に配信される)のいずれかです。

DigitalCompressedSignal Methods

DigitalCompressedSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	DecompressMeasurementValues	マスクされたビット値とデジタル圧縮グループ信号の測定値に従ってこのデジタル圧縮信号の測定値を計算することにより、digitalCompressedGROUPSignalのContinuousMeasurementValuesに従ってこの信号のContinuousMeasurementValuesを更新します。

DigitalCompressedSignal.DecompressMeasurementValues Method

マスクされたビット値とデジタル圧縮グループ信号の測定値に従ってこのデジタル圧縮信号の測定値を計算することにより、digitalCompressedGROUPSignalのContinuousMeasurementValuesに従ってこの信号のContinuousMeasurementValuesを更新します。

C++

```
void DecompressMeasurementValues(
    DigitalCompressedGroupSignal digitalCompressedGroupSignal
);
```

C#

```
public virtual void DecompressMeasurementValues(
    DigitalCompressedGroupSignal digitalCompressedGroupSignal
);
```

Visual Basic

```
Public virtual Function DecompressMeasurementValues(
    digitalCompressedGroupSignal As DigitalCompressedGroupSignal
) As void
```

戻引数

引数	説明
digitalCompressedGroupSignal	この信号の測定値(マスクされたビット値に従って)にマッピングする測定値を保持する信号。

DigitalCompressedSignal Properties

DigitalCompressedSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Group	この信号が属するグループ。グループは、実際には1つの物理信号にすぎないデジタル信号の論理的複合体を定義します。たとえば、1つの信号にマッピングされたn個のピンを持つデジタル入力ポートです。この物理信号の各ビットは、同じグループの別のデジタル信号に属します。
	MaskedBit	このデジタル信号にマッピングされる測定値のビットを定義します。可能な値は0から31です。
	MaskedBitValue	マスクされたビットに対応する値を返します。(例: MaskedBitis 3の場合、対応する値は2^(3)=8です)。

DigitalCompressedSignal.Group Property

この信号が属するグループ。グループは、実際には1つの物理信号にすぎないデジタル信号の論理的複合体を定義します。たとえば、1つの信号にマッピングされたn個のピンを持つデジタル入力ポートです。この物理信号の各ビットは、同じグループの別のデジタル信号に属します。

C++

```
__property int Group;
```

C#

```
public int Group;
```

Visual Basic

```
Public Property Group() As Integer
```

DigitalCompressedSignal.MaskedBit Property

このデジタル信号にマッピングされる測定値のビットを定義します。可能な値は0から31です。

C++

```
__property int MaskedBit;
```

C#

```
public virtual int MaskedBit;
```

Visual Basic

```
Public virtual Property MaskedBit() As Integer
```

DigitalCompressedSignal.MaskedBitValue Property

マスクされたビットに対応する値を返します。(例: MaskedBitis 3の場合、対応する値は2^(3)=8です)。

C++

```
__property int MaskedBitValue;
```

C#

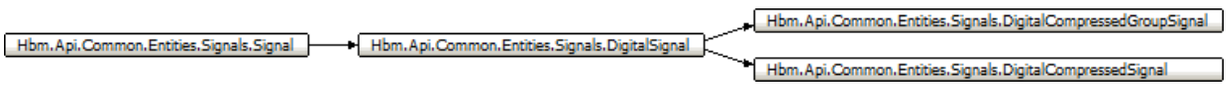
```
public virtual int MaskedBitValue;
```

Visual Basic

```
Public virtual Property MaskedBitValue() As Integer
```

DigitalSignal Class

単一のデジタル入力または出力を表します。



C++

```
class DigitalSignal : public Signal;
```

C#

```
public abstract class DigitalSignal : Signal;
```

Visual Basic

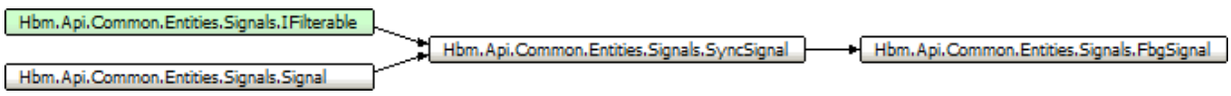
```
Public abstract Class DigitalSignal
Inherits Signal
```

ファイル

DigitalSignal.cs

FbgSignal Class

ファイバブラッググレーティングセンサによって送られる測定信号を表します。



C++

```
class FbgSignal : public SyncSignal;
```

C#

```
public abstract class FbgSignal : SyncSignal;
```

Visual Basic

```
Public abstract Class FbgSignal
Inherits SyncSignal
```

ファイル

FbgSignal.cs

GenericSignal Class

ReSharperはInheritdocConsiderUsageを一度無効にします。



C++

```
template <T>
class GenericSignal : public Signal, public IGenericSignal;
```

C#

```
public abstract class GenericSignal<T> : Signal, IGenericSignal;
```

Visual Basic

```
Public abstract Class GenericSignal(Of T)
Inherits Signal
Inherits IGenericSignal
```

ファイル

GenericSignal.cs

備考

ReSharperはInheritdocConsiderUsageを一度無効にします。

GenericSignal Members

次の表は、GenericSignalによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	AddValuesToCircularBuffer	numberOfValuesToAddの"measurement value lines"を循環バッファに追加します。
	CreateCircularBuffer	内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。!
	FreeCircularBuffer	内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Internal Properties

	名前	説明
R	GenericCircularBuffer	循環バッファ。測定値を一時的に保存するために使用します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、CircularBufferに直接アクセスしないでください!

Protected Methods

	名前	説明
	GenericSignal	

Public Methods

	名前	説明
	CopyAllValuesFromCircularBuffer ToMeasurementValues	
	CreateGenericCircularBuffer	
	FreeGenericCircularBuffer	
	GetSingleMeasurementValue	単一の測定値を返します。この値を更新するには、Hbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用する必要があります。
A	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。
	InitGenericMeasurementValues	
A	SetGenericMeasurementValues	
	SetMeasurementValues	これは、SetMeasurementValuesメソッドの多重定義の概要です。
	SetSingleMeasurementValue	単一の測定値を設定します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Properties

	名前	説明
	ContinuousMeasurementValues	ストリーミングなどの実行中の測定によって取得された複数の測定値を返します。
	ExcludeFromDaqTimeout	連続測定の実行中にこの信号が測定値を送信する必要があるまでのタイムアウト(msec)。この間に新しい測定値がない場合、現在実行中の測定中に信号は無視されます。
	GenericMeasurementValues	ストリーミングなどの実行中の測定を通じて取得した複数の測定値を返します。値が常にdouble型であるSignal.ContinuousMeasurementValuesとは異なり、GenericMeasurementValuesには値として任意のデータ型(object)を含めることができます。現時点では、これはCanRawSignalの場合の CanRawMessagesにのみ使用されます。
R	HasSampleRate	信号がサンプルレートを使用する場合はtrueを返します。
	IdleTime	連続測定の実行中に測定値を最後に受信してからの時間(msec)。IdleTime >= ExcludeFromDaqTimeoutで、それでも測定値が受信されない場合、IsExcludedFromDaqはtrueに設定され、現在実行中の測定中に信号は無視されます。
	IsExcludedFromDaq	この信号がタイムアウトになり、実行中の連続測定内で使用できなくなった場合、true。
	IsMeasurable	信号が測定の一部である可能性があるかどうかを示します(Hbm.Api.Common.DaqMeasurementまたはHbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用するなど)。
	Name	信号の名前。
	SampleRate	連続測定中に測定値を収集するために使用するサンプルレート。
	SynchronousMode	信号が測定値を送信する方法を定義します。これは、等距離方式(1秒内の測定値の数 = サンプルレート)または非等距離方式(測定値は予測できない時間に配信される)のいずれかです。

GenericSignal.GenericSignal Constructor

C++

```
GenericSignal();
```

C#

```
protected GenericSignal();
```

Visual Basic

```
protected Sub New()
```

GenericSignal Methods

GenericSignalクラスのメソッドの一覧です。

Internal Methods

	名前	説明
	AddValuesToCircularBuffer	numberOfValuesToAddの"measurement value lines"を循環バッファに追加します。

Public Methods

	名前	説明
	CopyAllValuesFromCircularBuffer ToMeasurementValues	
	CreateGenericCircularBuffer	
	FreeGenericCircularBuffer	
	InitGenericMeasurementValues	
	SetGenericMeasurementValues	

GenericSignal.AddValuesToCircularBuffer Method

numberOfValuesToAddの"measurement value lines"を循環バッファに追加します。

C++

```
void AddValuesToCircularBuffer(
    T[] measurementValues,
    double[] timeStampValues,
    MeasurementValueState[] statusValues,
    int numberOfValuesToAdd
);
```

C#

```
internal void AddValuesToCircularBuffer(
    T[] measurementValues,
    double[] timeStampValues,
    MeasurementValueState[] statusValues,
    int numberOfValuesToAdd
);
```

Visual Basic

```
Friend Function AddValuesToCircularBuffer(
    measurementValues As T[],
    timeStampValues As double[],
    statusValues As MeasurementValueState[],
    numberOfValuesToAdd As Integer
) As void
```

引数

引数	説明
measurementValues	測定値。
timeStampValues	タイムスタンプ。
statusValues	測定値のステータス。
numberOfValuesToAdd	追加する値の数。

例外

例外	説明
System.ArgumentException	numberOfValuesToAddが0未満の場合にスローされます。measurementValues、timeStampValues、またはstatusValuesのサイズがnumberOfValuesToAddより小さい場合にスローされます。
System.ArgumentNullException	MeasurementValues、timeStampValues、またはstatusValuesがnullの場合にスローされます。

GenericSignal.CopyAllValuesFromCircularBufferToMeasurementValues Method

C++

```
void CopyAllValuesFromCircularBufferToMeasurementValues();
```

C#

```
public void CopyAllValuesFromCircularBufferToMeasurementValues();
```

Visual Basic

```
Public Function CopyAllValuesFromCircularBufferToMeasurementValues() As void
```

GenericSignal.CreateGenericCircularBuffer Method

C++

```
void CreateGenericCircularBuffer(
    int bufferTimeout,
    int minNumberOfBufferedValues
);
```

C#

```
public void CreateGenericCircularBuffer(
    int bufferTimeout,
    int minNumberOfBufferedValues
);
```

Visual Basic

```
Public Function CreateGenericCircularBuffer(
    bufferTimeout As Integer,
    minNumberOfBufferedValues As Integer
) As void
```

GenericSignal.FreeGenericCircularBuffer Method

C++

```
void FreeGenericCircularBuffer();
```

C#

```
public void FreeGenericCircularBuffer();
```

Visual Basic

```
Public Function FreeGenericCircularBuffer() As void
```

GenericSignal.InitGenericMeasurementValues Method

C++

```
void InitGenericMeasurementValues();
```

C#

```
public void InitGenericMeasurementValues();
```

Visual Basic

```
Public Function InitGenericMeasurementValues() As void
```

GenericSignal.SetGenericMeasurementValues Method

C++

```
void SetGenericMeasurementValues(
    double[] timeStamps,
    double[] measurementValues,
    MeasurementValueState[] states,
    int numberOfValuesToAdd
);
```

C#

```
public abstract void SetGenericMeasurementValues(
    double[] timeStamps,
    double[] measurementValues,
    MeasurementValueState[] states,
    int numberOfValuesToAdd
);
```

Visual Basic

```
Public abstract Function SetGenericMeasurementValues(
    timeStamps As double[],
    measurementValues As double[],
    states As MeasurementValueState[],
    numberOfValuesToAdd As Integer
) As void
```

GenericSignal Properties

DigitalCompressedSignalクラスのプロパティの一覧です。

Internal Properties

	名前	説明
	GenericCircularBuffer	循環バッファ。測定値を一時的に保存するために使用します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、CircularBufferに直接アクセスしないでください!

GenericSignal.GenericCircularBuffer Property

循環バッファ。測定値を一時的に保存するために使用します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、CircularBufferに直接アクセスしないでください!

C++

```
__property MeasurementValuesCircularBufferBase<T> GenericCircularBuffer;
```

C#

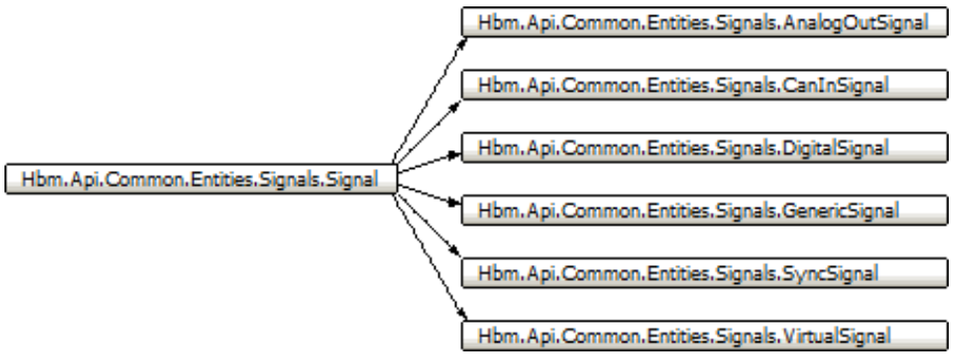
```
internal MeasurementValuesCircularBufferBase<T> GenericCircularBuffer;
```

Visual Basic

```
Friend Property GenericCircularBuffer() As MeasurementValuesCircularBufferBase<T>
```

Signal Class

測定信号を表します。



C++

```
template <T>
class Signal;
```

C#

```
public abstract class Signal;
```

Visual Basic

```
Public abstract Class Signal
```

ファイル

Signal.cs

Signal Members

次の表は、Signalによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	CreateCircularBuffer	内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。!
	FreeCircularBuffer	内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Protected Methods

	名前	説明
	Signal	クラスの新しいインスタンスを作成し、変数を初期化します。

Public Methods

	名前	説明
	GetSingleMeasurementValue	単一の測定値を返します。この値を更新するには、Hbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用する必要があります。
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。
	SetMeasurementValues	これは、SetMeasurementValuesメソッドの多重定義の概要です。
	SetSingleMeasurementValue	単一の測定値を設定します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Properties

	名前	説明
	ContinuousMeasurementValues	ストリーミングなどの実行中の測定によって取得された複数の測定値を返します。
	ExcludeFromDaqTimeout	連続測定の実行中にこの信号が測定値を送信する必要があるまでのタイムアウト(msec)。この間に新しい測定値がない場合、現在実行中の測定中に信号は無視されます。
R	HasSampleRate	信号がサンプルレートを使用する場合はtrueを返します。
	IdleTime	連続測定の実行中に測定値を最後に受信してからの時間(msec)。IdleTime >= ExcludeFromDaqTimeoutで、それでも測定値が受信されない場合、IsExcludedFromDaqはtrueに設定され、現在実行中の測定中に信号は無視されます。
	IsExcludedFromDaq	この信号がタイムアウトになり、実行中の連続測定内で使用できなくなった場合、true。
	IsMeasurable	信号が測定の一部である可能性があるかどうかを示します(Hbm.Api.Common.DaqMeasurementまたはHbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用するなど)。
	Name	信号の名前。
	SampleRate	連続測定中に測定値を収集するために使用するサンプルレート。
	SynchronousMode	信号が測定値を送信する方法を定義します。これは、等距離方式(1秒内の測定値の数 = サンプルレート)または非等距離方式(測定値は予測できない時間に配信される)のいずれかです。

Signal.Signal Constructor

クラスの新しいインスタンスを作成し、変数を初期化します。

C++

```
Signal();
```

C#

```
protected Signal();
```

Visual Basic

```
Protected Sub New()
```

Signal Methods

Signalクラスのメソッドの一覧です。

Internal Methods

	名前	説明
	CreateCircularBuffer	内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。!
	FreeCircularBuffer	内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Methods

	名前	説明
	GetSingleMeasurementValue	単一の測定値を返します。この値を更新するには、Hbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用する必要があります。
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。
	SetMeasurementValues	これは、SetMeasurementValuesメソッドの多重定義の概要です。
	SetSingleMeasurementValue	単一の測定値を設定します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Signal.CreateCircularBuffer Method

内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

C++

```
void CreateCircularBuffer(
    int bufferTimeout,
    int minNumberOfBufferedValues = MIN_CIRCULAR_BUFFER_SIZE
);
```

C#

```
internal void CreateCircularBuffer(
    int bufferTimeout,
    int minNumberOfBufferedValues
);
```

Visual Basic

```
Friend Function CreateCircularBuffer(
    bufferTimeout As Integer
    minNumberOfBufferedValues As Integer  = MIN_CIRCULAR_BUFFER_SIZE
) As void
```

引数

引数	説明
bufferTimeout	ミリ秒単位のタイムアウト(バッファは信号のサンプルレートと指定されたタイムアウトに従って割り当てられます)。
minNumberOfBufferedValues	循環バッファにバッファする値の最小数(デフォルトは1000)。

備考

バッファリングされる値の数を明示的に設定する場合は、バッファタイムアウトを0に設定し、バッファリングされる値の最小数を要求された値に設定します。

Signal.FreeCircularBuffer Method

内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

C++

```
void FreeCircularBuffer();
```

C#

```
internal void FreeCircularBuffer();
```

Visual Basic

```
Friend Function FreeCircularBuffer() As void
```

Signal.GetSingleMeasurementValue Method

単一の測定値を返します。この値を更新するには、Hbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用する必要があります。

C++

```
MeasurementValue GetSingleMeasurementValue();
```

C#

```
public MeasurementValue GetSingleMeasurementValue();
```

Visual Basic

```
Public Function GetSingleMeasurementValue() As MeasurementValue
```

戻り値

単一の測定値。

Signal.GetUniqueID Method

信号が属するデバイス内で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public abstract string GetUniqueID();
```

Visual Basic

```
Public abstract Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別、または識別子が設定されていない場合は`String.Empty`。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェアモジュール上の信号を識別するためにストリーミングで使用する情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

SetMeasurementValues Method

これは、SetMeasurementValuesメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	Signal.SetMeasurementValues (double[], double[], MeasurementValueState[])	指定された同じサイズを追加します。(timeStamps、measurementValues、およびstatesの)配列を信号の内部循環バッファに保存します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみです!
	Signal.SetMeasurementValues (double[], double[], MeasurementValueState[], int)	numberOfValuesToAddの3つ(それぞれ1つのtimeStamp、1つのmeasurementValue、および1つのstateで構成される)を信号の内部循環バッファに追加します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみです!
	Signal.SetMeasurementValues (List<double>, List<double>, List<MeasurementValueState>)	指定された同じサイズを追加します! リスト(timeStamps、measurementValues、およびstates)を信号の内部循環バッファに保存します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみです!

Signal.SetMeasurementValues Method (double[], double[], MeasurementValueState[])

指定された同じサイズを追加します。(timeStamps、measurementValues、およびstatesの)配列を信号の内部循環バッファに保存します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみです!

C++

```
public void SetMeasurementValues
    double[] timeStamps,
    double[] measurementValues,
    MeasurementValueState[] states
);
```

C#

```
public void SetMeasurementValues(
    double[] timeStamps,
    double[] measurementValues,
    MeasurementValueState[] states
);
```

Visual Basic

```
Public Function SetMeasurementValues(
    timeStamps As double[],
    measurementValues As double[]
    states As MeasurementValueState[]
) As void
```

引数

引数	説明
timeStamps	タイムスタンプ。
measurementValues	測定値。
states	測定値のステータス。

例外

例外	説明
System.ArgumentException	配列のサイズが同じでない場合にスローされます。
System.ArgumentNullException	timeStamps、measurementValues、またはstatesがnullの場合にスローされます。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Hbm.Api.Common.Entities.Signals Namespace -> Signal Class -> Signal Methods -> SetMeasurementValues Method

Signal.SetMeasurementValues Method (double[], double[], MeasurementValueState[], int)

numberOfValuesToAddの3つ(それぞれ1つのtimeStamp、1つのmeasurementValue、および1つのstateで構成される)を信号の内部循環バッファに追加します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみです!

C++

```
public void SetMeasurementValues
    double[] timeStamps,
    double[] measurementValues,
    MeasurementValueState[] states,
    int numberOfValuesToAdd
);
```

C#

```
public void SetMeasurementValues(
    double[] timeStamps,
    double[] measurementValues,
    MeasurementValueState[] states,
    int numberOfValuesToAdd
);
```

Visual Basic

```
Public Function SetMeasurementValues(
    timeStamps As double[],
    measurementValues As double[][],
    states As MeasurementValueState[][],
    numberOfValuesToAdd As Integer
) As void
```

引数

引数	説明
timeStamps	タイムスタンプ。
measurementValues	測定値。
states	測定値のステータス。
numberOfValuesToAdd	信号の内部循環バッファに追加する値の数。

例外

例外	説明
System.ArgumentException	numberOfValuesToAddが0未満の場合にスローされます。timeStamps、measurementValues、またはstatesのサイズがnumberOfValuesToAddより小さい場合にスローされます。
System.ArgumentNullException	timeStamps、measurementValues、またはstatesがnullの場合にスローされます。

Signal.SetMeasurementValues Method (List<double>, List<double>, List<MeasurementValueState>)

指定された同じサイズを追加します! リスト(timeStamps、measurementValues、およびstates)を信号の内部循環バッファに保存します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみです!

C++

```
public void SetMeasurementValues(
    List<double> timeStamps,
    List<double> measurementValues,
    List<MeasurementValueState> states
);
```

C#

```
public void SetMeasurementValues(
    List<double> timeStamps,
    List<double> measurementValues,
    List<MeasurementValueState> states
);
```

Visual Basic

```
Public Function SetMeasurementValues(
    timeStamps As List<double>,
    measurementValues As List<double>,
    states As List<MeasurementValueState>
) As void
```

引数

引数	説明
timeStamps	タイムスタンプ。
measurementValues	測定値。
states	測定値のステータス。

例外

例外	説明
System.ArgumentException	リストのサイズが同じでない場合にスローされます。
System.ArgumentNullException	timeStamps、measurementValues、またはstatesがnullの場合にスローされます。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Hbm.Api.Common.Entities.Signals Namespace -> Signal Class -> Signal Methods -> SetMeasurementValues Method

Signal.SetSingleMeasurementValue Method

単一の測定値を設定します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

C++

```
void SetSingleMeasurementValue
    double timeStamps,
    double value,
    MeasurementValueState states
);
```

C#

```
public void SetSingleMeasurementValue(
    double timeStamps,
    double value,
    MeasurementValueState states
);
```

Visual Basic

```
Public Function SetSingleMeasurementValue(
    timeStamps As double,
    value As double,
    state As MeasurementValueState
) As void
```

引数

引数	説明
timeStamps	測定値のタイムスタンプ。
value	測定値。
states	測定値のステータス。

Signal Properties

Signalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ContinuousMeasurementValues	ストリーミングなどの実行中の測定によって取得された複数の測定値を返します。
	ExcludeFromDaqTimeout	連続測定の実行中にこの信号が測定値を送信する必要があるまでのタイムアウト(msec)。この間に新しい測定値がない場合、現在実行中の測定中に信号は無視されます。
R	HasSampleRate	信号がサンプルレートを使用する場合はtrueを返します。
	IdleTime	連続測定の実行中に測定値を最後に受信してからの時間(msec)。IdleTime >= ExcludeFromDaqTimeoutで、それでも測定値が受信されない場合、IsExcludedFromDaqはtrueに設定され、現在実行中の測定中に信号は無視されます。
	IsExcludedFromDaq	この信号がタイムアウトになり、実行中の連続測定内で使用できなくなった場合、true。
	IsMeasurable	信号が測定の一部である可能性があるかどうかを示します(Hbm.Api.Common.DaqMeasurementまたはHbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用するなど)。
	Name	信号の名前。
	SampleRate	連続測定中に測定値を収集するために使用するサンプルレート。
	SynchronousMode	信号が測定値を送信する方法を定義します。これは、等距離方式(1秒内の測定値の数 = サンプルレート)または非等距離方式(測定値は予測できない時間に配信される)のいずれかです。

Signal.ContinuousMeasurementValues Property

ストリーミングなどの実行中の測定によって取得された複数の測定値を返します。

C++

```
[JsonIgnore]
__property MeasurementValues ContinuousMeasurementValues;
```

C#

```
[JsonIgnore]
public MeasurementValues ContinuousMeasurementValues;
```

Visual Basic

```
<JsonIgnore>
Public Property ContinuousMeasurementValues() As MeasurementValues
```

Signal.ExcludeFromDaqTimeout Property

連続測定の実行中にこの信号が測定値を送信する必要があるまでのタイムアウト(msec)。この間に新しい測定値がない場合、現在実行中の測定中に信号は無視されます。

C++

```
[JsonIgnore]
__property long ExcludeFromDaqTimeout;
```

C#

```
[JsonIgnore]
public long ExcludeFromDaqTimeout;
```

Visual Basic

```
<JsonIgnore>
Public Property ExcludeFromDaqTimeout() As long
```

Signal.HasSampleRate Property

信号がサンプルレートを使用する場合はtrueを返します。

C++

```
[JsonIgnore]
__property bool HasSampleRate;
```

C#

```
[JsonIgnore]
public bool HasSampleRate;
```

Visual Basic

```
<JsonIgnore>
Public Property HasSampleRate() As bool
```

備考

非同期信号(CanSignalsなど)にもサンプルレートがある場合があります。これは、これらの信号がデバイス自体によってサンプリングされる場合、予測できないタイムスタンプではなく固定サンプル レートを満たすために発生する可能性があります。

Signal.IdleTime Property

連続測定の実行中に測定値を最後に受信してからの時間(msec)。IdleTime >= ExcludeFromDaqTimeoutで、それでも測定値が受信されない場合、IsExcludedFromDaqはtrueに設定され、現在実行中の測定中に信号は無視されます。

C++

```
[JsonIgnore]
__property long IdleTime;
```

C#

```
[JsonIgnore]
public long IdleTime;
```

Visual Basic

```
<JsonIgnore>
Public Property IdleTime() As long
```

Signal.IsExcludedFromDaq Property

この信号がタイムアウトになり、実行中の連続測定内で使用できなくなった場合、true。

C++

```
[JsonIgnore]
__property bool IsExcludedFromDaq;
```

C#

```
[JsonIgnore]
public bool IsExcludedFromDaq;
```

Visual Basic

```
<JsonIgnore>
Public Property IsExcludedFromDaq() As bool
```

Signal.IsMeasurable Property

信号が測定の一部である可能性があるかどうかを示します(Hbm.Api.Common.DaqMeasurementまたはHbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用するなど)。

C++

```
__property bool IsMeasurable;
```

C#

```
public bool IsMeasurable;
```

Visual Basic

```
Public Property IsMeasurable() As bool
```

Signal.Name Property

信号の名前。

C++

```
__property string Name;
```

C#

```
public string Name;
```

Visual Basic

```
Public Property Name() As string
```

Signal.SampleRate Property

連続測定中に測定値を収集するために使用するサンプルレート。

C++

```
__property decimal SampleRate;
```

C#

```
public decimal SampleRate;
```

Visual Basic

```
Public Property SampleRate() As decimal
```

Signal.SynchronousMode Property

信号が測定値を送信する方法を定義します。これは、等距離方式(1秒内の測定値の数 = サンプルレート)または非等距離方式(測定値は予測できない時間に配信される)のいずれかです。

C++

```
__property SynchronousMode SynchronousMode;
```

C#

```
public SynchronousMode SynchronousMode;
```

Visual Basic

```
Public Property SynchronousMode() As SynchronousMode
```

SyncSignal Class

物理センサ(アナログまたはファイバブラッググレーティングセンサ)によって送られる測定信号を表します。



C++

```
class SyncSignal : public Signal, public IFilterable;
```

C#

```
public abstract class SyncSignal : Signal, IFilterable;
```

Visual Basic

```
Public abstract Class SyncSignal
Inherits Signal
Inherits IFilterable
```

ファイル

SyncSignal.cs

SyncSignal Members

次の表は、SyncSignalによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	CreateCircularBuffer	内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。!
	FreeCircularBuffer	内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Methods

	名前	説明
	GetSingleMeasurementValue	単一の測定値を返します。この値を更新するには、Hbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用する必要があります。
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。
	SetMeasurementValues	これは、SetMeasurementValuesメソッドの多重定義の概要です。
	SetSingleMeasurementValue	単一の測定値を設定します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

Public Properties

	名前	説明
	ContinuousMeasurementValues	ストリーミングなどの実行中の測定によって取得された複数の測定値を返します。
	ExcludeFromDaqTimeout	連続測定の実行中にこの信号が測定値を送信する必要があるまでのタイムアウト(msec)。この間に新しい測定値がない場合、現在実行中の測定中に信号は無視されます。
	HasSampleRate	信号がサンプルレートを使用する場合はtrueを返します。
	IdleTime	連続測定の実行中に測定値を最後に受信してからの時間(msec)。IdleTime >= ExcludeFromDaqTimeoutで、それでも測定値が受信されない場合、IsExcludedFromDaqはtrueに設定され、現在実行中の測定中に信号は無視されます。
	IsExcludedFromDaq	この信号がタイムアウトになり、実行中の連続測定内で使用できなくなった場合、true。
	IsMeasurable	信号が測定の一部である可能性があるかどうかを示します(Hbm.Api.Common.DaqMeasurementまたはHbm.Api.Common.Entities.Device.ReadSingleMeasurementValueを使用するなど)。
	Name	信号の名前。
	SampleRate	連続測定中に測定値を収集するために使用するサンプルレート。
	SelectedFilter	連続測定時に測定値をフィルタリングするために使用するフィルタ。
	SynchronousMode	信号が測定値を送信する方法を定義します。これは、等距離方式(1秒内の測定値の数 = サンプルレート)または非等距離方式(測定値は予測できない時間に配信される)のいずれかです。
	UseAutoFilter	SelectedFilterをCommon APIで設定する必要があるかどうかを示すフラグ。

SyncSignal Properties

SyncSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	SelectedFilter	連続測定時に測定値をフィルタリングするために使用するフィルタ。
	UseAutoFilter	SelectedFilterをCommon APIで設定する必要があるかどうかを示すフラグ。

SyncSignal.SelectedFilter Property

連続測定時に測定値をフィルタリングするために使用するフィルタ。

C++

```
__property Filter SelectedFilter;
```

C#

```
public Filter SelectedFilter;
```

Visual Basic

```
Public Property SelectedFilter() As Filter
```

SyncSignal.UseAutoFilter Property

SelectedFilterをCommon APIで設定する必要があるかどうかを示すフラグ。

C++

```
__property bool UseAutoFilter;
```

C#

```
public bool UseAutoFilter;
```

Visual Basic

```
Public Property UseAutoFilter() As bool
```

備考

フィルタは次の主要なデータによって計算されます: デバイスがベッセルフィルタをサポートしている場合、ベッセルフィルタが適用されます。フィルタ周波数は信号のサンプル レートの15%に設定されます。

VirtualSignal Class

物理センサによって送られたものではない測定信号を表します(演算信号など)。



C++

```
class VirtualSignal : public Signal;
```

C#

```
public abstract class VirtualSignal : Signal;
```

Visual Basic

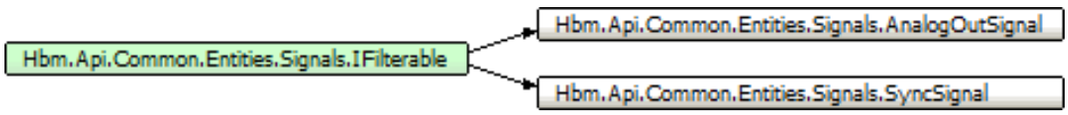
```
Public abstract Class VirtualSignal
Inherits Signal
```

ファイル

VirtualSignal.cs

IFilterable Interface

すべてのフィルタリング可能な信号用のインタフェース。一部の信号には関連付けられたフィルタがないため、これらの信号はフィルタリングできません。



C++

```
__interface IFilterable;
```

C#

```
public interface IFilterable;
```

Visual Basic

```
Public Interface IFilterable
```

ファイル

IFilterable.cs

IFilterable Members

次の表は、IFilterableによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	SelectedFilter	連続測定時に測定値をフィルタリングするために使用するフィルタ。

IFilterable Properties

IFilterableクラスのプロパティの一覧です。

Public Properties

	名前	説明
	SelectedFilter	連続測定時に測定値をフィルタリングするために使用するフィルタ。

IFilterable.SelectedFilter Property

連続測定時に測定値をフィルタリングするために使用するフィルタ。

C++

```
__property Filter SelectedFilter;
```

C#

```
Filter SelectedFilter;
```

Visual Basic

```
Property SelectedFilter() As Filter
```

IGenericSignal Interface

一般的な信号用のインタフェース。つまり、一般的な測定結果タイプ(測定値として CanRawMessagesを配信する CanRawSignalなど)を持つ信号を意味します。



C++

```
__interface IGenericSignal;
```

C#

```
public interface IGenericSignal;
```

Visual Basic

```
Public Interface IGenericSignal
```

ファイル

IGenericSignal.cs

IGenericSignal Members

次の表は、IGenericSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CopyAllValuesFromCircularBuffer ToMeasurementValues	すべての測定値を循環バッファから現在の測定値インスタンスにコピーします。これは内部使用のみを目的としています!
	CreateGenericCircularBuffer	内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。!
	FreeCircularBuffer	内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています!
	InitGenericMeasurementValues	内部の汎用測定値を初期化します。このメソッドは内部使用のみを目的としています!
	SetGenericMeasurementValues	指定した同じサイズを追加します(timeStamps、measurementValues、states)。配列を信号の内部汎用循環バッファに保存します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください! これは内部使用のみを目的としています!

Public Properties

	名前	説明
	GenericMeasurementValues	ストリーミングなどの実行中の測定を通じて取得した複数の測定値を返します。値が常にdouble型であるSignal.ContinuousMeasurementValuesとは異なり、GenericMeasurementValuesには値として任意のデータ型(object)を含めることができます。現時点では、これはCanRawSignalの場合の CanRawMessagesにのみ使用されます。

IGenericSignal Methods

IGenericSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	CopyAllValuesFromCircularBuffer ToMeasurementValues	すべての測定値を循環バッファから現在の測定値インスタンスにコピーします。これは内部使用のみを目的としています!
	CreateGenericCircularBuffer	内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。!
	FreeCircularBuffer	内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています!
	InitGenericMeasurementValues	内部の汎用測定値を初期化します。このメソッドは内部使用のみを目的としています!
	SetGenericMeasurementValues	指定した同じサイズを追加します(timeStamps、measurementValues、states)。配列を信号の内部汎用循環バッファに保存します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください! これは内部使用のみを目的としています!

IGenericSignal.CopyAllValuesFromCircularBufferToMeasurementValues Method

すべての測定値を循環バッファから現在の測定値インスタンスにコピーします。これは内部使用のみを目的としています!

C++

```
void CopyAllValuesFromCircularBufferToMeasurementValues();
```

C#

```
void CopyAllValuesFromCircularBufferToMeasurementValues();
```

Visual Basic

```
Function CopyAllValuesFromCircularBufferToMeasurementValues() As void
```

IGenericSignal.CreateGenericCircularBuffer Method

内部循環バッファを作成します。このバッファはデータ取得中に使用し、その後FreeCircularBufferを使用して解放する必要があります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています!

C++

```
void CreateGenericCircularBuffer(
    int bufferTimeout,
    int minNumberOfBufferedValues
);
```

C#

```
void CreateGenericCircularBuffer(
    int bufferTimeout,
    int minNumberOfBufferedValues
);
```

Visual Basic

```
Function CopyAllValuesFromCircularBufferToMeasurementValues(
    bufferTimeout As Integer,
    minNumberOfBufferedValues As Integer
) As void
```

引数

引数	説明
bufferTimeout	ミリ秒単位のタイムアウト(バッファは信号のサンプルレートと指定されたタイムアウトに従って割り当てられます)。
minNumberOfBufferedValues	循環バッファにバッファする値の最小数(デフォルトは1000)。

備考

バッファリングされる値の数を明示的に設定する場合は、バッファタイムアウトを0に設定し、バッファリングされる値の最小数を要求された値に設定します。

IGenericSignal.FreeGenericCircularBuffer Method

内部循環バッファを解放します。データ収集が停止されると、このバッファは必要なくなります。循環バッファに関する特別な調整が必要な、まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください。これは内部使用のみを目的としています。

C++

```
void FreeGenericCircularBuffer();
```

C#

```
void FreeGenericCircularBuffer();
```

Visual Basic

```
Function FreeGenericCircularBuffer() As void
```

IGenericSignal.InitGenericMeasurementValues Method

内部の汎用測定値を初期化します。このメソッドは内部使用のみを目的としています!

C++

```
void InitGenericMeasurementValues();
```

C#

```
void InitGenericMeasurementValues();
```

Visual Basic

```
Function InitGenericMeasurementValues() As void
```

IGenericSignal.SetGenericMeasurementValues Method

指定した同じサイズを追加します(timeStamps、measurementValues、states)。配列を信号の内部汎用循環バッファに保存します。まだサポートされていないデバイス用に独自のドライバを作成する必要がある場合を除き、この関数は使用しないでください! これは内部使用のみを目的としています!

C++

```
void SetGenericMeasurementValues(
    double[] timeStamps,
    double[] measurementValues,
    MeasurementValueState[] states,
    int numberOfValuesToAdd
);
```

C#

```
void SetGenericMeasurementValues(
    double[] timeStamps,
    double[] measurementValues,
    MeasurementValueState[] states,
    int numberOfValuesToAdd);
```

Visual Basic

```
Function SetGenericMeasurementValues(
    timeStamps As double[],
    measurementValues As double[],
    states As MeasurementValueState[],
    numberOfValuesToAdd As Integer
) As void
```

引数

引数	説明
timeStamps	タイムスタンプ。
measurementValues	測定値。
states	測定値のステータス。
numberOfValuesToAdd	信号の内部循環バッファーに追加する値の数。

例外

例外	説明
System.ArgumentException	numberOfValuesToAdd が0未満の場合にスローされます。timeStamps、measurementValues、またはstatesのサイズがnumberOfValuesToAddより小さい場合にスローされます。
System.ArgumentNullException	timeStamps、measurementValues、またはstatesがnullの場合にスローされます。

IGenericSignal Properties

IFilterableクラスのプロパティの一覧です。

Public Properties

	名前	説明
	GenericMeasurementValues	ストリーミングなどの実行中の測定を通じて取得した複数の測定値を返します。値が常にdouble型であるSignal.ContinuousMeasurementValuesとは異なり、GenericMeasurementValuesには値として任意のデータ型(object)を含めることができます。現時点では、これはCanRawSignalの場合の CanRawMessagesにのみ使用されます。

IGenericSignal.GenericMeasurementValues Property

ストリーミングなどの実行中の測定を通じて取得した複数の測定値を返します。値が常にdouble型であるSignal.ContinuousMeasurementValuesとは異なり、GenericMeasurementValuesには値として任意のデータ型(object)を含めることができます。現時点では、これはCanRawSignalの場合の CanRawMessagesにのみ使用されます。

C++

```
__property IMeasurementValuesBase Filter GenericMeasurementValues;
```

C#

```
IMeasurementValuesBase Interface GenericMeasurementValues;
```

Visual Basic

```
Property GenericMeasurementValues() As IMeasurementValuesBase Interface
```

Hbm.Api.Common.Entities.Statuses Namespace

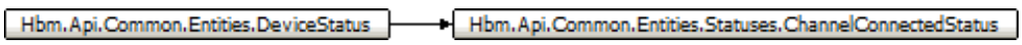
これは名前空間Hbm.Api.Common.Entities.Statusesです。

クラス

	名前	説明
	ChannelConnectedStatus	1チャネルのセンサ接続状態情報を表します。
	ChannelOverflowStatus	1つのチャネルのオーバーフローステータス情報を表します。
	ChannelTedsStatus	1つのチャネルのTEDSステータス情報を表します。
	ConnectorParamBusyStatus	1つのコネクタのパラメータ化ビジーステータス情報を表します。
	DeviceBusyStatus	1つのデバイスのオーバーフローステータス情報を表します。
S	DeviceStatusExtensions	DeviceStatusの拡張機能。

ChannelConnectedStatus Class

1チャネルのセンサ接続状態情報を表します。



C++

```
class ChannelConnectedStatus : public DeviceStatus;
```

C#

```
public class ChannelConnectedStatus : DeviceStatus;
```

Visual Basic

```
Public Class ChannelConnectedStatus
Inherits DeviceStatus
```

ファイル

ChannelConnectedStatus.cs

ChannelConnectedStatus Members

次の表は、ChannelConnectedStatusによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ChannelConnectedStatus	T:Hbm.Api.Common.Entities.Statuses.ChannelConnectedStatusクラスの新しいインスタンスを初期化します。
	DeviceStatus	これは、DeviceStatus Constructorの多重定義の概要です。
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。
	ToString	このインスタンスを表すSystem.Stringを返します。

Public Properties

	名前	説明
	Channel	このステータスのチャネルを取得します。
	Connector	このステータスのコネクタを取得します。
	OriginalID	このステータスの元の識別子(デバイス固有)を取得します。
	Signal	このステータスの信号を取得します。
	StatusType	このステータスの種類を取得します。
	Value	このステータスの値を取得します。

ChannelConnectedStatus.ChannelConnectedStatus Constructor

T:Hbm.Api.Common.Entities.Statuses.ChannelConnectedStatusクラスの新しいインスタンスを初期化します。

C++

```
ChannelConnectedStatus(
    int originalId,
    ChannelConnectedValueType value,
    Channel channel
);
```

C#

```
public ChannelConnectedStatus(
    int originalId,
    ChannelConnectedValueType value,
    Channel channel
);
```

Visual Basic

```
Public Sub New(
    originalId As Integer
    value As ChannelConnectedValueType,
    channel As Channel
)
```

引数

引数	説明
originalId	元の識別子。
value	値。
channel	チャネル。

ChannelConnectedStatus Methods

ChannelConnectedStatusクラスのメソッドの一覧です。

Public Methods

	名前	説明
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

ChannelConnectedStatus.IsEqual Method

指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

C++

```
bool IsEqual(
    DeviceStatus status
);
```

C#

```
public override bool IsEqual(
    DeviceStatus status
);
```

Visual Basic

```
Public override Function IsEqual(
    states As DeviceStatus
) As bool
```

引数

引数	説明
states	ステータス。

戻り値

指定したステータスが等しい場合は`true`。それ以外の場合は`false`。

ChannelConnectedStatus Properties

ChannelConnectedStatusクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Value	

ChannelConnectedStatus.Value Property

このステータスの値を取得します。

C++

```
__property ChannelConnectedValueType Value;
```

C#

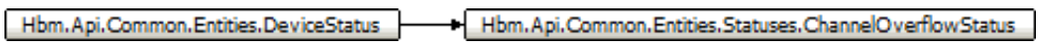
```
public ChannelConnectedValueType Value;
```

Visual Basic

```
Public Property Value() As ChannelConnectedValueType
```

ChannelOverflowStatus Class

1つのチャネルのオーバーフローステータス情報を表します。



C++

```
class ChannelOverflowStatus : public DeviceStatus;
```

C#

```
public class ChannelOverflowStatus : DeviceStatus;
```

Visual Basic

```
Public Class ChannelOverflowStatus
Inherits DeviceStatus
```

ファイル

ChannelOverflowStatus.cs

ChannelOverflowStatus Members

次の表は、ChannelOverflowStatusによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ChannelOverflowStatus	T:Hbm.Api.Common.Entities.Statuses.ChannelOverflowStatusクラスの新しいインスタンスを初期化します。
	DeviceStatus	これは、DeviceStatus Constructorの多重定義の概要です。
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。
	ToString	このインスタンスを表すSystem.Stringを返します。

Public Properties

	名前	説明
	Channel	このステータスのチャネルを取得します。
	Connector	このステータスのコネクタを取得します。
	OriginalID	このステータスの元の識別子(デバイス固有)を取得します。
	Signal	このステータスの信号を取得します。
	StatusType	このステータスの種類を取得します。
	Value	このステータスの値を取得します。

ChannelOverflowStatus.ChannelOverflowStatus Constructor

T:Hbm.Api.Common.Entities.Statuses.ChannelOverflowStatusクラスの新しいインスタンスを初期化します。

C++

```
ChannelOverflowStatus(
    int originalId,
    ChannelOverflowValueType value,
    Channel channel
);
```

C#

```
public ChannelOverflowStatus(
    int originalId,
    ChannelOverflowValueType value,
    Channel channel
);
```

Visual Basic

```
Public Sub New(
    originalId As Integer
    value As ChannelOverflowValueType,
    channel As Channel
)
```

引数

引数	説明
originalId	元の識別子。
value	値。
channel	チャネル。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Hbm.Api.Common.Entities.Statuses Namespace -> ChannelOverflowStatus Class

ChannelOverflowStatus Methods

ChannelOverflowStatus クラスのメソッドの一覧です。

Public Methods

	名前	説明
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

ChannelOverflowStatus.IsEqual Method

指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

C++

```
bool IsEqual(
    DeviceStatus status
);
```

C#

```
public override bool IsEqual(
    DeviceStatus status
);
```

Visual Basic

```
Public override Function IsEqual(
    states As DeviceStatus
) As bool
```

引数

引数	説明
states	ステータス。

戻り値

指定したステータスが等しい場合は`true`。それ以外の場合は`false`。

ChannelOverflowStatus Properties

ChannelOverflowStatusクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Value	このステータスの値を取得します。

ChannelOverflowStatus.Value Property

このステータスの値を取得します。

C++

```
[PublicAPI]
__property ChannelOverflowValueType Value;
```

C#

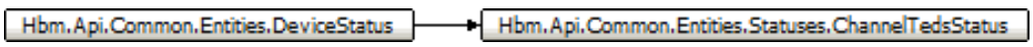
```
[PublicAPI]
public ChannelOverflowValueType Value;
```

Visual Basic

```
<PublicAPI>
Public Property Value() As ChannelOverflowValueType
```

ChannelTedsStatus Class

1つのチャネルのTEDSステータス情報を表します。



C++

```
class ChannelTedsStatus : public DeviceStatus;
```

C#

```
public class ChannelTedsStatus : DeviceStatus;
```

Visual Basic

```
Public Class ChannelTedsStatus
Inherits DeviceStatus
```

ファイル

ChannelTedsStatus.cs

ChannelTedsStatus Members

次の表は、ChannelTedsStatusによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ChannelTedsStatus	ChannelTedsStatusクラスの新しいインスタンスを初期化します。
	DeviceStatus	これは、DeviceStatus Constructorの多重定義の概要です。
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。
	ToString	このインスタンスを表すSystem.Stringを返します。

Public Properties

	名前	説明
	Channel	このステータスのチャネルを取得します。
	Connector	このステータスのコネクタを取得します。
	OriginalID	このステータスの元の識別子(デバイス固有)を取得します。
	Signal	このステータスの信号を取得します。
	StatusType	このステータスの種類を取得します。
	Value	このステータスの値を取得します。

ChannelTedsStatus.ChannelTedsStatus Constructor

ChannelTedsStatusクラスの新しいインスタンスを初期化します。

C++

```
ChannelTedsStatus(
    int originalId,
    TedsValidationValueType value,
    Channel channel
);
```

C#

```
public ChannelTedsStatus(
    int originalId,
    TedsValidationValueType value,
    Channel channel
);
```

Visual Basic

```
Public Sub New(
    originalId As Integer
    value As TedsValidationValueType,
    channel As Channel
)
```

引数

引数	説明
originalId	元の識別子。
value	値。
channel	チャネル。

ChannelTedsStatus Methods

ChannelTedsStatusクラスのメソッドの一覧です。

Public Methods

	名前	説明
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

ChannelTedsStatus.IsEqual Method

指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

C++

```
bool IsEqual(
    DeviceStatus status
);
```

C#

```
public override bool IsEqual(
    DeviceStatus status
);
```

Visual Basic

```
Public override Function IsEqual(
    states As DeviceStatus
) As bool
```

引数

引数	説明
states	ステータス。

戻り値

指定したステータスが等しい場合は`true`。それ以外の場合は`false`。

ChannelTedsStatus Properties

ChannelTedsStatusクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Value	このステータスの値を取得します。

ChannelTedsStatus.Value Property

このステータスの値を取得します。

C++

```
__property TedsValidationValueType Value;
```

C#

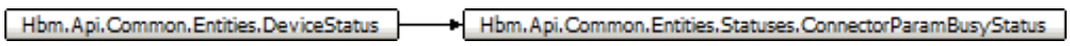
```
public TedsValidationValueType Value;
```

Visual Basic

```
Public Property Value() As TedsValidationValueType
```

ConnectorParamBusyStatus Class

1つのコネクタのパラメータ化ビジーステータス情報を表します。



C++

```
class ConnectorParamBusyStatus : public DeviceStatus;
```

C#

```
public class ConnectorParamBusyStatus : DeviceStatus;
```

Visual Basic

```
Public Class ConnectorParamBusyStatus
Inherits DeviceStatus
```

ファイル

ConnectorParamBusyStatus.cs

ConnectorParamBusyStatus Members

次の表は、ConnectorParamBusyStatusによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ConnectorParamBusyStatus	ConnectorParamBusyStatusクラスの新しいインスタンスを初期化します。
	DeviceStatus	これは、DeviceStatus Constructorの多重定義の概要です。
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。
	ToString	このインスタンスを表すSystem.Stringを返します。

Public Properties

	名前	説明
	Channel	このステータスのチャネルを取得します。
	Connector	このステータスのコネクタを取得します。
	OriginalID	このステータスの元の識別子(デバイス固有)を取得します。
	Signal	このステータスの信号を取得します。
	StatusType	このステータスの種類を取得します。
	Value	このステータスの値を取得します。

ConnectorParamBusyStatus.ConnectorParamBusyStatus Constructor

ConnectorParamBusyStatusクラスの新しいインスタンスを初期化します。

C++

```
ConnectorParamBusyStatus(
    int originalId,
    ConnectorParamBusyValueType value,
    Connector connector
);
```

C#

```
public ConnectorParamBusyStatus(
    int originalId,
    ConnectorParamBusyValueType value,
    Connector connector
);
```

Visual Basic

```
Public Sub New(
    originalId As Integer
    value As ConnectorParamBusyValueType,
    connector As Connector
)
```

引数

引数	説明
originalId	元の識別子。
value	値。
connector	コネクタ。

ConnectorParamBusyStatus Methods

ConnectorParamBusyStatusクラスのメソッドの一覧です。

Public Methods

	名前	説明
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

ConnectorParamBusyStatus.IsEqual Method

指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

C++

```
bool IsEqual(
    DeviceStatus status
);
```

C#

```
public override bool IsEqual(
    DeviceStatus status
);
```

Visual Basic

```
Public override Function IsEqual(
    states As DeviceStatus
) As bool
```

引数

引数	説明
states	ステータス。

戻り値

指定したステータスが等しい場合は`true`。それ以外の場合は`false`。

ConnectorParamBusyStatus Properties

ConnectorParamBusyStatusクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Value	このステータスの値を取得します。

ConnectorParamBusyStatus.Value Property

このステータスの値を取得します。

C++

```
__property ConnectorParamBusyValueType Value;
```

C#

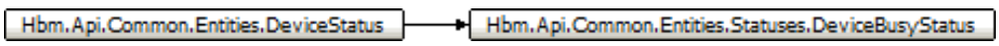
```
public ConnectorParamBusyValueType Value;
```

Visual Basic

```
Public Property Value() As ConnectorParamBusyValueType
```

DeviceBusyStatus Class

1つのデバイスのオーバーフローステータス情報を表します。



C++

```
class DeviceBusyStatus : public Device;
```

C#

```
public class DeviceBusyStatus : Device;
```

Visual Basic

```
Public Class DeviceBusyStatus
Inherits Device
```

ファイル

DeviceBusyStatus.cs

DeviceBusyStatus Members

次の表は、DeviceBusyStatusによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DeviceBusyStatus	ConnectorParamBusyStatusクラスの新しいインスタンスを初期化します。
	DeviceStatus	これは、DeviceStatus Constructorの多重定義の概要です。
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。
	ToString	このインスタンスを表すSystem.Stringを返します。

Public Properties

	名前	説明
	Channel	このステータスのチャネルを取得します。
	Connector	このステータスのコネクタを取得します。
	OriginalID	このステータスの元の識別子(デバイス固有)を取得します。
	Signal	このステータスの信号を取得します。
	StatusType	このステータスの種類を取得します。
	Value	このステータスの値を取得します。

DeviceBusyStatus.DeviceBusyStatus Constructor

ConnectorParamBusyStatusクラスの新しいインスタンスを初期化します。

C++

```
DeviceBusyStatus(
    int originalId,
    DeviceBusyValueType value
);
```

C#

```
public DeviceBusyStatus(
    int originalId,
    DeviceBusyValueType value
);
```

Visual Basic

```
Public Sub New(
    value As DeviceBusyValueType
)
```

引数

引数	説明
originalId	元の識別子。
value	ステータス値。

DeviceBusyStatus Methods

DeviceBusyStatusクラスのメソッドの一覧です。

Public Methods

	名前	説明
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

DeviceBusyStatus.IsEqual Method

指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

C++

```
bool IsEqual(
    DeviceStatus status
);
```

C#

```
public override bool IsEqual(
    DeviceStatus status
);
```

Visual Basic

```
Public override Function IsEqual(
    states As DeviceStatus
) As bool
```

引数

引数	説明
states	ステータス。

DeviceBusyStatus Properties

DeviceBusyStatusクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Value	このステータスの値を取得します。

DeviceBusyStatus.Value Property

このステータスの値を取得します。

C++

```
[PublicAPI]
__property DeviceBusyValueType Value;
```

C#

```
[PublicAPI]
public DeviceBusyValueType Value;
```

Visual Basic

```
<PublicAPI>
Public Property Value() As DeviceBusyValueType
```

DeviceStatusExtensions Class

DeviceStatusの拡張機能。



C++

```
class DeviceStatusExtensions;
```

C#

```
public static class DeviceStatusExtensions;
```

Visual Basic

```
Public static Class DeviceStatusExtensions
```

ファイル

DeviceStatusExtensions.cs

DeviceStatusExtensions Members

次の表は、DeviceStatusExtensionsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
S	FindStatusWithSameSource	指定したステータスと同じソースを持つステータスをリストから検索します。
S	HasSameSource	指定したステータスのソースが同じかどうかを判断します。
S	ReplaceStatusWithSameSource	指定したステータスと同じソースを持つリスト内のステータスを置き換えます。

DeviceStatusExtensions Methods

DeviceStatusExtensionsクラスのメソッドの一覧です。

Public Methods

	名前	説明
S	FindStatusWithSameSource	指定したステータスと同じソースを持つステータスをリストから検索します。
S	HasSameSource	指定したステータスのソースが同じかどうかを判断します。
S	ReplaceStatusWithSameSource	指定したステータスと同じソースを持つリスト内のステータスを置き換えます。

DeviceStatusExtensions.FindStatusWithSameSource Method

指定したステータスと同じソースを持つステータスをリストから検索します。

C++

```
DeviceStatus FindStatusWithSameSource(
    this List<DeviceStatus> statusList,
    DeviceStatus status
);
```

C#

```
public static DeviceStatus FindStatusWithSameSource(
    this List<DeviceStatus> statusList,
    DeviceStatus status
);
```

Visual Basic

```
Public static Function FindStatusWithSameSource(
    statusList As this List<DeviceStatus>,
    states As DeviceStatus
) As DeviceStatus
```

引数

引数	説明
statusList	ステータスリスト。
states	比較対象となるステータス。

戻り値

指定したステータスと同じソースを持つステータス、または見つからない場合は`null`。

DeviceStatusExtensions.HasSameSource Method

指定したステータスのソースが同じかどうかを判断します。

C++

```
bool HasSameSource(
    this DeviceStatus thisStatus,
    DeviceStatus status
);
```

C#

```
public static bool HasSameSource(
    this DeviceStatus thisStatus,
    DeviceStatus status
);
```

Visual Basic

```
Public static Function HasSameSource(
    thisStatus As this DeviceStatus,
    states As DeviceStatus
) As bool
```

引数

引数	説明
thisStatus	このステータス。
states	比較対象となるステータス。

戻り値

指定したステータスのソースが同じ場合は`true`。それ以外の場合は`false`。

DeviceStatusExtensions.ReplaceStatusWithSameSource Method

指定したステータスと同じソースを持つリスト内のステータスを置き換えます。

C++

```
void ReplaceStatusWithSameSource(
    this List<DeviceStatus> statusList,
    DeviceStatus status
);
```

C#

```
public static void ReplaceStatusWithSameSource(
    this List<DeviceStatus> statusList,
    DeviceStatus status
);
```

Visual Basic

```
Public static Function ReplaceStatusWithSameSource(
    statusList As this List<DeviceStatus>,
    states As DeviceStatus
) As void
```

引数

引数	説明
statusList	ステータスリスト。
states	置き換えるステータス。

戻り値

指定したステータスと同じソースを持つステータス、または見つからない場合は`null`。

Hbm.Api.Common.Entities.SpectrumInfos Namespace

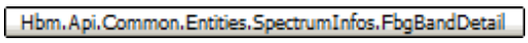
これは名前空間Hbm.Api.Common.Entities.SpectrumInfosです。

クラス

名前	説明
FbgBandDetail	ファイバブラッググレーティング対応デバイスの帯域に関する詳細を保持するクラス。
FbgBandsDetectionResult	特定のファイバの帯域検出結果を保持するクラス。
FbgFiberInfo	ファイバブラッググレーティング対応デバイスのファイバひずみ関連データを保持するクラス。
FbgSpectrum	ファイバブラッググレーティング対応デバイスから要求されたスペクトルデータを保持するクラス。

FbgBandDetail Class

ファイバブラッググレーティング対応デバイスの帯域に関する詳細を保持するクラス。



C++

```
class FbgBandDetail;
```

C#

```
public class FbgBandDetail;
```

Visual Basic

```
Public Class FbgBandDetail
```

ファイル

FbgBandDetail.cs

FbgBandDetail Members

次の表は、FbgBandDetailによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	FbgBandDetail	コンストラクタ。

FbgBandDetail.FbgBandDetail Constructor

コンストラクタ。

C++

```
FbgBandDetail();
```

C#

```
public FbgBandDetail();
```

Visual Basic

```
Public Sub New()
```

FbgBandDetail Properties

FbgBandDetailクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Amplitude	振幅(dB)。
	DetectionRange	検出範囲。検出されたピークが有効な波長(帯域)をnm単位で定義します。
	PositionIndex	バンドの位置インデックス(ゼロベース)。
	ReferenceWavelength	帯域内の基準波長(nm)。

FbgBandDetail.Amplitude Property

振幅(dB)。

C++

```
__property double Amplitude;
```

C#

```
public double Amplitude;
```

Visual Basic

```
Public Property Amplitude() As double
```

FbgBandDetail.DetectionRange Property

検出範囲。検出されたピークが有効な波長(帯域)をnm単位で定義します。

C++

```
__property DetectionRange DetectionRange;
```

C#

```
public DetectionRange DetectionRange;
```

Visual Basic

```
Public Property DetectionRange() As DetectionRange
```

FbgBandDetail.PositionIndex Property

バンドの位置インデックス(ゼロベース)。

C++

```
__property int PositionIndex;
```

C#

```
public int PositionIndex;
```

Visual Basic

```
Public Property PositionIndex() As Integer
```

FbgBandDetail.ReferenceWavelength Property

帯域内の基準波長(nm)。

C++

```
__property double ReferenceWavelength;
```

C#

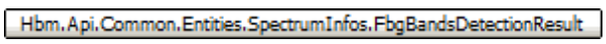
```
public double ReferenceWavelength;
```

Visual Basic

```
Public Property ReferenceWavelength() As double
```

FbgBandsDetectionResult Class

特定のファイバの帯域検出結果を保持するクラス。



C++

```
class FbgBandsDetectionResult;
```

C#

```
public class FbgBandsDetectionResult;
```

Visual Basic

```
Public Class FbgBandsDetectionResult
```

ファイル

FbgBandsDetectionResult.cs

FbgBandsDetectionResult Members

次の表は、FbgBandsDetectionResultによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	FbgBandsDetectionResult	コンストラクタ。

Public Properties

	名前	説明
	BandWidth	BandWidth、またはピークの自動検出検索内の単一の帯域幅。
	DetectedBands	自動検出された帯域のリスト。
	Threshold	自動検出に使用するしきい値。

FbgBandsDetectionResult.FbgBandsDetectionResult Constructor

コンストラクタ。

C++

```
FbgBandsDetectionResult();
```

C#

```
public FbgBandsDetectionResult();
```

Visual Basic

```
Public Sub New()
```

FbgBandsDetectionResult Properties

FbgBandsDetectionResultクラスのプロパティの一覧です。

Public Properties

	名前	説明
	BandWidth	BandWidth、またはピークの自動検出検索内の単一の帯域幅。
	DetectedBands	自動検出された帯域のリスト。
	Threshold	自動検出に使用するしきい値。

FbgBandsDetectionResult.BandWidth Property

BandWidth、またはピークの自動検出検索内の単一帯域幅。

C++

```
__property double BandWidth;
```

C#

```
public double BandWidth;
```

Visual Basic

```
Public Property BandWidth() As double
```

FbgBandsDetectionResult.DetectedBands Property

自動検出された帯域のリスト。

C++

```
__property List<FbgBandDetail> DetectedBands;
```

C#

```
public List<FbgBandDetail> DetectedBands;
```

Visual Basic

```
Public Property DetectedBands() As List<FbgBandDetail>
```

FbgBandsDetectionResult.Threshold Property

自動検出に使用するしきい値。

C++

```
__property double Threshold;
```

C#

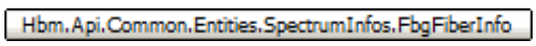
```
public double Threshold;
```

Visual Basic

```
Public Property Threshold() As double
```

FbgFiberInfo Class

ファイバブラッググレーティング対応デバイスのファイバひずみ関連データを保持するクラス。



C++

```
class FbgFiberInfo;
```

C#

```
public class FbgFiberInfo;
```

Visual Basic

```
Public Class FbgFiberInfo
```

ファイル

FbgFiberInfo.cs

FbgFiberInfo Members

次の表は、FbgFiberInfoによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	MinBandGap	2つの連続するチャネル帯域間の最小スペクトルギャップ(nm)。
	MinBandWidth	必要な最小帯域幅(nm)。

FbgFiberInfo Properties

FbgFiberInfoクラスのプロパティの一覧です。

Public Properties

	名前	説明
	MinBandGap	2つの連続するチャネル帯域間の最小スペクトルギャップ(nm)。
	MinBandWidth	必要な最小帯域幅(nm)。

FbgFiberInfo.MinBandGap Property

2つの連続するチャネル帯域間の最小スペクトルギャップ(nm)。

C++

```
__property double MinBandGap;
```

C#

```
public double MinBandGap;
```

Visual Basic

```
Public Property MinBandGap() As double
```

FbgFiberInfo.MinBandWidth Property

必要な最小帯域幅(nm)。

C++

```
__property double MinBandWidth;
```

C#

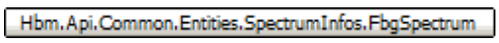
```
public double MinBandWidth;
```

Visual Basic

```
Public Property MinBandWidth() As double
```

FbgSpectrum Class

ファイバブラッググレーティング対応デバイスから要求されたスペクトルデータを保持するクラス。



C++

```
class FbgSpectrum;
```

C#

```
public class FbgSpectrum;
```

Visual Basic

```
Public Class FbgSpectrum
```

ファイル

FbgSpectrum.cs

FbgSpectrum Members

次の表は、FbgSpectrumによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	FbgSpectrum	コンストラクタ。

Public Properties

	名前	説明
	Amplitudes	振幅(スペクトル)。
	Delta	2つの振幅(y)間のデルタ(x) (nm)。
	SpectrumType	スペクトルのタイプ。
	Start	スペクトルの開始(最初のx単位はnm)。

FbgSpectrum.FbgSpectrum Constructor

コンストラクタ。

C++

```
FbgSpectrum();
```

C#

```
public FbgSpectrum();
```

Visual Basic

```
Public Sub New()
```

FbgSpectrum Properties

FbgSpectrumクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Amplitudes	振幅(スペクトル)。
	Delta	2つの振幅(y)間のデルタ(x) (nm)。
	SpectrumType	スペクトルのタイプ。
	Start	スペクトルの開始(最初のx単位はnm)。

FbgSpectrum.Amplitudes Property

振幅(スペクトル)。

C++

```
__property double[] Amplitudes;
```

C#

```
public double[] Amplitudes;
```

Visual Basic

```
Public Property Amplitudes() As double[]
```

FbgSpectrum.Delta Property

2つの振幅(y)間のデルタ(x) (nm)。

C++

```
__property double Delta;
```

C#

```
public double Delta;
```

Visual Basic

```
Public Property Delta() As double
```

FbgSpectrum.SpectrumType Property

スペクトルのタイプ。

C++

```
__property FbgSpectrumType SpectrumType;
```

C#

```
public FbgSpectrumType SpectrumType;
```

Visual Basic

```
Public Property SpectrumType() As FbgSpectrumType
```

FbgSpectrum.Start Property

スペクトルの開始(最初のx単位はnm)。

C++

```
__property double Start;
```

C#

```
public double Start;
```

Visual Basic

```
Public Property Start() As double
```

Hbm.Api.Common.Entities.TimeSources Namespace

デバイスを同期すると、モジュールのクロックが可能な限り相互に同期して実行されます。Common APIは通常、タイムスタンプを使用して複数のデバイス間で同期計測の開始を実現するため、この場合は同じクロックのデバイスが必須です。

同じファミリの複数のデバイスを計測する場合、通常すべてのファミリがケーブルベースの同期方法(MGCとPMXの場合はそれぞれの同期ケーブル、QuantumXの場合はFireWireケーブル)をサポートしており、クロックが確実に同期されるため、同期に関する問題は発生しません。ただし、使用するデバイスファミリによっては、1つのデバイスのみを計測する場合でも異なるサンプル レートの信号の最初のタイムスタンプが異なる場合があります(例: PMX)。

異なるデバイス ファミリの複数のデバイス(例: PMX と QuantumXを組み合わせ)を測定する場合、または同期ケーブルで接続できない(例: 距離が原因で)1つのファミリの複数のデバイスを測定する場合、次の問題が関連します:

- 同一サンプルレートでも、異なるデバイスの測定値のタイムスタンプが異なる
- 同一サンプルレートでも、異なるデバイスの測定値の測定値数も異なる

これは、デバイスのクロック/クォーツが100%同期して動作しないためです。タイムスタンプ同期測定を実行する場合、APIは常に、同じサンプルレートを使用する信号に対して同じ数の測定値を提供します。サンプルグループ内の"最も遅い"信号(例: 19200Hz)がサンプルグループ全体に配信される測定値の数を定義するため、一定の時間が経過すると、API内で内部バッファオーバーランが発生する可能性があります。適切なタイムソースを使用するか、非同期で測定を開始することにより、影響を最小限に抑えたり排除したりできます。

非同期で測定する場合、APIはサンプル レートグループ内の"最も遅い"信号の測定値数だけではなく、常にすべての信号のすべての測定値を返すため、バッファオーバーランは発生しません。

モジュールを同期するために、APIはいくつかのTimeSourceを提供しますが、これらは必ずしもすべてのデバイスでサポートされているわけではありません:

- AutoTimeSource: デバイスは内部同期方式を使用します。たとえば、FireWireケーブル経由で接続している QuantumX デバイスの数を測定する場合、または同期ケーブルを使用してMGCやPMXデバイスの数を測定する場合に、このタイムソースを使用します。
- NtpTimeSource: デバイスは、ネットワークタイムプロトコル(Network Time Protocol: NTP)を使用して、コンピュータネットワーク全体でクロックを特定のタイムサーバと同期します。ローカルエリアネットワークでは、1ミリ秒を下回るクロック精度を実現します。たとえば、QuantumXとPMXまたはMGCデバイスを並行して測定する場合に、このタイムソースを使用します。
- PtpTimeSource: デバイスは、Precision Time Protocol(PTP)を使用して、コンピュータネットワーク全体でクロックを同期します。ローカルエリアネットワークでは、マイクロ秒未満の範囲のクロック精度を実現します。たとえば、FireWire経由で接続されていないさまざまなQuantumXデバイスを測定する場合に、このタイムソースを使用します。
- EtherCatTimeSource: デバイスは、コントローラおよびオートメーションテクノロジにイーサネットを使用して、他のデバイスと同期します。サブマイクロ秒の範囲のクロック精度を実現します。たとえば、FireWire経由で接続されていないさまざまなQuantumXデバイスを測定する場合に、このタイムソースを使用します。

同期の品質を確認するには、Device.GetTimeSourceQuality(…)を呼び出します。これにより、現在のタイムソースへのオフセットと、使用されているTimeSourceの種類に応じた詳細情報を含む文字列が返されます。

例

```
// Assume we have a list of connected devices:
// First one is e.g. a QuantumXDevice (_deviceList[0])
// Second device is e.g. a PmxDevice  (_deviceList[1])
List<Problem> problemsDuringAssign;
// Setup time source...
foreach(Device device in _deviceList)
{
  // Create an NTP time source with a valid NTP-Server
  device.TimeSource = new NtpTimeSource("172.19.160.111");
  // and assign it
  if (!device.AssignTimeSource(out problemsDuringAssign))
  {
    // error handling: e.g. check the problems here...
  }
}
// wait a while (let the devices synchronize this may take a moment...)
System.Threading.Thread.Sleep(5000);
foreach(Device device in _deviceList)
{
  // check offset to NTP server in ms and further quality parameters
  double offset;
  string quality;
  device.GetTimeSourceQuality(out offset, out quality);
  Console.WriteLine(string.Format("Time offset in ms between NTP server and device {0} = {1}
", device.Name, offset));
}

// Following similar output will be generated:
// Time offset in ms between NTP server and device MX840_RT = -1,072
// Time offset in ms between NTP server and device PMX_01 = -2,141
```

クラス

	名前	説明
	AutoTimeSource	デバイスは内部同期方式を使用します。たとえば、FireWireケーブル経由で接続している QuantumXデバイスの数を測定する場合、または同期ケーブルを使用してMGCやPMXデバイスの数を測定する場合に、このタイムソースを使用します。
	EtherCatTimeSource	デバイスは、コントローラおよびオートメーションテクノロジにイーサネットを使用して、他のデバイスと同期します。サブマイクロ秒の範囲のクロック精度を実現します。たとえば、FireWire経由で接続されていないさまざまなQuantumXデバイスを測定する場合に、このタイムソースを使用します。
	NotSupportedTimeSource	サポートされていない時刻同期タイプの設定を保持するクラス。
	NtpTimeSource	デバイスは、ネットワークタイムプロトコル(Network Time Protocol: NTP)を使用して、コンピュータネットワーク全体でクロックを特定のタイムサーバと同期します。ローカルエリアネットワークでは、1ミリ秒を下回るクロック精度を実現します。たとえば、QuantumXとPMXまたはMGCデバイスを並行して測定する場合に、このタイムソースを使用します。
	PtpTimeSource	デバイスは、Precision Time Protocol(PTP)を使用して、コンピュータネットワーク全体でクロックを同期します。ローカルエリアネットワークでは、マイクロ秒未満の範囲のクロック精度を実現します。たとえば、FireWire経由で接続されていないさまざまなQuantumXデバイスを測定する場合に、このタイムソースを使用します。
A	TimeSource	すべての具体的なタイムソース実装の抽象基本クラス。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace

AutoTimeSource Class

デバイスは内部同期方式を使用します。たとえば、FireWireケーブル経由で接続している QuantumXデバイスの数を測定する場合、または同期ケーブルを使用してMGCやPMXデバイスの数を測定する場合に、このタイムソースを使用します。



C++

```
class AutoTimeSource : public TimeSource;
```

C#

```
public class AutoTimeSource : TimeSource;
```

Visual Basic

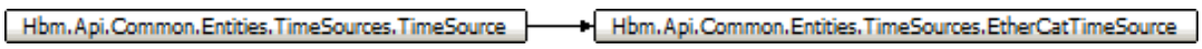
```
Public Class AutoTimeSource
Inherits TimeSource
```

ファイル

AutoTimeSource.cs

EtherCatTimeSource Class

デバイスは、コントローラおよびオートメーションテクノロジにイーサネットを使用して、他のデバイスと同期します。サブマイクロ秒の範囲のクロック精度を実現します。たとえば、FireWire経由で接続されていないさまざまなQuantumXデバイスを測定する場合に、このタイムソースを使用します。



C++

```
class EtherCatTimeSource : public TimeSource;
```

C#

```
public class EtherCatTimeSource : TimeSource;
```

Visual Basic

```
Public Class EtherCatTimeSource
Inherits TimeSource
```

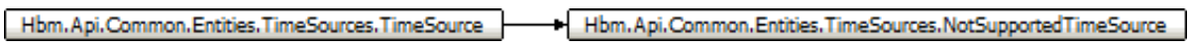
ファイル

EtherCatTimeSource.cs

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Hbm.Api.Common.Entities.TimeSources Namespace

NotSupportedTimeSource Class

サポートされていない時刻同期タイプの設定を保持するクラス。



C++

```
class NotSupportedTimeSource : public TimeSource;
```

C#

```
public class NotSupportedTimeSource : TimeSource;
```

Visual Basic

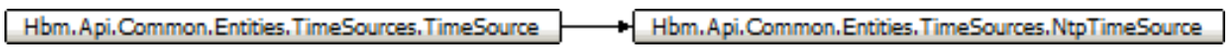
```
Public Class NotSupportedTimeSource
Inherits TimeSource
```

ファイル

NotSupportedTimeSource.cs

NtpTimeSource Class

デバイスは、ネットワークタイムプロトコル(Network Time Protocol: NTP)を使用して、コンピュータネットワーク全体でクロックを特定のタイムサーバと同期します。ローカルエリアネットワークでは、1ミリ秒を下回るクロック精度を実現します。たとえば、QuantumXとPMXまたはMGCデバイスを並行して測定する場合に、このタイムソースを使用します。



C++

```
class NtpTimeSource : public TimeSource;
```

C#

```
public class NtpTimeSource : TimeSource;
```

Visual Basic

```
Public Class NtpTimeSource
Inherits TimeSource
```

ファイル

NtpTimeSource.cs

NtpTimeSource Members

次の表は、NtpTimeSourceによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	NtpTimeSource	指定したパラメータを使用して新しいインスタンスを作成します。
	ToString	このインスタンスを表すSystem.Stringを返します。

Public Properties

	名前	説明
	ServerIpAddress	NTPタイムサーバのIPアドレス。

NtpTimeSource.NtpTimeSource Constructor

指定したパラメータを使用して新しいインスタンスを作成します。

C++

```
NtpTimeSource(
    string serverIpAddress
);
```

C#

```
public NtpTimeSource;
    string serverIpAddress
);
```

Visual Basic

```
Public Sub New(
    serverIpAddress As string
)
```

引数

引数	説明
serverIpAddress	NTPタイムサーバのIPアドレス。

NtpTimeSource Properties

FbgBandDetailクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ServerIpAddress	NTPタイムサーバのIPアドレス。

NtpTimeSource.ServerIpAddress Property

NTPタイムサーバのIPアドレス。

C++

```
__property string ServerIpAddress;
```

C#

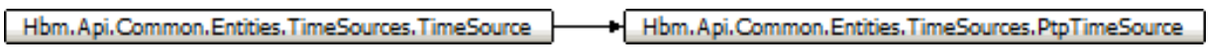
```
public string ServerIpAddress;
```

Visual Basic

```
Public Property ServerIpAddress() As string
```

PtpTimeSource Class

デバイスは、Precision Time Protocol(PTP)を使用して、コンピュータネットワーク全体でクロックを同期します。ローカルエリアネットワークでは、マイクロ秒未満の範囲のクロック精度を実現します。たとえば、FireWire経由で接続されていないさまざまなQuantumXデバイスを測定する場合に、このタイムソースを使用します。



C++

```
class PtpTimeSource : public TimeSource;
```

C#

```
public class PtpTimeSource : TimeSource;
```

Visual Basic

```
Public Class PtpTimeSource
Inherits TimeSource
```

ファイル

PtpTimeSource.cs

PtpTimeSource Members

次の表は、PtpTimeSourceによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	PtpTimeSource	指定したパラメータを使用して新しいインスタンスを作成します。
	ToString	このインスタンスを表すSystem.Stringを返します。

Public Properties

	名前	説明
	DelayMechanism	デバイスの同期に使用するPTP遅延メカニズム。
	TransportMode	デバイスの同期に使用するPTP転送モード。

PtpTimeSource.PtpTimeSource Constructor

指定したパラメータを使用して新しいインスタンスを作成します。

C++

```
PtpTimeSource(
    DelayMechanism delayMechanism,
    TransportMode transportMode
);
```

C#

```
public PtpTimeSource(
    DelayMechanism delayMechanism,
    TransportMode transportMode
);
```

Visual Basic

```
Public Sub New(
    delayMechanism As DelayMechanism
    transportMode As TransportMode
)
```

引数

引数	説明
delayMechanism	デバイスの同期に使用するPTP遅延メカニズム。
transportMode	デバイスの同期に使用するPTP転送モード。

PtpTimeSource Properties

PtpTimeSourceクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DelayMechanism	デバイスの同期に使用するPTP遅延メカニズム。
	TransportMode	デバイスの同期に使用するPTP転送モード。

PtpTimeSource.DelayMechanism Property

デバイスの同期に使用するPTP遅延メカニズム。

C++

```
__property DelayMechanism DelayMechanism;
```

C#

```
public DelayMechanism DelayMechanism;
```

Visual Basic

```
Public Property DelayMechanism() As DelayMechanism
```

PtpTimeSource.TransportMode Property

デバイスの同期に使用するPTP転送モード。

C++

```
__property TransportMode TransportMode;
```

C#

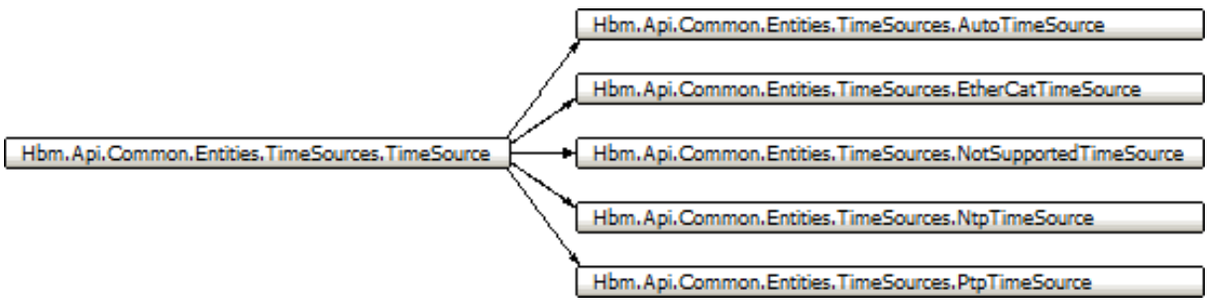
```
public TransportMode TransportMode;
```

Visual Basic

```
Public Property TransportMode() As TransportMode
```

TimeSource Class

すべての具体的なタイムソース実装の抽象基本クラス。



C++

```
class TimeSource;
```

C#

```
public abstract class TimeSource;
```

Visual Basic

```
Public abstract Class TimeSource
```

ファイル

TimeSource.cs

TimeSource Members

次の表は、TimeSourceによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ToString	このインスタンスを表すSystem.Stringを返します。

TimeSource Methods

TimeSourceクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ToString	このインスタンスを表すSystem.Stringを返します。

TimeSource.ToString Method
このインスタンスを表すSystem.Stringを返します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

```
Public override Function ToString() As string
```

戻り値
このインスタンスを表すSystem.String(形式 "<class-name> {Value = <value-as-string>}")。

Hbm.Api.Common.Entities.Values Namespace

これは名前空間Hbm.Api.Common.Entities.Valuesです。

クラス

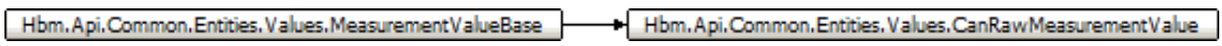
	名前	説明
	CanRawMeasurementValue	1つのCAN raw測定値、1つのtimestamp、および1つのstateで構成される単一のCanRawMessageを表します。
	CanRawMeasurementValues	ReSharperはInheritdocConsiderUsageを一度無効にします。
	CanRawMessage	CanRawメッセージを表します。
	MeasurementValueBase	1つの測定値、1つのtimestamp、および1つのstateで構成される単一のmeasurement valueを表します。
	MeasurementValuesBase	連続測定の測定値を保持するクラス。データ取得の実行中に測定値を読み取るには、`Hbm.API.Common.DaqMeasurement.FillMeasurementValues`を呼び出す必要があります。これにより、信号の循環バッファから信号のContinuousMeasurementValueに特定の数の測定値がコピーされます。
	MeasurementValuesCircularBuffer	ReSharperはInheritdocConsiderUsageを一度無効にします。
	MeasurementValuesCircularBufferBase	`Hbm.API.`Common.Entities.MeasurementValues をバッファする循環バッファを管理します。バッファがいっぱいになる前に値がフェッチされない場合、バッファ内の最も古い値が上書きされ、バッファから値が再びフェッチされるまでBufferOverflowOccurredがtrueに設定されます。

インタフェース

	名前	説明
	IMeasurementValuesBase	測定値の型不変部にアクセスするためのインタフェース(実際には、値は任意の型である可能性があるため、値自体を除くすべて)。

CanRawMeasurementValue Class

1つのCAN raw測定値、1つのtimestamp、および1つのstateで構成される単一のCanRawMessageを表します。



C++

```
class CanRawMeasurementValue : public MeasurementValueBase<CanRawMessage>;
```

C#

```
public class CanRawMeasurementValue : MeasurementValueBase<CanRawMessage>;
```

Visual Basic

```
Public Class CanRawMeasurementValue
Inherits MeasurementValueBase
```

ファイル

CanRawMeasurementValue.cs

CanRawMeasurementValue Members

次の表は、CanRawMeasurementValueによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CanRawMeasurementValue	

Public Properties

	名前	説明
	State	測定値の妥当性を示します。
	Timestamp	値のタイムスタンプ。
	Value	測定値。

CanRawMeasurementValue.CanRawMeasurementValue Constructor

C++

```
CanRawMeasurementValue(
    CanRawMessage value,
    double timeStamp,
    MeasurementValueState state
);
```

C#

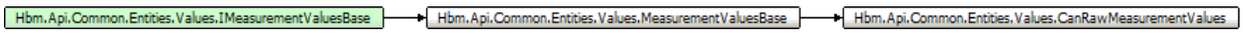
```
public CanRawMeasurementValue(
    CanRawMessage value,
    double timeStamp,
    MeasurementValueState state
);
```

Visual Basic

```
Public Sub New(
    value As CanRawMessage,
    timeStamp As double,
    state As MeasurementValueState
)
```

CanRawMeasurementValues Class

ReSharperはInheritdocConsiderUsageを一度無効にします。



C++

```
class CanRawMeasurementValues : public MeasurementValuesBase<CanRawMessage>;
```

C#

```
public class CanRawMeasurementValues : MeasurementValuesBase<CanRawMessage>;
```

Visual Basic

```
Public Class CanRawMeasurementValues
Inherits MeasurementValuesBase
```

ファイル

CanRawMeasurementValues.cs

CanRawMessage Class

CanRawメッセージを表します。



C++

```
class CanRawMessage;
```

C#

```
public class CanRawMessage;
```

Visual Basic

```
Public Class CanRawMessage
```

ファイル

CanRawMessage.cs

CanRawMessage Members

次の表は、CanRawMessageによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CanRawMessage	コンストラクタ。

Public Properties

	名前	説明
	Header	メッセージのヘッダ(IDの4バイト、DataLengthCodeの2バイトおよび未使用の2バイト = 合計8バイトで構成され、ulongとしてコード化されます)。
	Payload	byte[]として指定するメッセージのペイロード。サイズは64バイトに固定設定されているため、CAN FDによって配信される最大64バイトを含めることができます。標準CANの場合、8バイトのペイロードのみが埋められます。
	Timestamp	メッセージのタイムスタンプ。これは、デバイスから送信されたメッセージの絶対時間です。

CanRawMessage.CanRawMessage Constructor

コンストラクタ。

C++

```
CanRawMessage(
    double timestamp,
    double header,
    double[] payload
);
```

C#

```
public CanRawMessage(
    double timestamp,
    double header,
    double[] payload
);
```

Visual Basic

```
Public Sub New(
    timestamp As double,
    header As double,
    payload As double[]
)
```

引数

引数	説明
timestamp	メッセージのタイムスタンプ。
header	メッセージのヘッダ。
payload	メッセージのペイロード。

CanRawMessage Properties

CanRawMessageクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Header	メッセージのヘッダ(IDの4バイト、DataLengthCodeの2バイトおよび未使用の2バイト = 合計8バイトで構成され、ulongとしてコード化されます)。
	Payload	byte[]として指定するメッセージのペイロード。サイズは64バイトに固定設定されているため、CAN FDによって配信される最大64バイトを含めることができます。標準CANの場合、8バイトのペイロードのみが埋められます。
	Timestamp	メッセージのタイムスタンプ。これは、デバイスから送信されたメッセージの絶対時間です。

CanRawMessage.Header Property

メッセージのヘッダ(IDの4バイト、DataLengthCodeの2バイトおよび未使用の2バイト = 合計8バイトで構成され、ulongとしてコード化されます)。

C++

```
__property ulong Header;
```

C#

```
public ulong Header;
```

Visual Basic

```
Public Property Header() As ulong
```

CanRawMessage.Payload Property

byte[]として指定するメッセージのペイロード。サイズは64バイトに固定設定されているため、CAN FDによって配信される最大64バイトを含めることができます。標準CANの場合、8バイトのペイロードのみが埋められます。

C++

```
__property byte[] Payload;
```

C#

```
public byte[] Payload;
```

Visual Basic

```
Public Property Payload() As byte[]
```

CanRawMessage.Timestamp Property

メッセージのタイムスタンプ。これは、デバイスから送信されたメッセージの絶対時間です。

C++

```
__property double Timestamp;
```

C#

```
public double Timestamp;
```

Visual Basic

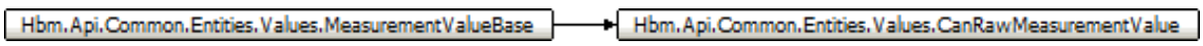
```
Public Property Timestamp() As double
```

備考

Signal.ContinuousMeasurementValues[].TimestampsやIGenericSignal.GenericMeasurementValues[].Timestampsと混合しないでください。これらは、測定の開始を基準とした相対的なタイムスタンプであり、通常は約0.0秒から始まります。

MeasurementValueBase Class

1つの測定値、1つのtimestamp、および1つのstateで構成される単一のmeasurement valueを表します。



C++

```
template <T>
class MeasurementValueBase;
```

C#

```
public abstract class MeasurementValueBase<T>;
```

Visual Basic

```
Public abstract Class MeasurementValueBase(Of T)
```

ファイル

MeasurementValueBase.cs

MeasurementValueBase Members

次の表は、MeasurementValueBaseによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	MeasurementValueBase	コンストラクタ。

Public Properties

	名前	説明
	State	測定値の妥当性を示します。
	Timestamp	値のタイムスタンプ。
	Value	測定値。

MeasurementValueBase.MeasurementValueBase Constructor

コンストラクタ。

C++

```
protected MeasurementValueBase(
    T value,
    double timestamp,
    MeasurementValueState state
);
```

C#

```
protected MeasurementValueBase(
    T value,
    double timestamp,
    MeasurementValueState state
);
```

Visual Basic

```
protected Sub New(
    value As T,
    timestamp As double,
    state As MeasurementValueState
)
```

引数

引数	説明
value	測定値。
timestamp	メッセージのタイムスタンプ。
state	測定値のステータス。

MeasurementValueBase Properties

MeasurementValueBaseクラスのプロパティの一覧です。

Public Properties

	名前	説明
	State	測定値の妥当性を示します。
	Timestamp	値のタイムスタンプ。
	Value	測定値。

MeasurementValueBase.State Property

測定値の妥当性を示します。

C++

```
__property MeasurementValueState State;
```

C#

```
public MeasurementValueState State;
```

Visual Basic

```
Public Property State() As MeasurementValueState
```

MeasurementValueBase.Timestamp Property

値のタイムスタンプ。

C++

```
__property double Timestamp;
```

C#

```
public double Timestamp;
```

Visual Basic

```
Public Property Timestamp() As double
```

MeasurementValueBase.Value Property

測定値。

C++

```
__property T Value;
```

C#

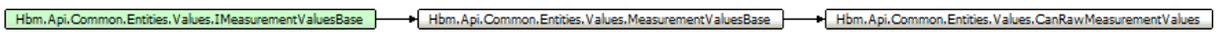
```
public T Value;
```

Visual Basic

```
Public Property Value() As T
```

MeasurementValuesBase Class

連続測定の測定値を保持するクラス。データ取得の実行中に測定値を読み取るには、Hbm.API.Common.DaqMeasurement.FillMeasurementValuesを呼び出す必要があります。これにより、信号の循環バッファから信号のContinuousMeasurementValueに特定の数の測定値がコピーされます。



C++

```
template <T>
class  MeasurementValuesBase : public IMeasurementValuesBase;
```

C#

```
public class MeasurementValuesBase<T> : IMeasurementValuesBase;
```

Visual Basic

```
Public Class MeasurementValuesBase(Of T)
Inherits IMeasurementValuesBase
```

ファイル

MeasurementValuesBase.cs

引数

引数	説明
T	値のデータ型。通常、これはすべての種類の信号に対して2倍になります。Hbm.Api.Common.Entities.Signals.CanRawSignal)の場合、値のデータ型はCanRawMessageです。

MeasurementValuesBase Members

次の表は、MeasurementValuesBaseによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ResetBuffers	内部バッファとステータスをリセットします。

Public Properties

	名前	説明
	BufferOverrunOccurred	基になる循環バッファがバッファオーバーランを登録した場合、True。これは、測定値の取得時間が長すぎる場合に発生します。基になる循環バッファ内の"最も古い"値は、最後のリクエスト以降に上書きされます。
	BufferSize	バッファに保存できる測定値、タイムスタンプ、および状態の最大数。
	IsFirstStatusOnlyRequested	trueの場合、States[0]のみを使用します。
	IsFirstTimeStampOnlyRequested	trueの場合、TimeStamps[0]のみを使用します。
	States	値の状態。
	Timestamps	タイムスタンプ。測定開始からの秒数(DaqMeasurement.MeasurementStartTime、DaqMeasurement.MeasurementStartUTCTime)。
	TotalValueCount	測定開始以降に取得した測定値の総数。
	UpdatedValueCount	最後のHbm.Api.Common.DaqMeasurement.FillMeasurementValues呼び出し以降の新しい測定値の数。
	Values	測定値。

MeasurementValuesBase Methods

MeasurementValuesBaseクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ResetBuffers	内部バッファとステータスをリセットします。

MeasurementValuesBase.ResetBuffers Method

内部バッファとステータスをリセットします。

C++

```
void ResetBuffers();
```

C#

```
public void ResetBuffers();
```

Visual Basic

```
Public Function ResetBuffers() As void
```

MeasurementValuesBase Properties

MeasurementValuesBaseクラスのプロパティの一覧です。

Public Properties

	名前	説明
	BufferOverrunOccurred	基になる循環バッファがバッファオーバーランを登録した場合、True。これは、測定値の取得時間が長すぎる場合に発生します。基になる循環バッファ内の"最も古い"値は、最後のリクエスト以降に上書きされます。
	BufferSize	バッファに保存できる測定値、タイムスタンプ、および状態の最大数。
	IsFirstStatusOnlyRequested	trueの場合、States[0]のみを使用します。
	IsFirstTimeStampOnlyRequested	trueの場合、TimeStamps[0]のみを使用します。
	States	値の状態。
	Timestamps	タイムスタンプ。測定開始からの秒数(DaqMeasurement.MeasurementStartTime、DaqMeasurement.MeasurementStartUTCTime)。
	TotalValueCount	測定開始以降に取得した測定値の総数。
	UpdatedValueCount	最後のHbm.Api.Common.DaqMeasurement.FillMeasurementValues呼び出し以降の新しい測定値の数。
	Values	測定値。

MeasurementValuesBase.BufferOverrunOccurred Property

基になる循環バッファがバッファオーバーランを登録した場合、True。これは、測定値の取得時間が長すぎる場合に発生します。基になる循環バッファ内の"最も古い"値は、最後のリクエスト以降に上書きされます。

C++

```
__property bool BufferOverrunOccurred;
```

C#

```
public bool BufferOverrunOccurred;
```

Visual Basic

```
Public Property BufferOverrunOccurred() As bool
```

MeasurementValuesBase.BufferSize Property

バッファに保存できる測定値、タイムスタンプ、および状態の最大数。

C++

```
__property int BufferSize;
```

C#

```
public int BufferSize;
```

Visual Basic

```
Public Property BufferSize() As Integer
```

MeasurementValuesBase.IsFirstStatusOnlyRequested Property

trueの場合、States[0]のみを使用します。

C++

```
__property bool IsFirstStatusOnlyRequested;
```

C#

```
public bool IsFirstStatusOnlyRequested;
```

Visual Basic

```
Public Property IsFirstStatusOnlyRequested() As bool
```

MeasurementValuesBase.IsFirstTimeStampOnlyRequested Property

trueの場合、TimeStamps[0]のみを使用します。

C++

```
__property bool IsFirstTimeStampOnlyRequested;
```

C#

```
public bool IsFirstTimeStampOnlyRequested;
```

Visual Basic

```
Public Property IsFirstTimeStampOnlyRequested() As bool
```

MeasurementValuesBase.States Property

値の状態。

C++

```
__property MeasurementValueState[] States;
```

C#

```
public MeasurementValueState[] States;
```

Visual Basic

```
Public Property States() As MeasurementValueState[]
```

MeasurementValuesBase.Timestamps Property

タイムスタンプ。測定開始からの秒数(DaqMeasurement.MeasurementStartTime、DaqMeasurement.MeasurementStartUTCTime)。

C++

```
__property double[] Timestamps;
```

C#

```
public double[] Timestamps;
```

Visual Basic

```
Public Property Timestamps() As double[]
```

MeasurementValuesBase.TotalValueCount Property

測定開始以降に取得した測定値の総数。

C++

```
__property long TotalValueCount;
```

C#

```
public long TotalValueCount;
```

Visual Basic

```
Public Property TotalValueCount() As long
```

MeasurementValuesBase.UpdatedValueCount Property

最後のHbm.Api.Common.DaqMeasurement.FillMeasurementValues呼び出し以降の新しい測定値の数。

C++

```
__property int UpdatedValueCount;
```

C#

```
public int UpdatedValueCount;
```

Visual Basic

```
Public Property UpdatedValueCount() As Integer
```

MeasurementValuesBase.Values Property

測定値。

C++

```
__property T[] Values;
```

C#

```
public T[] Values;
```

Visual Basic

```
Public Property Values() As T[]
```

MeasurementValuesCircularBuffer Class

ReSharperはInheritdocConsiderUsageを一度無効にします。



C++

```
class MeasurementValuesCircularBuffer : public MeasurementValuesCircularBufferBase<double>;
```

C#

```
internal class MeasurementValuesCircularBuffer : MeasurementValuesCircularBufferBase<double>;
```

Visual Basic

```
Friend Class MeasurementValuesCircularBuffer
Inherits MeasurementValuesCircularBufferBase
```

ファイル

MeasurementValuesCircularBuffer.cs

備考

ReSharperはInheritdocConsiderUsageを一度無効にします。

MeasurementValuesCircularBuffer Members

次の表は、MeasurementValuesCircularBufferによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	AddValues	循環バッファにnumberOfValuesToAddの"measurement value lines"を追加します。
	GetValues	循環バッファから、要求した数の測定値を取得します。
	IncrementReadPointer	(循環バッファの"wrap around"を考慮して)読み取りポインタを指定された増分だけインクリメントします。この関数は、DaqMeasurement.StartDaq()関数の実行中に使用します(すべての信号に対して同じ開始タイムスタンプを取得するため)。
	MeasurementValuesCircularBuffer	
	PeekValues	次に利用可能なMeasurementValuesを返しますが、それらは消費されません。したがって、要求された値は循環バッファから"削除"されません!
	Reset	循環バッファをリセットします。この関数の呼び出し後、NumberOfValuesInCircularBufferは0を返します。循環バッファのサイズは変更されません。
	SetStartTime	タイムスタンプの開始時刻を設定します。循環バッファに既に格納されているタイムスタンプは、この関数の実行中にこの時点まで減少します。

Internal Properties

	名前	説明
	BufferOverflowOccurred	GetValuesの最後の呼び出し以降にバッファオーバーフローが発生した場合はTrue(valuesAdded > bufferSize + valuesFetched の場合、バッファオーバーフローが発生します)。
R	IsValueAdded	この関数の前回の実行以降の値が循環バッファに追加されている場合はtrueを返します。これは、信号がまだ新しい測定データを受信しているかどうかを確認するために使用します(信号損失タイムアウトを検出するため)。
	NumberOfNewValuesSinceLastAdd	最後のデータフェッチ以降の新しい値の数を取得します。
R	NumberOfValuesInCircularBuffer	バッファ内のまだフェッチされていない値の数。
R	StartTime	測定の開始時刻。測定が開始されている場合、同期されたStartTimeは、測定に参加するすべての信号の(同期後の)最も早いタイムスタンプです。非同期で測定が開始されている場合、StartTimeは0です。

MeasurementValuesCircularBuffer.MeasurementValuesCircularBuffer Constructor

測定値の循環バッファ。

C++

```
MeasurementValuesCircularBuffer(
    int bufferSize
);
```

C#

```
internal MeasurementValuesCircularBuffer(
    int bufferSize
);
```

Visual Basic

```
Friend Sub New(
    bufferSize As Integer
)
```

MeasurementValuesCircularBufferBase Class

Hbm.API.Common.Entities.MeasurementValues をバッファする循環バッファを管理します。バッファがいっぱいになる前に値がフェッチされない場合、バッファ内の最も古い値が上書きされ、バッファから値が再びフェッチされるまでBufferOverflowOccurredがtrueに設定されます。



C++

```
template <T>
class  MeasurementValuesCircularBufferBase;
```

C#

```
internal class MeasurementValuesCircularBufferBase<T>;
```

Visual Basic

```
Friend Class MeasurementValuesCircularBufferBase(Of T)
```

ファイル

MeasurementValuesCircularBufferBase.cs

MeasurementValuesCircularBufferBase Members

次の表は、MeasurementValuesCircularBufferによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	AddValues	循環バッファにnumberOfValuesToAddの"measurement value lines"を追加します。
	GetValues	循環バッファから、要求した数の測定値を取得します。
	IncrementReadPointer	(循環バッファの"wrap around"を考慮して)読み取りポインタを指定された増分だけインクリメントします。この関数は、DaqMeasurement.StartDaq()関数の実行中に使用します(すべての信号に対して同じ開始タイムスタンプを取得するため)。
	MeasurementValuesCircularBuffer	測定値の循環バッファ。
	PeekValues	次に利用可能なMeasurementValuesを返しますが、それらは消費されません。したがって、要求された値は循環バッファから"削除"されません!
	Reset	循環バッファをリセットします。この関数の呼び出し後、NumberOfValuesInCircularBufferは0を返します。循環バッファのサイズは変更されません。
	SetStartTime	タイムスタンプの開始時刻を設定します。循環バッファに既に格納されているタイムスタンプは、この関数の実行中にこの時点まで減少します。

Internal Properties

	名前	説明
	BufferOverflowOccurred	GetValuesの最後の呼び出し以降にバッファオーバーフローが発生した場合はTrue(valuesAdded > bufferSize + valuesFetched の場合、バッファオーバーフローが発生します)。
R	IsValueAdded	この関数の前回の実行以降の値が循環バッファに追加されている場合はtrueを返します。これは、信号がまだ新しい測定データを受信しているかどうかを確認するために使用します(信号損失タイムアウトを検出するため)。
	NumberOfNewValuesSinceLastAdd	最後のデータフェッチ以降の新しい値の数を取得します。
R	NumberOfValuesInCircularBuffer	バッファ内のまだフェッチされていない値の数。
R	StartTime	測定の開始時刻。測定が開始されている場合、同期されたStartTimeは、測定に参加するすべての信号の(同期後の)最も早いタイムスタンプです。非同期で測定が開始されている場合、StartTimeは0です。

MeasurementValuesCircularBufferBase.MeasurementValuesCircularBufferBase Constructor

測定値の循環バッファ。

C++

```
MeasurementValuesCircularBufferBase(
    int bufferSize
);
```

C#

```
internal MeasurementValuesCircularBufferBase(
    int bufferSize
);
```

Visual Basic

```
Friend Sub New(
    bufferSize As Integer
)
```

引数

引数	説明
bufferSize	循環バッファのサイズ。

MeasurementValuesCircularBufferBase Methods

MeasurementValuesCircularBufferBaseクラスのメソッドの一覧です。

Internal Methods

	名前	説明
	AddValues	循環バッファにnumberOfValuesToAddの"measurement value lines"を追加します。
	GetValues	循環バッファから、要求した数の測定値を取得します。
	IncrementReadPointer	(循環バッファの"wrap around"を考慮して)読み取りポインタを指定された増分だけインクリメントします。この関数は、DaqMeasurement.StartDaq()関数の実行中に使用します(すべての信号に対して同じ開始タイムスタンプを取得するため)。
	MeasurementValuesCircularBuffer	測定値の循環バッファ。
	PeekValues	次に利用可能なMeasurementValuesを返しますが、それらは消費されません。したがって、要求された値は循環バッファから"削除"されません!
	Reset	循環バッファをリセットします。この関数の呼び出し後、NumberOfValuesInCircularBufferは0を返します。循環バッファのサイズは変更されません。
	SetStartTime	タイムスタンプの開始時刻を設定します。循環バッファに既に格納されているタイムスタンプは、この関数の実行中にこの時点まで減少します。

MeasurementValuesCircularBufferBase.AddValues Method

循環バッファにnumberOfValuesToAddの"measurement value lines"を追加します。

C++

```
void AddValues(
    T[] measurementValues,
    double[] timeStampValues,
    MeasurementValueState[] statusValues,
    int numberOfValuesToAdd
);
```

C#

```
internal void AddValues(
    T[] measurementValues,
    double[] timeStampValues,
    MeasurementValueState[] statusValues,
    int numberOfValuesToAdd
);
```

Visual Basic

```
Friend Function AddValues(
    measurementValues As T[],
    timeStampValues As double[],
    statusValues As MeasurementValueState[],
    numberOfValuesToAdd As Integer
) As void
```

引数

引数	説明
measurementValues	測定値。
timeStampValues	タイムスタンプ。
statusValues	測定値の状態。
numberOfValuesToAdd	追加する値の数。

例外

例外	説明
System.ArgumentException	numberOfValuesToAddが0未満の場合にスローされます。 measurementValues、timeStampValues、またはstatusValuesのサイズが numberOfValuesToAddより小さい場合にスローされます。
System.ArgumentNullException	MeasurementValues、timeStampValues、またはstatusValuesがnullの場合に スローされます。

MeasurementValuesCircularBufferBase.GetValues Method

循環バッファから、要求した数の測定値を取得します。

C++

```
void GetValues(
    int numberOfValuesToGet,
    MeasurementValuesBase<T> measurementValues
);
```

C#

```
internal void GetValues(
    int numberOfValuesToGet,
    MeasurementValuesBase<T> measurementValues
);
```

Visual Basic

```
Friend Function GetValues(
    numberOfValuesToGet As Integer,
    measurementValues As MeasurementValuesBase<T>
) As void
```

引数

引数	説明
numberOfValuesToGet	循環バッファから取得する測定値の数。
measurementValues	循環バッファ値のコピー先となるHbm.API.Common.Entities.MeasurementValuesオブジェクト。

備考

現在のバッファサイズが小さすぎて要求された値をすべて取得できない場合、Hbm.API.Common.Entities.MeasurementValuesのサイズが拡張されます。numberOfValuesToGetが循環バッファ内の使用可能な値の数より大きい場合、更新された値の数が使用可能な値の数に設定されます!

例外

例外	説明
System.ArgumentException	numberOfValuesToAddが0未満の場合にスローされます。
System.ArgumentNullException	測定値がnullの場合にスローされます。

MeasurementValuesCircularBufferBase.IncrementReadPointer Method

(循環バッファの"wrap around"を考慮して)読み取りポインタを指定された増分だけインクリメントします。この関数は、DaqMeasurement.StartDaq()関数の実行中に使用します(すべての信号に対して同じ開始タイムスタンプを取得するため)。

C++

```
void IncrementReadPointer(
    int increment
);
```

C#

```
internal void IncrementReadPointer(
    int increment
);
```

Visual Basic

```
Friend Function IncrementReadPointer(
    increment As Integer
) As void
```

引数

引数	説明
increment	インクリメントする位置の数。

MeasurementValuesCircularBufferBase.PeekValues Method

次に利用可能なMeasurementValueを返しますが、それらは消費されません。したがって、要求された値は循環バッファから"削除"されません!

C++

```
void PeekValues(,
    int numberOfValuesToPeek,
    MeasurementValuesBase<T> measurementValues
);
```

C#

```
internal void PeekValues(
    int numberOfValuesToPeek,
    MeasurementValuesBase<T> measurementValues
);
```

Visual Basic

```
Friend Function PeekValues(
    numberOfValuesToPeek As Integer
    measurementValues As Integer MeasurementValuesBase<T>
) As void
```

引数

引数	説明
numberOfValuesToPeek	測定値のピーク。
measurementValues	既存のMeasurementValuesオブジェクトへの参照。

例外

例外	説明
System.ArgumentException	numberOfValuesToPeekが0未満の場合にスローされます。
System.ArgumentNullException	measurementValuesがnullの場合にスローされます。

MeasurementValuesCircularBufferBase.Reset Method

循環バッファをリセットします。この関数の呼び出し後、NumberOfValuesInCircularBufferは0を返します。循環バッファのサイズは変更されません。

C++

```
void Reset();
```

C#

```
internal void Reset();
```

Visual Basic

```
Friend Function Reset() As void
```

引数

引数	説明
increment	インクリメントする位置の数。

MeasurementValuesCircularBufferBase.SetStartTime Method

タイムスタンプの開始時刻を設定します。循環バッファに既に格納されているタイムスタンプは、この関数の実行中にこの時点まで減少します。

C++

```
void SetStartTime(
    double startTime
);
```

C#

```
internal void SetStartTime(
    double startTime
);
```

Visual Basic

```
Friend Function SetStartTime(
    startTime As double
) As void
```

引数

引数	説明
startTime	循環バッファ内のすべてのタイムスタンプから減算される、1.1.2000からの秒単位の時間。 AddValuesを介して循環バッファに値を追加する場合、この開始時刻は追加する値のタイムスタンプから減算されます。

MeasurementValuesCircularBufferBase Properties

MeasurementValuesCircularBufferBaseクラスのプロパティの一覧です。

Internal Properties

	名前	説明
	BufferOverflowOccurred	GetValuesの最後の呼び出し以降にバッファオーバーフローが発生した場合はTrue(valuesAdded > bufferSize + valuesFetched の場合、バッファオーバーフローが発生します)。
R	IsValueAdded	この関数の前回の実行以降の値が循環バッファに追加されている場合はtrueを返します。これは、信号がまだ新しい測定データを受信しているかどうかを確認するために使用します(信号損失タイムアウトを検出するため)。
	NumberOfNewValuesSinceLastAdd	最後のデータフェッチ以降の新しい値の数を取得します。
R	NumberOfValuesInCircularBuffer	バッファ内のまだフェッチされていない値の数。
R	StartTime	測定の開始時刻。測定が開始されている場合、同期されたStartTimeは、測定に参加するすべての信号の(同期後の)最も早いタイムスタンプです。非同期で測定が開始されている場合、StartTimeは0です。

MeasurementValuesCircularBufferBase.BufferOverflowOccurred Property

GetValuesの最後の呼び出し以降にバッファオーバーフローが発生した場合はTrue(valuesAdded > bufferSize + valuesFetched の場合、バッファオーバーフローが発生します)。

C++

```
__property bool BufferOverflowOccurred;
```

C#

```
internal bool BufferOverflowOccurred;
```

Visual Basic

```
Friend Property BufferOverflowOccurred() As bool
```

MeasurementValuesCircularBufferBase.IsValueAdded Property

この関数の前回の実行以降の値が循環バッファに追加されている場合はtrueを返します。これは、信号がまだ新しい測定データを受信しているかどうかを確認するために使用します(信号損失タイムアウトを検出するため)。

C++

```
__property bool IsValueAdded;
```

C#

```
internal bool IsValueAdded;
```

Visual Basic

```
Friend Property IsValueAdded() As bool
```

MeasurementValuesCircularBufferBase.NumberOfNewValuesSinceLastAdd Property

最後のデータフェッチ以降の新しい値の数を取得します。

C++

```
__property long NumberOfNewValuesSinceLastAdd;
```

C#

```
internal long NumberOfNewValuesSinceLastAdd;
```

Visual Basic

```
Friend Property NumberOfNewValuesSinceLastAdd() As long
```

MeasurementValuesCircularBufferBase.NumberOfValuesInCircularBuffer Property

バッファ内のまだフェッチされていない値の数。

C++

```
__property int NumberOfValuesInCircularBuffer;
```

C#

```
internal int NumberOfValuesInCircularBuffer;
```

Visual Basic

```
Friend Property NumberOfValuesInCircularBuffer() As Integer
```

MeasurementValuesCircularBufferBase.StartTime Property

測定の開始時刻。測定が開始されている場合、同期されたStartTimeは、測定に参加するすべての信号の(同期後の)最も早いタイムスタンプです。非同期で測定が開始されている場合、StartTimeは0です。

C++

```
__property double StartTime;
```

C#

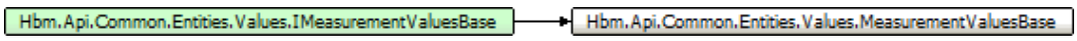
```
internal double StartTime;
```

Visual Basic

```
Friend Property StartTime() As double
```

IMeasurementValuesBase Interface

測定値の型不変部にアクセスするためのインタフェース(実際には、値は任意の型である可能性があるため、値自体を除くすべて)。



C++

```
__interface IMeasurementValuesBase;
```

C#

```
public interface IMeasurementValuesBase;
```

Visual Basic

```
Public Interface IMeasurementValuesBase
```

ファイル

IMeasurementValuesBase.cs

IMeasurementValuesBase Members

次の表は、IMeasurementValuesBaseによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ResetBuffers	内部バッファとステータスをリセットします。

Public Properties

	名前	説明
	BufferOverrunOccurred	基になる循環バッファがバッファオーバーランを登録した場合、True。これは、測定値の取得時間が長すぎる場合に発生します。基になる循環バッファ内の"最も古い"値は、最後のリクエスト以降に上書きされます。
	BufferSize	バッファに保存できる測定値、タイムスタンプ、および状態の最大数。
	IsFirstStatusOnlyRequested	trueの場合、States[0]のみを使用します。
	IsFirstTimeStampOnlyRequested	trueの場合、TimeStamps[0]のみを使用します。
	States	値の状態。
	Timestamps	タイムスタンプ。測定開始からの秒数 (DaqMeasurement.MeasurementStartTime、DaqMeasurement.MeasurementStartUTCTime)。
	TotalValueCount	測定開始以降に取得した測定値の総数。
	UpdatedValueCount	最後のHbm.Api.Common.DaqMeasurement.FillMeasurementValues呼び出し以降の新しい測定値の数。

IMeasurementValuesBase Methods

IMeasurementValuesBaseクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ResetBuffers	内部バッファとステータスをリセットします。

IMeasurementValuesBase.ResetBuffers Method

内部バッファとステータスをリセットします。

C++

```
void ResetBuffers();
```

C#

```
void ResetBuffers();
```

Visual Basic

```
Function ResetBuffers() As void
```

IMeasurementValuesBase Properties

IMeasurementValuesBaseクラスのプロパティの一覧です。

Public Properties

	名前	説明
	BufferOverrunOccurred	基になる循環バッファがバッファオーバーランを登録した場合、True。これは、測定値の取得時間が長すぎる場合に発生します。基になる循環バッファ内の"最も古い"値は、最後のリクエスト以降に上書きされます。
	BufferSize	バッファに保存できる測定値、タイムスタンプ、および状態の最大数。
	IsFirstStatusOnlyRequested	trueの場合、States[0]のみを使用します。
	IsFirstTimeStampOnlyRequested	trueの場合、TimeStamps[0]のみを使用します。
	States	値の状態。
	Timestamps	タイムスタンプ。測定開始からの秒数 (DaqMeasurement.MeasurementStartTime、DaqMeasurement.MeasurementStartUTCTime)。
	TotalValueCount	測定開始以降に取得した測定値の総数。
	UpdatedValueCount	最後のHbm.Api.Common.DaqMeasurement.FillMeasurementValues呼び出し以降の新しい測定値の数。

IMeasurementValuesBase.BufferOverrunOccurred Property

基になる循環バッファがバッファオーバーランを登録した場合、True。これは、測定値の取得時間が長すぎる場合に発生します。基になる循環バッファ内の"最も古い"値は、最後のリクエスト以降に上書きされます。

C++

```
__property bool BufferOverrunOccurred;
```

C#

```
bool BufferOverrunOccurred;
```

Visual Basic

```
Property BufferOverrunOccurred() As bool
```

IMeasurementValuesBase.BufferSize Property

バッファに保存できる測定値、タイムスタンプ、および状態の最大数。

C++

```
__property int BufferSize;
```

C#

```
int BufferSize;
```

Visual Basic

```
Property BufferSize() As Integer
```

IMeasurementValuesBase.IsFirstStatusOnlyRequested Property

trueの場合、States[0]のみを使用します。

C++

```
__property bool IsFirstStatusOnlyRequested;
```

C#

```
bool IsFirstStatusOnlyRequested;
```

Visual Basic

```
Property IsFirstStatusOnlyRequested() As bool
```

IMeasurementValuesBase.IsFirstTimeStampOnlyRequested Property

trueの場合、TimeStamps[0]のみを使用します。

C++

```
__property bool IsFirstTimeStampOnlyRequested;
```

C#

```
bool IsFirstTimeStampOnlyRequested;
```

Visual Basic

```
Property IsFirstTimeStampOnlyRequested() As bool
```

IMeasurementValuesBase.States Property

値の状態。

C++

```
__property MeasurementValueState[] States;
```

C#

```
MeasurementValueState[] States;
```

Visual Basic

```
Property States() As MeasurementValueState[]
```

IMeasurementValuesBase.Timestamps Property

タイムスタンプ。測定開始からの秒数 (DaqMeasurement.MeasurementStartTime、DaqMeasurement.MeasurementStartUTCTime)。

C++

```
__property double[] Timestamps;
```

C#

```
double[] Timestamps;
```

Visual Basic

```
Property Timestamps() As double[]
```

IMeasurementValuesBase.TotalValueCount Property

測定開始以降に取得した測定値の総数。

C++

```
__property long TotalValueCount;
```

C#

```
long TotalValueCount;
```

Visual Basic

```
Property TotalValueCount() As long
```

IMeasurementValuesBase.UpdatedValueCount Property

最後のHbm.Api.Common.DaqMeasurement.FillMeasurementValues呼び出し以降の新しい測定値の数

C++

```
__property int UpdatedValueCount;
```

C#

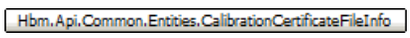
```
int UpdatedValueCount;
```

Visual Basic

```
Property UpdatedValueCount() As Integer
```

CalibrationCertificateFileInfo Class

校正証明に関する情報を格納するクラス。



C++

```
class CalibrationCertificateFileInfo;
```

C#

```
public class CalibrationCertificateFileInfo;
```

Visual Basic

```
Public Class CalibrationCertificateFileInfo
```

ファイル

CalibrationCertificateFileInfo.cs

CalibrationCertificateFileInfo Members

次の表は、CalibrationCertificateFileInfoによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	Date	校正証明書が作成された日時。
	DeviceUniqueId	デバイスの一意のID。
	Directory	校正証明書のディレクトリ。
	FileName	校正証明書のファイル名。
R	FullPath	校正証明書のフルパス(ディレクトリとファイル名)。

CalibrationCertificateFileInfo Properties

IMeasurementValuesBaseクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Date	校正証明書が作成された日時。
	DeviceUniqueId	デバイスの一意のID。
	Directory	校正証明書のディレクトリ。
	FileName	校正証明書のファイル名。
R	FullPath	校正証明書のフルパス(ディレクトリとファイル名)。

CalibrationCertificateFileInfo.Date Property

校正証明書が作成された日時。

C++

```
__property DateTime Date;
```

C#

```
public DateTime Date;
```

Visual Basic

```
Public Property Date() As DateTime
```

CalibrationCertificateFileInfo.DeviceUniqueId Property

デバイスの一意のID。

C++

```
__property string DeviceUniqueId;
```

C#

```
public string  DeviceUniqueId;
```

Visual Basic

```
Public Property DeviceUniqueId() As string
```

CalibrationCertificateFileInfo.Directory Property

校正証明書のディレクトリ。

C++

```
__property string Directory;
```

C#

```
public string  Directory;
```

Visual Basic

```
Public Property Directory() As string
```

CalibrationCertificateFileInfo.FileName Property

校正証明書のファイル名。

C++

```
__property string FileName;
```

C#

```
public string  FileName;
```

Visual Basic

```
Public Property FileName() As string
```

CalibrationCertificateFileInfo.FullPath Property

校正証明書のフルパス(ディレクトリとファイル名)。

C++

```
__property string FullPath;
```

C#

```
public string  FullPath;
```

Visual Basic

```
Public Property FullPath() As string
```

Device Class

単一のデバイスを表します。



C++

```
class Device;
```

C#

```
public abstract class Device;
```

Visual Basic

```
Public abstract Class Device
```

ファイル

Device.cs

備考

通常、デバイスオブジェクトのプロパティは、デバイスの接続後に設定します。デバイスをスキャンすると、部分的に設定されたプロパティを持つデバイスのリストを返します。スキャンメカニズムは次のプロパティを入力します: ConnectionInfo、FamilyName、FirmwareVersion、Model、Name、SerialNo 。

Device Members

次の表は、Deviceによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	GetAndBroadcastDeviceStatusList	デバイスのステータスリストを取得し、サブスクリプションをフィルタリングして、変更された状態をブロードキャストします。

Protected internal Methods

	名前	説明
	StoreStatusIfChanged	ステータスがリストに保存されていない場合は保存し、変更された場合は置き換えます。

Protected Methods

	名前	説明
	BroadcastChannelStatusChanges	ステータスの変化をブロードキャストします。
	BroadcastTedsStatus	tedステータスを処理します。
	MakeChecks	複数のメソッド間で使用するいくつかのプロパティのチェックを集中管理するヘルパメソッド。
	SetAnalogOutOutputScaling	アナログ出力信号のデフォルトの出力テーブルのスケーリングを設定します。

Public Methods

	名前	説明
	ActivateTeds	指定したコネクタのTEDS設定をアクティブにし、新しいチャネルパラメータをロードします。
	AssignAutoCalibrationMode	指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignChannel	指定したチャネルの設定(センサ、信号などを含む)をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignConnector	指定したコネクタの設定(チャネル、センサ信号などを含む)をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSensor	指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSignal	指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignTimeSource	TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignZero	指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AutoDetectFbgBands	指定したFbgConnectorに対する帯域検出の結果を返します。
	ChangeIp	デバイスのipAddressを変更します。
	ClearZeroBalance	これは、ClearZeroBalance メソッドの多重定義の概要です。
	Device	コンストラクタ。デフォルト値を初期化します。

	名前	説明
	DisableRouting	ルーティング機能を無効にします。
	EnableRouting	ルーティング機能を有効にします。
	Find	指定した接続情報を使用してデバイスを検索します(実際には、コネクタ情報を読み取ることなくデバイス情報を読み取ります)。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	FindChannel	これは、FindChannelメソッドの多重定義の概要です。
	FindConnector	これは、FindConnectorメソッドの多重定義の概要です。
	FindSignal	指定したフィルタが配置されている信号を返します。
	FlashLed	これは、FlashLedメソッドの多重定義の概要です。
	GetAllChannels	デバイスのすべてのチャネルを取得します。
	GetAllSignals	デバイスのすべての信号を取得します。
	GetAmplifierSerialNumber	チャネルのアンプのシリアル番号を取得します。
	GetAmplifierType	チャネルのアンプタイプを取得します。
	GetBestFittingAutoFilter	信号のサンプルレートに応じて最適な自動フィルタを取得します。
	GetCalibrationCertificateFileInfos	要求したリリースの情報に応じて、校正証明ファイル情報のリストを返します。校正証明書を作成し、指定した出力ディレクトリにダウンロードします。
	GetChannelHardwareInfo	チャネルのハードウェア情報(アンプの種類など)を取得します。
	GetDeviceErrorStatusList	これは、GetDeviceErrorStatusListメソッドの多重定義の概要です。
	GetDeviceStatusList	これは、GetDeviceStatusListメソッドの多重定義の概要です。
	GetPossibleDataAcquisitionModes	デバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブル経由で確立)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedを含みます。デバイスが時間ベースの同期(PTPまたはNTPなど)を使用している場合、返されるリストにはDataAcquisitionMode.TimestampSynchronizedモードを含みます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはDataAcquisitionMode.Unsynchronizedモードを含む可能性が高くなります。デバイスが接続されていない場合、使用可能なデータ収集モードのリストは空になります。
	GetShunt	これは、GetShuntメソッドの多重定義の概要です。
	GetSupportedCanBitRates	サポートしているCANのビットレートを取得します。
	GetSupportedFilterFrequencies	指定したフィルタのサポートされているフィルタ周波数のリストを取得します。
	GetSupportedFilters	指定した信号のサポートされているフィルタのリストを取得します。
	GetSupportedJackType	指定した信号でサポートしているジャックの種類を取得します。
	GetSupportedSampleRates	サポートしているサンプルレートのリストを取得します。
	GetSupportedScalingTypes	指定したチャネルでサポートしているスケーリングタイプのリストを取得します。
	GetSupportedSensorTypes	指定したチャネルでサポートしているセンサタイプのリストを取得します。
	GetTimeSourceQuality	オフセット(同期タイムソースとデバイスの時間の間のタイムスパン)をミリ秒単位で返し、同期の品質に関する詳細情報を含むTimeSourceに依存する文字列を返します。NtpTimeSourceの場合、この文字列にはパラメータ(カンマ区切り)が含まれます: "remote、refid、st、t、when、pol、reach、delay、offset、jitter"。PtpTimeSourceの場合、この文字列には次のパラメータ (カンマ区切り)が含まれます: "GrandmasterID、SyncMode"、TimeScaleFlags、UtcOffset、UtcOffsetused、MasterOffset"
	GetUniqueID	すべてのデバイスファミリの中からデバイスを識別する一意の文字列を返します。
	GetUnit	指定したチャネルの単位を返します。
	IsChannelPartOfDevice	指定したチャネルがデバイスに属しているかどうかをチェックします。

	名前	説明
	IsConnectorPartOfDevice	指定したコネクタがデバイスに属しているかどうかをチェックします。
	IsFirmwareVersionSupported	指定したファームウェアのバージョンをサポートしているかどうかを判断します。
	IsSignalPartOfDevice	指定した信号がデバイスに属しているかどうかをチェックします。
	ReadDigitalIO	デジタル信号の現在の測定値を返します。
	ReadFbgSpectrum	指定したスペクトルタイプとコネクタに従ってFbgSpectrumを返します。この機能は、ファイバブラッググレーティング関連デバイスでのみサポートしいます。
	ReadSingleElectricalValue	指定したAnalogInSignalの現在の電気値を返します。
	ReadSingleMeasurementValue	DAQセッションを完全に初期化せずに、指定したSignalごとの単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadSingleMeasurementValueOfAllSignals	DAQセッションを完全に初期化せずに、デバイスのすべてのSignalに対して単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadTeds	指定したコネクタに接続されているTEDSからセンサを読み取ります。
	RemoveInputSignalReference	指定したチャネルから入力信号のリファレンスを削除します。
	Reset	デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数の呼び出し後に、デバイスは切断されます。これが必要なのはデバイスのパラメータ設定が変更され、デバイスがデバイスの関連オブジェクトモデルと一致しなくなったとき、工場出荷時のデフォルトをロードします(この関数を実装するときは、必ずデバイスを切断し、Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知します。
	ResetStatusMonitoringStates	ステータス監視状態をリセットして、次回のステータスポーリングですべてのイベントが再度発生できるようにします。
	SetAnalogOut	アナログ出力信号を指定した値に設定します。
	SetAnalogOutChannelActivation	提供されたチャネルのリストをアクティブまたは非アクティブにします。AnalogOutChannelのみをアクティブ化または非アクティブ化できます。
	SetCanBusParameters	指定したCanConnectorのすべてのCAN busパラメータ(CanConnector.BitRate、CanConnector.IsTerminated、CanConnector.SupplyVoltageなど)を割り当てますが、チャネルは割り当てません。
	SetCanChannelActivation	提供されたチャネルのリストをアクティブまたは非アクティブにします。CanInChannelとCanOutChannelのみをアクティブ化または非アクティブ化できます。
	SetChannelName	指定したチャネルにチャネル名を割り当てます。
	SetDataRateDomain	現在のモジュールに指定したdataRateDomainを設定します。重要:デバイスの種類によっては、再起動が自動的に行われます。この場合、デバイスはしばらくの間アクセスできなくなります。
	SetDeviceName	基になる物理デバイスの名前を設定します。派生クラスでオーバーライドする必要があります。
	SetDigitalIO	指定した値に従ってデジタル信号を設定します。

	名前	説明
	SetFbgBandsDetectionResult	AutoDetectFbgBands経由で事前に取得した、指定されたFbgBandsDetectionResultに従って、指定されたFbgConnectorのFbgChannelのFBG関連プロパティ(DetectionRange、FbgSensor.ReferenceWavelength、FbgChannel.IsActivatedなど)を設定します。
	SetFbgChannelActivation	指定したチャネルのリストをアクティブまたは非アクティブにします。FbgChannelのみをアクティブ化または非アクティブ化できます。
	SetFbgChannelDetectionRange	指定した値に従って、指定されたFbgChannelのFbgChannel.DetectionRangeを設定します。
	SetFbgDetectionThreshold	指定した値に従って、指定したFbgConnectorのFbgConnector.Detection Thresholdを設定します。
	SetInputSignalReference	入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。
	SetShunt	これは、SetShuntメソッドの多重定義の概要です。
	SetSignalName	指定した信号に信号名を割り当てます。
	SetTedsUsageMode	TEDS使用モードを設定し、コネクタ設定を再読み込みします。
	SetUnit	指定したチャネルの単位を割り当てます。
	SetZeroBalance	これは、SetZeroBalanceメソッドの多重定義の概要です。
	SubscribeToStatusMonitoring	指定したチャネルをステータス監視に登録します。
	UnsubscribeFromStatusMonitoring	指定したチャネルをステータス監視から登録解除します。
	WriteTeds	指定したコネクタに接続されたTEDSに指定したセンサを書き込みます。

Public Properties

	名前	説明
	CalCertSyncLock	基礎となるデバイスの校正証明書の伝達への並列アクセスを同期するために使用します。
	ConnectionInfo	物理デバイスへの接続方法に関する情報。
	Connectors	すべてのコネクタのリスト。
	DataAcquisitionMode	測定に使用するデータ収集モードを保持します。このモードは通常、DaqMeasurement.StartDaq(Enums.DataAcquisitionMode, int)中に呼び出されるDeviceFamily.PrepareDataAcquisition(Enums.DataAcquisitionMode,List{Device})によって設定されます。特にEnums.DataAcquisitionMode.Autoの場合、このプロパティは、測定の開始後に選択したデータ取得モードを返します。このプロパティを手動で設定しないでください! 内部使用のみを目的としています。
	DataRateDomain	データレートドメインを取得または設定します。
	FamilyName	このデバイスが属するファミリの名前。
	FbgFiberInfo	ファイバセンシングベースのデバイスの帯域情報を保持します。これは、ファイバセンシングベースのデバイスの場合にのみ入力します。
	FirmwareVersion	ファームウェアのバージョン。
	HardwareVersion	ハードウェアのバージョン。
	HasAdditionalTedsFieldsSupport	デバイスファミリがキャリブレーションおよびシリアル番号のTEDSフィールドをサポートしているかどうかを示す値を取得します。
	IsAnalogOutChannelDisablingSupported	デバイスがアナログ出力チャネルの無効化をサポートしているかどうかを示す値を取得または設定します。
	IsAutomaticTedsMonitoringSupported	このインスタンスが自動TEDS監視をサポートしているかどうかを示す値を取得します。
	IsCanBusSupplyVoltageSupported	デバイスがCAN busの供給電圧をサポートしているかどうかを示す値を取得または設定します。
	IsCanBusTerminationSupported	デバイスがCAN busターミネーションをサポートしているかどうかを示す値を取得または設定します。

	名前	説明
	IsConnectable	デバイスが接続可能かどうかを示します。
	IsConnected	デバイスが接続されているかどうかを示します。
R V	IsElectricalSnapshotSupported	ReadSingleElectricalValue(AnalogInSignal)関数を使用して、このインスタンスが電気値の要求をサポートしているかどうかを示す値を取得します。
	IsEnablingRoutingSupported	デバイスがルーティング機能の有効化/無効化をサポートしているかどうかを示す値を取得または設定します。
V	IsFirmwareUpdateSupported	このインスタンスがファームウェアの更新をサポートしているかどうかを示す値を取得または設定します。
	IsIpChangeSupported	デバイスがIPアドレスの変更をサポートしているかどうかを示す値を取得または設定します。
	IsLedFlashSupported	デバイスがLEDの点滅をサポートしているかどうかを示す値を取得または設定します。
	IsReadingDaqValues	デバイスが測定値を更新中かどうかを示します。
R V	IsReadingFbgSpectrumSupported	このインスタンスがスペクトルの読み取りをサポートしているかどうかを示す値を取得します。
	IsRoutingEnabled	デバイスのルーティング機能が現在有効になっているかどうかを示す値を取得または設定します。
	IsUnsupportedModel	特定のモデルがAPIでサポートされているかどうかを示します。
	IsUsingHbmClassicSampleRates	デバイスが10進数のサンプル レート(10Hz、100Hz、1000Hzなど)ではなく、従来のHBMサンプル レート(1200Hz、2400Hz、4800Hz)など) を使用しているかどうかを示します。このプロパティはデバイスの接続後に有効になります。現在、QuantumXデバイスファミリのみが10進数のサンプルレートをサポートしています。
	MinimumFirmwareVersionSupported	ドライバがサポートする最小ファームウェア バージョン。
	Model	デバイスモデルのタイプ。
R	MonitoredStatusTypes	監視対象のステータスの種類を取得します。
	Name	デバイス名。
R	NtpStartDate	秒数のカウントの開始日を協定世界時(UTC)で決定する日付。タイムスタンプは、この日付/時刻からの秒数で表されます。
S	Overflow	無効な測定値(オーバーフロー)の代わりに使用する値を定義します。ステータスが無効な各測定値については、対応する測定値がこのオーバーフロー値に設定されます。DaqEnvironment.Overflowを使用してこの値を設定します(デフォルトは1000000)。
	ProtectionType	デバイスの保護等級タイプ(例: IP55)。
	SerialNo	デバイスのシリアル番号。
	SupportedDataRateDomains	サポートしているデータレートドメインを取得または設定します。
	SyncLock	基礎となるデバイスへの並列アクセスを同期するために使用します。
	SyncMode	測定開始時のデバイスの役割を定義するモード。
	Timeout	デバイスとの通信のタイムアウト。
	TimeSource	物理デバイスの時刻を同期する方法に関する情報。

Device.Device Constructor

コンストラクタ。デフォルト値を初期化します。

C++

```
Device();
```

C#

```
public Device();
```

Visual Basic

```
Public Sub New()
```

Device Methods

Deviceクラスのメソッドの一覧です。

Internal Methods

	名前	説明
	GetAndBroadcastDeviceStatusList	デバイスのステータスリストを取得し、サブスクリプションをフィルタリングして、変更された状態をブロードキャストします。

Protected internal Methods

	名前	説明
	StoreStatusIfChanged	ステータスがリストに保存されていない場合は保存し、変更された場合は置き換えます。

Protected Methods

	名前	説明
	BroadcastChannelStatusChanges	ステータスの変化をブロードキャストします。
	BroadcastTedsStatus	tedステータスを処理します。
	MakeChecks	複数のメソッド間で使用するいくつかのプロパティのチェックを集中管理するヘルパメソッド。
	SetAnalogOutOutputScaling	アナログ出力信号のデフォルトの出力テーブルのスケーリングを設定します。

Public Methods

	名前	説明
A	ActivateTeds	指定したコネクタのTEDS設定をアクティブにし、新しいチャネルパラメータをロードします。
A	AssignAutoCalibrationMode	指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
A	AssignChannel	指定したチャネルの設定(センサ、信号などを含む)をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
A	AssignConnector	指定したコネクタの設定(チャネル、センサ信号などを含む)をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
A	AssignSensor	指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
A	AssignSignal	指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
A	AssignTimeSource	TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
A	AssignZero	指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
V	AutoDetectFbgBands	指定したFbgConnectorに対する帯域検出の結果を返します。
A	ChangeIp	デバイスのipAddressを変更します。
	ClearZeroBalance	これは、ClearZeroBalance メソッドの多重定義の概要です。
V	DisableRouting	ルーティング機能を無効にします。
V	EnableRouting	ルーティング機能を有効にします。

	名前	説明
	Find	指定した接続情報を使用してデバイスを検索します(実際には、コネクタ情報を読み取ることなくデバイス情報を読み取ります)。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	FindChannel	これは、FindChannelメソッドの多重定義の概要です。
	FindConnector	これは、FindConnectorメソッドの多重定義の概要です。
	FindSignal	指定したフィルタが配置されている信号を返します。
	FlashLed	これは、FlashLedメソッドの多重定義の概要です。
	GetAllChannels	デバイスのすべてのチャネルを取得します。
	GetAllSignals	デバイスのすべての信号を取得します。
	GetAmplifierSerialNumber	チャネルのアンプのシリアル番号を取得します。
	GetAmplifierType	チャネルのアンプタイプを取得します。
	GetBestFittingAutoFilter	信号のサンプルレートに応じて最適な自動フィルタを取得します。
	GetCalibrationCertificateFileInfos	要求したリリースの情報に応じて、校正証明ファイル情報のリストを返します。校正証明書を作成し、指定した出力ディレクトリにダウンロードします。
	GetChannelHardwareInfo	チャネルのハードウェア情報(アンプの種類など)を取得します。
	GetDeviceErrorStatusList	これは、GetDeviceErrorStatusListメソッドの多重定義の概要です。
	GetDeviceStatusList	これは、GetDeviceStatusListメソッドの多重定義の概要です。
	GetPossibleDataAcquisitionModes	デバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブル経由で確立)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedを含みます。デバイスが時間ベースの同期(PTPまたはNTPなど)を使用している場合、返されるリストにはDataAcquisitionMode.TimestampSynchronizedモードを含みます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはDataAcquisitionMode.Unsynchronizedモードを含む可能性が高くなります。デバイスが接続されていない場合、使用可能なデータ収集モードのリストは空になります。
	GetShunt	これは、GetShuntメソッドの多重定義の概要です。
	GetSupportedCanBitRates	サポートしているCANのビットレートを取得します。
	GetSupportedFilterFrequencies	指定したフィルタのサポートされているフィルタ周波数のリストを取得します。
	GetSupportedFilters	指定した信号のサポートされているフィルタのリストを取得します。
	GetSupportedJackType	指定した信号でサポートしているジャックの種類を取得します。
	GetSupportedSampleRates	サポートしているサンプルレートのリストを取得します。
	GetSupportedScalingTypes	指定したチャネルでサポートしているスケーリングタイプのリストを取得します。
	GetSupportedSensorTypes	指定したチャネルでサポートしているセンサタイプのリストを取得します。
	GetTimeSourceQuality	オフセット(同期タイムソースとデバイスの時間の間のタイムスパン)をミリ秒単位で返し、同期の品質に関する詳細情報を含むTimeSourceに依存する文字列を返します。NtpTimeSourceの場合、この文字列にはパラメータ(カンマ区切り)が含まれます: "remote、refid、st、t、when、pol、reach、delay、offset、jitter"。PtpTimeSourceの場合、この文字列には次のパラメータ (カンマ区切り)が含まれます: "GrandmasterID、SyncMode"、TimeScaleFlags、UtcOffset、UtcOffsetused、MasterOffset"
	GetUniqueID	すべてのデバイスファミリの中からデバイスを識別する一意の文字列を返します。
	GetUnit	指定したチャネルの単位を返します。
	IsChannelPartOfDevice	指定したチャネルがデバイスに属しているかどうかをチェックします。
	IsConnectorPartOfDevice	指定したコネクタがデバイスに属しているかどうかをチェックします。

	名前	説明
	IsFirmwareVersionSupported	指定したファームウェアのバージョンをサポートしているかどうかを判断します。
	IsSignalPartOfDevice	指定した信号がデバイスに属しているかどうかをチェックします。
	ReadDigitalIO	デジタル信号の現在の測定値を返します。
	ReadFbgSpectrum	指定したスペクトルタイプとコネクタに従ってFbgSpectrumを返します。この機能は、ファイバブラッググレーティング関連デバイスでのみサポートしいます。
	ReadSingleElectricalValue	指定したAnalogInSignalの現在の電気値を返します。
	ReadSingleMeasurementValue	DAQセッションを完全に初期化せずに、指定したSignalごとの単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadSingleMeasurementValueOfAllSignals	DAQセッションを完全に初期化せずに、デバイスのすべてのSignalに対して単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadTeds	指定したコネクタに接続されているTEDSからセンサを読み取ります。
	RemoveInputSignalReference	指定したチャネルから入力信号のリファレンスを削除します。
	Reset	デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数の呼び出し後に、デバイスは切断されます。これが必要なのはデバイスのパラメータ設定が変更され、デバイスがデバイスの関連オブジェクトモデルと一致しなくなったとき、工場出荷時のデフォルトをロードします(この関数を実装するときは、必ずデバイスを切断し、Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知します。
	ResetStatusMonitoringStates	ステータス監視状態をリセットして、次回のステータスポーリングですべてのイベントが再度発生できるようにします。
	SetAnalogOut	アナログ出力信号を指定した値に設定します。
	SetAnalogOutChannelActivation	提供されたチャネルのリストをアクティブまたは非アクティブにします。AnalogOutChannelのみをアクティブ化または非アクティブ化できます。
	SetCanBusParameters	指定したCanConnectorのすべてのCAN busパラメータ(CanConnector.BitRate、CanConnector.IsTerminated、CanConnector.SupplyVoltageなど)を割り当てますが、チャネルは割り当てません。
	SetCanChannelActivation	提供されたチャネルのリストをアクティブまたは非アクティブにします。CanInChannelとCanOutChannelのみをアクティブ化または非アクティブ化できます。
	SetChannelName	指定したチャネルにチャネル名を割り当てます。
	SetDataRateDomain	現在のモジュールに指定したdataRateDomainを設定します。重要:デバイスの種類によっては、再起動が自動的に行われます。この場合、デバイスはしばらくの間アクセスできなくなります。
	SetDeviceName	基になる物理デバイスの名前を設定します。派生クラスでオーバーライドする必要があります。
	SetDigitalIO	指定した値に従ってデジタル信号を設定します。
	SetFbgBandsDetectionResult	AutoDetectFbgBands経由で事前に取得した、指定されたFbgBandsDetectionResultに従って、指定されたFbgConnectorのFbgChannelのFBG関連プロパティ(DetectionRange、FbgSensor.ReferenceWavelength、FbgChannel.IsActivatedなど)を設定します。

	名前	説明
	SetFbgChannelActivation	指定したチャネルのリストをアクティブまたは非アクティブにします。FbgChannelのみをアクティブ化または非アクティブ化できます。
	SetFbgChannelDetectionRange	指定した値に従って、指定されたFbgChannelのFbgChannel.DetectionRangeを設定します。
	SetFbgDetectionThreshold	指定した値に従って、指定したFbgConnectorのFbgConnector.Detection Thresholdを設定します。
	SetInputSignalReference	入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。
	SetShunt	これは、SetShuntメソッドの多重定義の概要です。
	SetSignalName	指定した信号に信号名を割り当てます。
	SetTedsUsageMode	TEDS使用モードを設定し、コネクタ設定を再読み込みします。
	SetUnit	指定したチャネルの単位を割り当てます。
	SetZeroBalance	これは、SetZeroBalanceメソッドの多重定義の概要です。
	SubscribeToStatusMonitoring	指定したチャネルをステータス監視に登録します。
	UnsubscribeFromStatusMonitoring	指定したチャネルをステータス監視から登録解除します。
	WriteTeds	指定したコネクタに接続されたTEDSに指定したセンサを書き込みます。

Device.ActivateTeds Method

指定したコネクタのTEDS設定をアクティブにし、新しいチャネルパラメータをロードします。

C++

```
bool ActivateTeds(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public abstract bool ActivateTeds(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function ActivateTeds(
    connector As Connector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	接続されたTEDSをアクティブ化するコネクタ。
problems	空きまたは警告リスト、およびアクティブ化プロセス中に発生したエラー。

戻り値

TEDS設定がアクティブ化されている場合は`true`、それ以外の場合は`false`。

備考

成功すると、指定したコネクタのセンサオブジェクトと一部のチャネルプロパティが、TEDS設定に従って更新されたバージョンに置き換えられます。

Device.AssignAutoCalibrationMode Method

指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignAutoCalibrationMode(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public abstract bool AssignAutoCalibrationMode(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function AssignAutoCalibrationMode(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.AssignChannel Method

指定したチャネルの設定(センサ、信号などを含む)をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignChannel(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public abstract bool AssignChannel(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function AssignChannel(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル 。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	チャネルが`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

Device.AssignConnector Method

指定したコネクタの設定(チャネル、センサ信号などを含む)をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignConnector(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public abstract bool AssignConnector(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function AssignConnector(
    connector As Connector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	既存の(変更された)コネクタ。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

問題が発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	コネクタが`null`です。
Hbm.Api.Common.Exceptions.UnknownConnectorException	コネクタがこのデバイスに属していない場合にスローされます。

Device.AssignSensor Method

指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignSensor(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public abstract bool AssignSensor(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function AssignSensor(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.AssignSignal Method

指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignSignal(
    Signal signal,
    out List<Problem> problems
);
```

C#

```
public abstract bool AssignSignal(
    Signal signal,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function AssignSignal(
    signal As Signal,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
signal	既存の(変更された)信号。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	信号が`null`です。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

Device.AssignTimeSource Method

TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignTimeSource(
    out List<Problem> problems
);
```

C#

```
public abstract bool AssignTimeSource(
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function AssignTimeSource(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.AssignZero Method

指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignZero(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public abstract bool AssignZero(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function AssignZero(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.AutoDetectFbgBands Method

指定したFbgConnectorに対する帯域検出の結果を返します。

C++

```
FbgBandsDetectionResult AutoDetectFbgBands(
    FbgConnector fbgConnector,
    double threshold,
    double bandWidth
);
```

C#

```
public virtual FbgBandsDetectionResult AutoDetectFbgBands(
    FbgConnector fbgConnector,
    double threshold,
    double bandWidth
);
```

Visual Basic

```
Public virtual Function AutoDetectFbgBands(
    fbgConnector As FbgConnector,
    threshold As double,
    bandWidth As double
) As FbgBandsDetectionResult
```

引数

引数	説明
fbgConnector	検出を実行するコネクタ。
threshold	帯域内のピーク検出のしきい値(dB)。MXFSの場合は3dBのみ可能です。
bandWidth	検出するバンドの最小要求幅(nm)。

戻り値

指定した検出パラメータに従って検出された帯域。

Device.BroadcastChannelStatusChanges Method

ステータスの変化をブロードキャストします。

C++

```
void BroadcastChannelStatusChanges(
    Dictionary<StatusType, List<DeviceStatus>> statusLists
);
```

C#

```
protected void BroadcastChannelStatusChanges(
    Dictionary<StatusType, List<DeviceStatus>> statusLists
);
```

Visual Basic

```
Protected Function BroadcastChannelStatusChanges(
    statusLists As Dictionary<StatusType, List<DeviceStatus>>
) As void
```

引数

引数	説明
statusLists	さまざまなステータスタイプのステータスリスト辞書。

Device.BroadcastTedsStatus Method

tedステータスを処理します。

C++

```
void BroadcastTedsStatus(
    List<DeviceStatus>> devicesStatus
);
```

C#

```
protected void BroadcastTedsStatus(
    List<DeviceStatus> devicesStatus
);
```

Visual Basic

```
Protected Function BroadcastTedsStatus(
    devicesStatus As List<DeviceStatus>
) As void
```

引数

引数	説明
devicesStatus	デバイスの状態。

Device.ChangeIp Method

デバイスのipAddressを変更します。

C++

```
bool ChangeIp(
    string ipAddress,
    string subnetMask,
    bool isDhcp,
    string gateWay,
    unsigned tcpPort,
    string interfaceName,
    out List<Problem> problems
);
```

C#

```
public abstract bool ChangeIp(
    string ipAddress,
    string subnetMask,
    bool isDhcp,
    string gateWay,
    uint tcpPort,
    string interfaceName,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function ChangeIp(
    ipAddress As string,
    subnetMask As string,
    isDhcp As bool,
    gateWay As string,
    tcpPort As UInteger,
    interfaceName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
ipAddress	適用するIPアドレス。
subnetMask	サブネットマスク。
isDhcp	自動割り当てであるかどうかを示す表示器。
gateWay	ゲートウェイ。
tcpPort	TCPポート。
interfaceName	ネットワークインタフェースの名前。
problems	問題のリスト。

戻り値

成功した場合はtrue、それ以外の場合はfalse。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Device Class -> Device Methods

ClearZeroBalance Method

これは、ClearZeroBalanceメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	Device.ClearZeroBalance (Channel, out List<Problem>)	Zero.Offsetを0に設定することで、指定したチャンネルのゼロバランスをクリアします。Zero.Target値は影響を受けません!
	Device.ClearZeroBalance (List<Channel>, out List<Problem>)	Zero.Offsetを0に設定することで、指定したチャンネルのゼロバランスをクリアします。Zero.Target値は影響を受けません!

Device.ClearZeroBalance Method (Channel, out List<Problem>)

Zero.Offsetを0に設定することで、指定したチャンネルのゼロバランスをクリアします。Zero.Target値は影響を受けません!

C++

```
bool ClearZeroBalance(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public bool ClearZeroBalance(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function Connect(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	ゼロバランスをクリアするチャネル。
problems	ゼロバランスのクリア中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

Device.ClearZeroBalance Method (List<Channel>, out List<Problem>)

Zero.Offsetを0に設定することで、指定したチャンネルのゼロバランスをクリアします。Zero.Target値は影響を受けません!

C++

```
bool ClearZeroBalance(
    List<Channel> channels,
    out List<Problem> problems
);
```

C#

```
public bool ClearZeroBalance(
    List<Channel> channels,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function Connect(
    channels As <Channel>,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channels	ゼロバランスをクリアするチャネル。
problems	ゼロバランスのクリア中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	チャンネルが`null`です。

Device.DisableRouting Method

ルーティング機能を無効にします。

C++

```
bool DisableRouting(
    out List<Problem> problems
);
```

C#

```
public virtual bool DisableRouting(
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function DisableRouting(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

Device.EnableRouting Method

ルーティング機能を有効にします。

C++

```
bool EnableRouting(
    out List<Problem> problems
);
```

C#

```
public virtual bool EnableRouting(
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function EnableRouting(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

Device.Find Method

指定した接続情報を使用してデバイスを検索します(実際には、コネクタ情報を読み取ることなくデバイス情報を読み取ります)。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool Find(
    ConnectionInfo connectionInfo,
    out List<Problem> problems
);
```

C#

```
public virtual bool Find(
    ConnectionInfo connectionInfo,
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function Find(
    connectionInfo As ConnectionInfo,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connectionInfo	デバイス情報の検索に使用する接続情報。
problems	デバイスへの接続中およびデバイス情報の読み取り中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	接続情報が`null`です。

FindChannel Method

これは、FindChannelメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	Device.FindChannel (Sensor)	指定したセンサが配置されているチャネルを返します。
	Device.FindChannel (Signal)	指定した信号が配置されているチャネルを返します。

Device.FindChannel Method (Sensor)

指定したセンサが配置されているチャネルを返します。

C++

```
Channel FindChannel(
    Sensor sensor
);
```

C#

```
public Channel FindChannel(
    Sensor sensor
);
```

Visual Basic

```
Public Function FindChannel(
    sensor As Sensor
) As Channel
```

引数

引数	説明
sensor	センサ。

戻り値

指定したセンサが配置されているチャネル、またはチャネルが見つからない場合はnull。

Device.FindChannel Method (Signal)

指定した信号が配置されているチャネルを返します。

C++

```
Channel FindChannel(
    Signal signal
);
```

C#

```
public Channel FindChannel(
    Signal signal
);
```

Visual Basic

```
Public Function FindChannel(
    signal As Signal
) As Channel
```

引数

引数	説明
signal	信号。

戻り値

指定した信号が配置されているチャネル、またはチャネルが見つからない場合はnull。

FindConnector Method

これは、FindConnectorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	Device.FindConnector (Channel)	指定したチャネルが配置されているコネクタを返します。
	Device.FindConnector (Sensor)	指定したセンサが配置されているコネクタを返します。
	Device.FindConnector (Signal)	指定した信号が配置されているコネクタを返します。

Device.FindConnector Method (Channel)

指定したチャネルが配置されているコネクタを返します。

C++

```
Connector FindConnector(
    Channel channel
);
```

C#

```
public Connector FindConnector(
    Channel channel
);
```

Visual Basic

```
Public Function FindConnector(
    channel As Channel
) As Connector
```

引数

引数	説明
channel	チャネル。

戻り値

指定されたチャネルが配置されているコネクタ、またはコネクタが見つからない場合はnull。

Device.FindConnector Method (Sensor)

指定したセンサが配置されているコネクタを返します。

C++

```
Connector FindConnector(
    Sensor sensor
);
```

C#

```
public Connector FindConnector(
    Sensor sensor
);
```

Visual Basic

```
Public Function FindConnector(
    sensor As Sensor
) As Connector
```

引数

引数	説明
channel	センサ。

戻り値

指定されたチャネルが配置されているコネクタ、またはコネクタが見つからない場合はnull。

Device.FindConnector Method (Signal)

指定した信号が配置されているコネクタを返します。

C++

```
Connector FindConnector(
    Signal signal
);
```

C#

```
public Connector FindConnector(
    Signal signal
);
```

Visual Basic

```
Public Function FindConnector(
    signal As Signal
) As Connector
```

引数

引数	説明
signal	信号。

戻り値

指定されたチャネルが配置されているコネクタ、またはコネクタが見つからない場合はnull。

Device.FindSignal Method

指定したフィルタが配置されている信号を返します。

C++

```
Signal FindSignal(
    Filter filter
);
```

C#

```
public Signal FindSignal(
    Filter filter
);
```

Visual Basic

```
Public Function FindSignal(
    filter As Filter
) As Signal
```

引数

引数	説明
Filter	フィルタ。

戻り値

指定されたフィルタが配置されている信号、または信号が見つからない場合はnull。

FlashLed Method

これは、FlashLedメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	Device.FlashLed (Connector, LedFlashMode, out List<Problem>)	指定したコネクタのLEDを、指定したモード/色で点滅させます。
	Device.FlashLed (LedFlashMode, out List<Problem>)	デバイスのLEDを指定したモード/色で点滅させます。
	Device.FlashLed (List<Connector>, LedFlashMode, out List<Problem>)	指定したコネクタのLEDを、指定したモード/色で点滅させます。

Device.FlashLed Method (Connector, LedFlashMode, out List<Problem>)

指定したコネクタのLEDを、指定したモード/色で点滅させます。

C++

```
bool FlashLed(
    Connector connector,
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public abstract bool FlashLed(
    Connector connector,
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function FlashLed(
    connector As Connector,
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	点滅させるコネクタ。
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Device Class -> Device Methods -> FlashLed Method

Device.FlashLed Method (LedFlashMode, out List<Problem>)

デバイスのLEDを指定したモード/色で点滅させます。

C++

```
bool FlashLed(
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public abstract bool FlashLed(
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function FlashLed(
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.FlashLed Method (List<Connector>, LedFlashMode, out List<Problem>)

指定したコネクタのLEDを、指定したモード/色で点滅させます。

C++

```
bool FlashLed(
    List<Connector> connectors,
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public abstract bool FlashLed(
    List<Connector> connectors,
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function FlashLed(
    connectors As List<Connector>,
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	点滅させるコネクタ。
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.GetAllChannels Method

デバイスのすべてのチャネルを取得します。

C++

```
List<Channel> GetAllChannels();
```

C#

```
public List<Channel> GetAllChannels();
```

Visual Basic

```
Public Function GetAllChannels() As List<Channel>
```

戻り値

現在のデバイスのチャネルリスト。

Device.GetAllSignals Method

デバイスのすべての信号を取得します。

C++

```
List<Signal> GetAllSignals();
```

C#

```
public List<Signal> GetAllSignals();
```

Visual Basic

```
Public Function GetAllSignals() As List<Signal>
```

戻り値

現在のデバイスの信号リスト。

Device.GetAmplifierSerialNumber Method

チャネルのアンプのシリアル番号を取得します。

C++

```
string GetAmplifierSerialNumber(
    Channel channel
);
```

C#

```
public abstract string GetAmplifierSerialNumber(
    Channel channel
);
```

Visual Basic

```
Public abstract Function GetAmplifierSerialNumber(
    channel As Channel
) As string
```

引数

引数	説明
channel	アンプのシリアル番号を返すチャネル。

戻り値

指定したチャネルのアンプのシリアル番号。

Device.GetAmplifierType Method

チャネルのアンプタイプを取得します。

C++

```
string GetAmplifierType(
    Channel channel
);
```

C#

```
public abstract string GetAmplifierType(
    Channel channel
);
```

Visual Basic

```
Public abstract Function GetAmplifierType(
    channel As Channel
) As string
```

引数

引数	説明
channel	アンプのタイプを返すチャネル。

戻り値

指定したチャネルのアンプタイプ。

Device.GetAndBroadcastDeviceStatusList Method

デバイスのステータスリストを取得し、サブスクリプションをフィルタリングして、変更された状態をブロードキャストします。

C++

```
void GetAndBroadcastDeviceStatusList(
    StatusType statusTypesToGet
);
```

C#

```
internal void GetAndBroadcastDeviceStatusList(
    StatusType statusTypesToGet
);
```

Visual Basic

```
Friend Function GetAndBroadcastDeviceStatusList(
    statusTypesToGet As StatusType
) As void
```

引数

引数	説明
statusTypesToGet	StatusTypeには要求したステータスタイプが含まれます。

Device.GetBestFittingAutoFilter Method

信号のサンプルレートに応じて最適な自動フィルタを取得します。

C++

```
Filter GetBestFittingAutoFilter(
    Signal signal
);
```

C#

```
public virtual Filter GetBestFittingAutoFilter(
    Signal signal
);
```

Visual Basic

```
Friend virtual Function GetBestFittingAutoFilter(
    signal As Signal
) As Filter
```

引数

引数	説明
signal	最適なフィルタを取得するための信号。

戻り値

最適なフィルタ。

Device.GetCalibrationCertificateFileInfos Method

要求したリリースの種類に応じて、校正証明ファイル情報のリストを返します。校正証明書を作成し、指定した出力ディレクトリにダウンロードします。

C++

```
List<CalibrationCertificateFileInfo> GetCalibrationCertificateFileInfos(
    ReleaseType releaseType,
    string outputDirectory,
    out List<Problem> problems
);
```

C#

```
public virtual List<CalibrationCertificateFileInfo> GetCalibrationCertificateFileInfos(
    ReleaseType releaseType,
    string outputDirectory,
    out List<Problem> problems
);
```

Visual Basic

```
Friend virtual Function GetCalibrationCertificateFileInfos(
    releaseType As ReleaseType,
    outputDirectory As string,
    problems As out List<Problem>
) As CalibrationCertificateFileInfo
```

引数

引数	説明
releaseType	要求した校正証明のリリースタイプ。
outputDirectory	校正証明を保存/作成するディレクトリ。
problems	校正証明の取得中に発生した警告とエラー。

戻り値

校正証明ファイル情報のリスト。

Device.GetChannelHardwareInfo Method

チャネルのハードウェア情報(アンプの種類など)を取得します。

C++

```
string GetChannelHardwareInfo(
    Channel channel
);
```

C#

```
public abstract string GetChannelHardwareInfo(
    Channel channel
);
```

Visual Basic

```
Public abstract Function GetChannelHardwareInfo(
    channel As Channel
) As string
```

引数

引数	説明
channel	ハードウェア情報が返されるチャネル。

戻り値

指定したチャネルのハードウェア情報。

GetDeviceErrorStatusList Method

これは、GetDeviceErrorStatusListメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	Device.GetDeviceErrorStatusList (Channel, out List<Problem>)	指定したチャネルのデバイスエラーのステータスの一覧を取得します。
	Device.GetDeviceErrorStatusList (Connector, out List<Problem>)	指定したコネクタのデバイスエラーのステータスの一覧を取得します。
	Device.GetDeviceErrorStatusList (out List<Problem>)	デバイスエラーのステータスの一覧を取得します。

Device.GetDeviceErrorStatusList Method (Channel, out List<Problem>)

指定したチャネルのデバイスエラーのステータスの一覧を取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public virtual List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function GetDeviceErrorStatusList(
    channel As Channel,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
channel	ステータスを取得するチャネル。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

デバイスエラーのステータスの一覧。

Device.GetDeviceErrorStatusList Method (Connector, out List<Problem>)

指定したコネクタのデバイスエラーのステータスの一覧を取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public virtual List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function GetDeviceErrorStatusList(
    connector As Connector,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
connector	ステータスを取得するコネクタ。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

デバイスエラーのステータスの一覧。

Device.GetDeviceErrorStatusList Method (out List<Problem>)

デバイスエラーのステータスの一覧を取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    out List<Problem> problems
);
```

C#

```
public virtual List<DeviceErrorStatus> GetDeviceErrorStatusList(
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function GetDeviceErrorStatusList(
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

デバイスエラーのステータスの一覧。

GetDeviceStatusList Method

これは、GetDeviceStatusListメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	Device.GetDeviceStatusList (Channel, out List<Problem>)	指定したチャネルのデバイスステータスの一覧を取得します。
	Device.GetDeviceStatusList (Connector, out List<Problem>)	指定したコネクタのデバイスステータスの一覧を取得します。
	Device.GetDeviceStatusList (out List<Problem>, StatusType)	デバイスステータスの一覧を取得します。

Device.GetDeviceStatusList Method (Channel, out List<Problem>)

指定したチャネルのデバイスステータスの一覧を取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public abstract List<DeviceStatus> GetDeviceStatusList(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function GetDeviceStatusList(
    channel As Channel,
    problems As out List<Problem>
) As List<DeviceStatus>
```

引数

引数	説明
channel	ステータスを取得するチャネル。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

デバイスのステータスの一覧。

Device.GetDeviceStatusList Method (Connector, out List<Problem>)

指定したコネクタのデバイスステータスの一覧を取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public abstract List<DeviceStatus> GetDeviceStatusList(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function GetDeviceStatusList(
    connector As Connector,
    problems As out List<Problem>
) As List<DeviceStatus>
```

引数

引数	説明
connector	ステータスを取得するコネクタ。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

デバイスのステータスの一覧。

Device.GetDeviceStatusList Method (out List<Problem>, StatusType)

デバイスステータスの一覧を取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    out List<Problem> problems,
    StatusType statusTypesToGet = StatusType.All
);
```

C#

```
public abstract List<DeviceStatus> GetDeviceStatusList(
    out List<Problem> problems,
    StatusType statusTypesToGet
);
```

Visual Basic

```
Public abstract Function GetDeviceStatusList(
    problems As out List<Problem>,
    statusTypesToGet As StatusType = StatusType.All
) As List<DeviceStatus>
```

引数

引数	説明
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。
statusTypesToGet	要求したステータスタイプを含むStatusType(デフォルトはすべてのステータス情報)。

戻り値

デバイスのステータスの一覧。

Device.GetPossibleDataAcquisitionModes Method

デバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブル経由で確立)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedを含みます。デバイスが時間ベースの同期(PTPまたはNTPなど)を使用している場合、返されるリストにはDataAcquisitionMode.TimestampSynchronizedモードを含みます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはDataAcquisitionMode.Unsynchronizedモードを含む可能性が高くなります。デバイスが接続されていない場合、使用可能なデータ収集モードのリストは空になります。

C++

```
List<DataAcquisitionMode> GetPossibleDataAcquisitionModes();
```

C#

```
public abstract List<DataAcquisitionMode> GetPossibleDataAcquisitionModes();
```

Visual Basic

```
Public abstract Function GetPossibleDataAcquisitionModes() As List<DataAcquisitionMode>
```

戻り値

現在の同期設定に応じて、このデバイスで可能なDataAcquisitionModeの一覧。

GetShunt Method

これは、GetShuntメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	Device.GetShunt (Connector)	指定したコネクタのShuntModeを取得します。
	Device.GetShunt (List<Connector>)	指定したコネクタのShuntModeを取得します。

Device.GetShunt Method (Connector)

指定したコネクタのShuntModeを取得します。

C++

```
ShuntMode GetShunt(
    Connector connector
);
```

C#

```
public abstract ShuntMode GetShunt(
    Connector connector
);
```

Visual Basic

```
Public abstract Function GetShunt(
    connector As Connector
) As ShuntMode
```

引数

引数	説明
connector	ShuntModeを取得するコネクタ。

戻り値

指定したコネクタのShuntMode。

Device.GetShunt Method (List<Connector>)

指定したコネクタのShuntModeを取得します。

C++

```
List<ShuntMode> GetShunt(
    List<Connector> connectors
);
```

C#

```
public abstract List<ShuntMode> GetShunt(
    List<Connector> connectors
);
```

Visual Basic

```
Public abstract Function GetShunt(
    connectors As List<Connector>
) As List<ShuntMode>
```

引数

引数	説明
connector	ShuntModeを取得するコネクタ。

戻り値

指定したコネクタのShuntMode。

Device.GetSupportedCanBitRates Method

サポートしているCANのビットレートを取得します。

C++

```
List<int> GetSupportedCanBitRates();
```

C#

```
public virtual List<int> GetSupportedCanBitRates();
```

Visual Basic

```
Public virtual Function GetSupportedCanBitRates() As List<int>
```

戻り値

サポートしているCANのビットレートリスト。

Device.GetSupportedFilterFrequencies Method

指定したフィルタのサポートされているフィルタ周波数のリストを取得します。

C++

```
List<decimal> GetSupportedFilterFrequencies(
    Filter filter,
    bool isOffineQuery = false
);
```

C#

```
public abstract List<decimal> GetSupportedFilterFrequencies(
    Filter filter,
    bool isOffineQuery
);
```

Visual Basic

```
Public abstract Function GetSupportedFilterFrequencies(
    filter As Filter,
    isOffineQuery As bool = false
) As List<decimal>
```

引数

引数	説明
filter	取得したいフィルタ周波数を含むフィルタ。
isOfflineQuery	True、可能であればデバイスと通信せずにフィルタを返すべきです(パフォーマンス上の理由など)。

戻り値

利用可能なフィルタ周波数のリスト。

例外

例外	説明
System.ArgumentNullException	フィルタは`null`です。
Hbm.Api.Common.Exceptions.UnknownFilterException	フィルタがこのデバイスに属していない場合にスローされます。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Device Class -> Device Methods

Device.GetSupportedFilters Method

指定した信号のサポートされているフィルタのリストを取得します。

C++

```
List<Filter> GetSupportedFilters(
    Signal signal,
    bool isOfflineQuery = false
);
```

C#

```
public abstract List<Filter> GetSupportedFilters(
    Signal signal,
    bool isOfflineQuery
);
```

Visual Basic

```
Public abstract Function GetSupportedFilters(
    signal As Signal,
    isOfflineQuery As bool = false
) As List<Filter>
```

引数

引数	説明
signal	取得したいフィルタを含む信号。
isOfflineQuery	True、可能であればデバイスと通信せずにフィルタを返すべきです(パフォーマンス上の理由など)。

戻り値

利用可能なフィルタのリスト。

例外

例外	説明
System.ArgumentNullException	信号は`null`です。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

Device.GetSupportedJackType Method

指定した信号でサポートしているジャックの種類を取得します。

C++

```
string GetSupportedJackType(
    Signal signal,
    bool isOffineQuery = false
);
```

C#

```
public abstract string GetSupportedJackType(
    Signal signal,
    bool isOffineQuery
);
```

Visual Basic

```
Public abstract Function GetSupportedJackType(
    signal As Signal,
    isOffineQuery As bool = false
) As string
```

引数

引数	説明
signal	信号。
isOfflineQuery	True、可能であればデバイスと通信せずにジャックの種類を返すべきです(パフォーマンス上の理由など)。

戻り値

ジャックの種類の名前。

Device.GetSupportedSampleRates Method
サポートしているサンプルレートのリストを取得します。

C++

```
List<decimal> GetSupportedSampleRates(
    Signal signal,
    bool isOfflineQuery = false
);
```

C#

```
public abstract List<decimal> GetSupportedSampleRates(
    Signal signal,
    bool isOfflineQuery
);
```

Visual Basic

```
Public abstract Function GetSupportedSampleRates(
    signal As Signal,
    isOfflineQuery As bool = false
) As List<decimal>
```

引数

引数	説明
signal	取得したいサンプルレートの信号。
isOfflineQuery	True、可能であればデバイスと通信せずにサンプルレートを返すべきです(パフォーマンス上の理由など)。

戻り値
利用可能なサンプルレートのリスト。

例外

例外	説明
System.ArgumentNullException	信号は`null`です。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

Device.GetSupportedScalingTypes Method

指定したチャネルでサポートしているスケーリングタイプのリストを取得します。

C++

```
List<ScalingType> GetSupportedScalingTypes(
    Channel channel,
    bool isOffineQuery = false
);
```

C#

```
public abstract List<ScalingType> GetSupportedScalingTypes(
    Channel channel,
    bool isOffineQuery
);
```

Visual Basic

```
Public abstract Function GetSupportedScalingTypes(
    channel As Channel,
    isOffineQuery As bool = false
) As List<ScalingType>
```

引数

引数	説明
channel	取得したいスケーリングタイプのチャネル。
isOfflineQuery	True、可能であればデバイスと通信せずにスケーリングタイプを返すべきです(パフォーマンス上の理由など)。

戻り値

利用可能なスケーリングタイプのリスト。

例外

例外	説明
System.ArgumentNullException	チャネルは`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

Device.GetSupportedSensorTypes Method

指定したチャネルでサポートしているセンサタイプのリストを取得します。

C++

```
List<SensorType> GetSupportedScalingTypes(
    Channel channel,
    bool isOfflineQuery = false
);
```

C#

```
public abstract List<SensorType> GetSupportedScalingTypes(
    Channel channel,
    bool isOfflineQuery
);
```

Visual Basic

```
Public abstract Function GetSupportedScalingTypes(
    channel As Channel,
    isOfflineQuery As bool = false
) As List<SensorType>
```

引数

引数	説明
channel	取得したいスケーリングタイプのチャネル。
isOfflineQuery	True、可能であればデバイスと通信せずにセンサタイプを返すべきです(パフォーマンス上の理由など)。

戻り値

利用可能なセンサタイプのリスト。

例外

例外	説明
System.ArgumentNullException	チャネルは`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

Device.GetTimeSourceQuality Method

オフセット(同期タイムソースとデバイスの時間の間のタイムスパン)をミリ秒単位で返し、同期の品質に関する詳細情報を含むTimeSourceに依存する文字列を返します。NtpTimeSourceの場合、この文字列にはパラメータ(カンマ区切り)が含まれます: "remote、refid、st、t、when、pol、reach、delay、offset、jitter"。PtpTimeSourceの場合、この文字列には次のパラメータ (カンマ区切り)が含まれます: "GrandmasterID、SyncMode"、TimeScaleFlags、UtcOffset、UtcOffsetused、MasterOffset"

C++

```
void GetTimeSourceQuality(
    out double offsetInMs,
    out string quality

);
```

C#

```
public abstract void GetTimeSourceQuality(
    out double offsetInMs,
    out string quality
);
```

Visual Basic

```
Public abstract Function GetTimeSourceQuality(
    offsetInMs As out double,
    quality As out string
) As void
```

引数

引数	説明
offsetInMs	タイムソースへのオフセット(ミリ秒単位)、オフセットが決定できない場合はdouble.MaxValue。
quality	TimeSourceの種類に依存した同期品質に関する情報。

Device.GetUniqueID Method

すべてのデバイスファミリの中からデバイスを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public virtual string GetUniqueID();
```

Visual Basic

```
Public virtual Function GetUniqueID() As string
```

戻り値

一意のデバイスID、またはIDが設定されていない場合は`String.Empty`。

備考

デフォルトでは、IDはFamilyNameとSerialNoの間にハイフンを挟んで構成されます。このメソッドを呼び出す前に、プロパティSerialNoを設定する必要があります。通常、これはスキャンまたは接続中に行われます。

Device.GetUnit Method

指定したチャネルの単位を返します。

C++

```
string GetUnit(
    Channel channel
);
```

C#

```
public virtual string GetUnit(
    Channel channel
);
```

Visual Basic

```
Public virtual Function GetUnit(
    channel As Channel
) As void
```

引数

引数	説明
channel	単位を返すチャネル。

戻り値

指定したチャネルの単位。

例外

例外	説明
System.ArgumentNullException	チャネルは`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

Device.IsChannelPartOfDevice Method

指定したチャネルがデバイスに属しているかどうかをチェックします。

C++

```
bool IsChannelPartOfDevice(
    Channel channelToCheck
);
```

C#

```
public bool IsChannelPartOfDevice(
    Channel channelToCheck
);
```

Visual Basic

```
Public Function IsChannelPartOfDevice(
    channelToCheck As Channel
) As bool
```

引数

引数	説明
channelToCheck	チェックするチャネルのリファレンス。

戻り値

チャネルがデバイスのチャネルの場合はtrue。

例外

例外	説明
ArgumentNullException	channelToCheckは`null`です。

Device.IsConnectorPartOfDevice Method

指定したコネクタがデバイスに属しているかどうかをチェックします。

C++

```
bool IsConnectorPartOfDevice(
    Connector connectorToCheck
);
```

C#

```
public bool IsConnectorPartOfDevice(
    Connector connectorToCheck
);
```

Visual Basic

```
Public Function IsConnectorPartOfDevice(
    connectorToCheck As Connector
) As bool
```

引数

引数	説明
connectorToCheck	

戻り値

コネクタがデバイスのコネクタの場合はtrue。

例外

例外	説明
System.ArgumentNullException	connectorToCheckは`null`です。

Device.IsFirmwareVersionSupported Method

指定したファームウェアのバージョンをサポートしているかどうかを判断します。

C++

```
bool IsFirmwareVersionSupported(
    string firmwareVersion
);
```

C#

```
public abstract bool IsFirmwareVersionSupported(
    string firmwareVersion
);
```

Visual Basic

```
Public abstract Function IsFirmwareVersionSupported(
    firmwareVersion As string
) As bool
```

引数

引数	説明
firmwareVersion	チェックするファームウェアのバージョン。

戻り値

指定したファームウェアのバージョンがサポートされていない場合、false。

備考

この関数は、指定したファームウェアのバージョンが(APIがリリースされた時点で)最新の既知のバージョンよりも高い場合にtrueを返します。

例外

例外	説明
System.ArgumentNullException	ファームウェアのバージョンは`null`または`empty`です。

Device.IsSignalPartOfDevice Method

指定した信号がデバイスに属しているかどうかをチェックします。

C++

```
bool IsSignalPartOfDevice(
    Signal signalToCheck
);
```

C#

```
public bool IsSignalPartOfDevice(
    Signal signalToCheck
);
```

Visual Basic

```
Public Function IsSignalPartOfDevice(
    signalToCheck As Signal
) As bool
```

引数

引数	説明
signalToCheck	チェックする信号のリファレンス。

戻り値

信号がデバイスの信号の場合はtrue。

例外

例外	説明
System.ArgumentNullException	signalToCheckは`null`です。

Device.MakeChecks Method

複数のメソッド間で使用するいくつかのプロパティのチェックを集中管理するヘルパメソッド。

C++

```
bool MakeChecks(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public bool MakeChecks(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function MakeChecks(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	チェックするチャネル。
problems	問題のリスト。

戻り値

指定したチャネルが有効な場合はtrue。それ以外の場合はfalse。

Device.ReadDigitalIO Method

デジタル信号の現在の測定値を返します。

C++

```
double ReadDigitalIO(
    DigitalSignal digitalSignal
);
```

C#

```
public virtual double ReadDigitalIO(
    DigitalSignal digitalSignal
);
```

Visual Basic

```
Public virtual Function ReadDigitalIO(
    digitalSignal As DigitalSignal
) As double
```

引数

引数	説明
digitalSignal	測定値を取得するデジタル信号。

戻り値

指定したデジタル信号の現在の測定値。

Device.ReadFbgSpectrum Method

指定したスペクトルタイプとコネクタに従ってFbgSpectrumを返します。この機能は、ファイバブラッググレーティング関連デバイスでのみサポートしいます。

C++

```
FbgSpectrum ReadFbgSpectrum(
    FbgSpectrumType fbgSpectrumType,
    FbgConnector fbgConnector
);
```

C#

```
public virtual FbgSpectrum ReadFbgSpectrum(
    FbgSpectrumType fbgSpectrumType,
    FbgConnector fbgConnector
);
```

Visual Basic

```
Public virtual Function ReadFbgSpectrum(
    fbgSpectrumType As FbgSpectrumType,
    fbgConnector As FbgConnector
) As FbgSpectrum
```

引数

引数	説明
fbgSpectrumType	読み取りたいスペクトルの種類。
fbgConnector	スペクトルを読み取るコネクタ。

戻り値

指定したFbgSpectrumTypeに基づくスペクトル。

Device.ReadSingleElectricalValue Method

指定したAnalogInSignalの現在の電気値を返します。

C++

```
double ReadSingleElectricalValue(
    AnalogInSignal analogInSignal
);
```

C#

```
public virtual double ReadSingleElectricalValue(
    AnalogInSignal analogInSignal
);
```

Visual Basic

```
Public virtual Function ReadSingleElectricalValue(
    analogInSignal As AnalogInSignal
) As double
```

引数

引数	説明
analogInSignal	AnalogInSignalから電気値を取得します。

戻り値

指定したAnalogInSignalの現在の電気値、または電気値が要求できない場合はオーバーフロー。

Device.ReadSingleMeasurementValue Method

DAQセッションを完全に初期化せずに、指定したSignalごとの単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。

C++

```
void ReadSingleMeasurementValue(
    List<Signal> measurementSignals
);
```

C#

```
public abstract void ReadSingleMeasurementValue(
    List<Signal> measurementSignals
);
```

Visual Basic

```
Public abstract Function ReadSingleMeasurementValue(
    measurementSignals As List<Signal>
) As void
```

引数

引数	説明
measurementSignals	測定値を取得する信号。

備考

デバイスが単一の測定値のタイムスタンプをサポートしていない場合、PCの現在時刻がタイムスタンプの生成に使用されます。

Device.ReadSingleMeasurementValueOfAllSignals Method

DAQセッションを完全に初期化せずに、デバイスのすべてのSignalに対して単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。

C++

```
void ReadSingleMeasurementValueOfAllSignals();
```

C#

```
Public abstract void ReadSingleMeasurementValueOfAllSignals();
```

備考

デバイスが単一の測定値のタイムスタンプをサポートしていない場合、PCの現在時刻がタイムスタンプの生成に使用されます。

Device.ReadTeds Method

指定したコネクタに接続されているTEDSからセンサを読み取ります。

C++

```
Sensor ReadTeds(
    Connector connector,
    out string tid,
    out List<Problem> problems
);
```

C#

```
public abstract Sensor ReadTeds(
    Connector connector,
    out string tid,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function ReadTeds(
    connector As Connector,
    tid As out string,
    problems As out List<Problem>
) As Sensor
```

引数

引数	説明
connector	コネクタ。
tid	TEDS ID。
problems	問題。

戻り値

センサ(TEDSにHBM Channel Nameテンプレートがある場合、`sensor.Description`がそれに応じて入力されます)。

Device.RemoveInputSignalReference Method

指定したチャネルから入力信号のリファレンスを削除します。

C++

```
bool RemoveInputSignalReference(
    IInputSignalReference channel,
    out List<Problem> problems
);
```

C#

```
public abstract bool RemoveInputSignalReference(
    IInputSignalReference channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function RemoveInputSignalReference(
    channel As IInputSignalReference,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	入力信号リファレンスが削除される(出力)チャネル。
problems	エラーが発生していない場合は`true`、それ以外の場合は`false`。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	チャネル、inputSignalDevice、またはinputSignalが`null`です。
System.ArgumentException	inputSignalがDigitalSignalまたはAnalogOutSignalの場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownSignalException	入力信号がこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	デバイスが接続されていない場合にスローされます。

Device.Reset Method

デバイスを工場出荷時のデフォルト設定にリセットします。注意！この関数の呼び出し後に、デバイスは切断されます。これが必要なのはデバイスのパラメータ設定が変更され、デバイスがデバイスの関連オブジェクトモデルと一致しなくなったとき、工場出荷時のデフォルトをロードします(この関数を実装するときは、必ずデバイスを切断し、Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知します。

C++

```
void Reset(
    out List<Problem> problems
);
```

C#

```
public abstract void Reset(
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function Reset(
    problems As out List<Problem>
) As void
```

引数

引数	説明
problems	デバイスのリセット中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

備考

一定時間デバイスにアクセスできない場合があり、この関数の実行後にデバイスが切断されることに注意してください。

Device.ResetStatusMonitoringStates Method

ステータス監視状態をリセットして、次回のステータスポーリングですべてのイベントが再度発生できるようにします。

C++

```
void ResetStatusMonitoringStates();
```

C#

```
public void ResetStatusMonitoringStates();
```

Visual Basic

```
Public Function ResetStatusMonitoringStates() As void
```

Device.SetAnalogOut Method

アナログ出力信号を指定した値に設定します。

C++

```
void SetAnalogOut(
    AnalogOutSignal analogOutSignal,
    decimal value
);
```

C#

```
public abstract void SetAnalogOut(
    AnalogOutSignal analogOutSignal,
    decimal value
);
```

Visual Basic

```
Public abstract Function SetAnalogOut(
    analogOutSignal As AnalogOutSignal,
    value As decimal
) As void
```

引数

引数	説明
analogOutSignal	調整するアナログ出力信号。
value	設定する値。

Device.SetAnalogOutChannelActivation Method

提供されたチャネルのリストをアクティブまたは非アクティブにします。AnalogOutChannelのみをアクティブ化または非アクティブ化できます。

C++

```
bool SetAnalogOutChannelActivation(
    List<Channel> analogOutChannels,
    bool activate,
    out List<Problem> problems
);
```

C#

```
public virtual bool SetAnalogOutChannelActivation(
    List<Channel> analogOutChannels,
    bool activate,
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function SetAnalogOutChannelActivation(
    analogOutSignal As List<Channel>,
    activate As bool,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
analogOutSignal	アクティブまたは非アクティブにするアナログ出力チャネル。
activate	チャネルをアクティブにする場合はtrue、それ以外の場合はfalse。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合はtrue、それ以外の場合はfalse。

Device.SetAnalogOutOutputScaling Method

アナログ出力信号のデフォルトの出力テーブルのスケーリングを設定します。

C++

```
List<Problem> SetAnalogOutOutputScaling(
    AnalogInChannel inputChannel,
    AnalogOutSignal analogOutSignal
);
```

C#

```
protected List<Problem> SetAnalogOutOutputScaling(
    AnalogInChannel inputChannel,
    AnalogOutSignal analogOutSignal
);
```

Visual Basic

```
Public Function SetAnalogOutOutputScaling(
    inputChannel As AnalogInChannel,
    analogOutSignal As AnalogOutSignal,
) As List<Problem>
```

引数

引数	説明
inputChannel	デフォルトのスケーリング値となる入力チャネル。
analogOutSignal	設定する信号。

戻り値

スケーリング値の割り当て中に発生した問題のリスト。

Device.SetCanBusParameters Method

指定したCanConnectorのすべてのCAN busパラメータ(CanConnector.BitRate、CanConnector.IsTerminated、CanConnector.SupplyVoltageなど)を割り当てますが、チャネルは割り当てません。

C++

```
bool SetCanBusParameters(
    CanConnector canConnector,
    out List<Problem> problems
);
```

C#

```
public abstract bool SetCanBusParameters(
    CanConnector canConnector,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function SetCanBusParameters(
    canConnector As CanConnector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
canConnector	プロパティを割り当てる必要がある既存の(変更された)CanConnector。
problems	アクティブ化/非アクティブ化プロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.SetCanChannelActivation Method

提供されたチャネルのリストをアクティブまたは非アクティブにします。CanInChannelとCanOutChannelのみをアクティブ化または非アクティブ化できます。

C++

```
bool SetCanChannelActivation(
    List<Channel> canChannels,
    bool activate,
    out List<Problem> problems
);
```

C#

```
public abstract bool SetCanChannelActivation(
    List<Channel> canChannels,
    bool activate,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function SetCanChannelActivation(
    canChannels As List<Channel>,
    activate As bool,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
canChannels	アクティブまたは非アクティブにするCANチャネル。
activate	チャネルをアクティブにする場合は`true`、それ以外の場合は`false`。
problems	アクティブ化/非アクティブ化プロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.SetChannelName Method

指定したチャネルにチャネル名を割り当てます。

C++

```
bool SetChannelName(
    Channel channel,
    string channelName,
    out  List<Problem> problems
);
```

C#

```
public abstract bool SetChannelName(
    Channel channel,
    string channelName,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function SetChannelName(
    channel As Channel,
    channelName  As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	変更するチャネルの名。
channelName	新しいチャネル名。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

備考

実装では、チャネル名の割り当てが成功した後、Hbm.Api.Common.Messaging.MessageBroker.RaiseChannelNameChangedAsyncを呼び出す必要があります。

Device.SetDataRateDomain Method

現在のモジュールに指定したdataRateDomainを設定します。重要: デバイスの種類によっては、再起動が自動的に行われます。この場合、デバイスはしばらくの間アクセスできなくなります。

C++

```
bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

C#

```
public abstract bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function SetDataRateDomain(
    dataRateDomain As DataRateDomainType,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
dataRateDomain	モジュールに設定するデータレートドメインタイプ。
problems	データレートドメインの設定中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.SetDeviceName Method

基になる物理デバイスの名前を設定します。派生クラスでオーバーライドする必要があります。

C++

```
bool SetDeviceName(
    string newName,
    out List<Problem> problems
);
```

C#

```
public abstract bool SetDeviceName(
    string newName,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function SetDeviceName(
    newName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
newName	デバイスの新しい名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

備考

実装では、チャネル名の割り当てが成功した後、Hbm.Api.Common.Messaging.MessageBroker.RaiseDeviceNameChangedAsyncを呼び出す必要があります。

Device.SetDigitalIO Method

指定した値に従ってデジタル信号を設定します。

C++

```
void SetDigitalIO(
    DigitalSignal digitalSignal,
    DigitalValueType value
);
```

C#

```
public abstract void SetDigitalIO(
    DigitalSignal digitalSignal,
    DigitalValueType value
);
```

Visual Basic

```
Public abstract Function SetDigitalIO(
    digitalSignal As DigitalSignal,
    value As out DigitalValueType
) As void
```

引数

引数	説明
digitalSignal	調整するデジタル信号。
value	設定する値。

Device.SetFbgBandsDetectionResult Method

AutoDetectFbgBands経由で事前に取得した、指定されたFbgBandsDetectionResultに従って、指定されたFbgConnectorのFbgChannelのFBG関連プロパティ(DetectionRange、FbgSensor.ReferenceWavelength、FbgChannel.IsActivatedなど)を設定します。

C++

```
bool SetFbgBandsDetectionResult(
    FbgConnector fbgConnector,
    FbgBandsDetectionResult fbgBandsDetectionResult,
    out List<Problem> problems
);
```

C#

```
public virtual bool SetFbgBandsDetectionResult(
    FbgConnector fbgConnector,
    FbgBandsDetectionResult fbgBandsDetectionResult,
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function SetFbgBandsDetectionResult(
    fbgConnector As FbgConnector,
    fbgBandsDetectionResult As FbgBandsDetectionResult,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
fbgConnector	FbgBandsDetectionResultを割り当てるコネクタ。
fbgBandsDetectionResult	指定したFbgConnectorに割り当てるFbgBandsDetectionResult。
problems	指定したFbgConnectorのFbgBandsDetectionResultの設定中に発生した問題のリスト。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

備考

センサの種類と信号設定(Signal.SampleRateなど)は変更されないことに注意してください!

Device.SetFbgChannelActivation Method

指定したチャネルのリストをアクティブまたは非アクティブにします。FbgChannelのみをアクティブ化または非アクティブ化できます。

C++

```
bool SetFbgChannelActivation(
    List<FbgChannel> fbgChannels,
    bool activate,
    out List<Problem> problems
);
```

C#

```
public virtual bool SetFbgChannelActivation(
    List<FbgChannel> fbgChannels,
    bool activate,
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function SetFbgChannelActivation(
    fbgChannels As List<FbgChannel>,
    activate As bool,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
fbgChannels	アクティブまたは非アクティブにするFBGチャネル。
activate	チャネルをアクティブにする場合は`true`、それ以外の場合は`false`。
problems	アクティブ化/非アクティブ化プロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.SetFbgChannelDetectionRange Method

指定した値に従って、指定されたFbgChannelのFbgChannel.DetectionRangeを設定します。

C++

```
bool SetFbgChannelDetectionRange(
    FbgChannel fbgChannel,
    double wavelengthLow,
    double wavelengthHigh,
    out List<Problem> problems
);
```

C#

```
public virtual bool SetFbgChannelDetectionRange(
    FbgChannel fbgChannel,
    double wavelengthLow,
    double wavelengthHigh,
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function SetFbgChannelDetectionRange(
    fbgChannel As FbgChannel,
    wavelengthLow As double,
    wavelengthHigh As double,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
fbgChannel	検出範囲を設定するチャネル。
wavelengthLow	新しい低波長ボーダー。
wavelengthHigh	新しい高波長ボーダー。
problems	検出範囲の設定中に発生した警告とエラー。

戻り値

エラーが発生していない場合はtrue、それ以外の場合はfalse。

Device.SetFbgDetectionThreshold Method

指定した値に従って、指定したFbgConnectorのFbgConnector.Detection Thresholdを設定します。

C++

```
bool SetFbgDetectionThreshold(
    FbgConnector fbgConnector,
    double detectionThreshold,
    out List<Problem> problems
);
```

C#

```
public virtual bool SetFbgDetectionThreshold(
    FbgConnector fbgConnector,
    double detectionThreshold,
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function SetFbgDetectionThreshold(
    fbgConnector As FbgConnector,
    detectionThreshold As double,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
fbgConnector	検出しきい値を設定するFbgConnector。
detectionThreshold	設定する新しい検出しきい値。MXFSの場合は3dBのみ可能です。
problems	検出しきい値の設定中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.SetInputSignalReference Method

入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。

C++

```
bool SetInputSignalReference(
    IInputSignalReference channel,
    Device inputSignalDevice,
    Signal inputSignal,
    out List<Problem> problems
);
```

C#

```
public abstract bool SetInputSignalReference(
    IInputSignalReference channel,
    Device inputSignalDevice,
    Signal inputSignal,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function SetInputSignalReference(
    channel As IInputSignalReference,
    inputSignalDevice As Device,
    inputSignal As Signal,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	入力信号リファレンスが含まれる(出力)チャネル。
inputSignalDevice	入力信号のデバイス。
inputSignal	入力信号。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

SetShunt Method

これは、SetShuntメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	Device.SetShunt (Connector, ShuntMode, out List<Problem>)	指定したコネクタのシャント出力の構成を設定します。
	Device.SetShunt (List<Connector>, ShuntMode, out List<Problem>)	指定したコネクタのシャント出力の構成を設定します。

Device.SetShunt Method (Connector, ShuntMode, out List<Problem>)

指定したコネクタのシャント出力の構成を設定します。

C++

```
bool SetShunt(
    Connector connector,
    ShuntMode mode,
    out List<Problem> problems
);
```

C#

```
public abstract bool SetShunt(
    Connector connector,
    ShuntMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function SetShunt(
    connector As Connector,
    mode As ShuntMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	シャント出力を設定するコネクタ。
mode	指定されたコネクタのShuntMode。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.SetShunt Method (List<Connector>, ShuntMode, out List<Problem>)

指定したコネクタのシャント出力の構成を設定します。

C++

```
bool SetShunt(
List<Connector> connectors,
    ShuntMode mode,
    out List<Problem> problems
);
```

C#

```
public abstract bool SetShunt(
List<Connector> connectors,
    ShuntMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function SetShunt(
    connectors As List<Connector>,
    mode As ShuntMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connectors	シャント出力を設定するコネクタ。
mode	指定されたコネクタのShuntMode。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.SetSignalName Method

指定した信号に信号名を割り当てます。

C++

```
bool SetSignalName(
    Signal signal,
    string signalName,
    out List<Problem> problems
);
```

C#

```
public abstract bool SetSignalName(
    Signal signal,
    string signalName,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function SetSignalName(
    signal As Signal,
    signalName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
signal	名前を変更する信号。
signalName	新しい信号の名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

実装では、シグナル名の割り当てが成功した後、Hbm.Api.Common.Messaging.MessageBroker.RaiseSignalNameChangedAsyncを呼び出す必要があります。

Device.SetTedsUsageMode Method

TEDS使用モードを設定し、コネクタ設定を再読み込みします。

C++

```
bool SetTedsUsageMode(
    Connector connector,
    TedsUsageMode tedsUsageMode,
    out List<Problem> problems
);
```

C#

```
public virtual bool SetTedsUsageMode(
    Connector connector,
    TedsUsageMode tedsUsageMode,
    out List<Problem> problems
);
```

Visual Basic

```
Public virtual Function SetTedsUsageMode(
    connector As Connector,
    tedsUsageMode As TedsUsageMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	コネクタ。
tedsUsageMode	設定するTEDSの使用モード。
problems	出力: 問題のリスト、または問題が発生しなかった場合は空です。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device.SetUnit Method

指定したチャネルの単位を割り当てます。

C++

```
bool SetUnit(
    Channel channel,
    string unit,
    out  List<Problem> problems
);
```

C#

```
public abstract bool SetUnit(
    Channel channel,
    string unit,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function SetUnit(
    channel As Channel,
    unit  As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	単位を設定するチャネル。
unit	割り当てる単位。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Device Class -> Device Methods

SetZeroBalance Method

これは、SetZeroBalanceメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	Device.SetZeroBalance (Channel, out List<Problem>, int)	指定したチャネルの現在のZero.Targetを使用して、Zero.Offsetを新たに計算することにより、ゼロバランスを実行します。
	Device.SetZeroBalance (List<Channel>, out List<Problem>, int)	指定したチャネルの現在のZero.Targetを使用して、Zero.Offsetを新たに計算することにより、ゼロバランスを実行します。

Device.SetZeroBalance Method (Channel, out List<Problem>, int)

指定したチャネルの現在のZero.Targetを使用して、Zero.Offsetを新たに計算することにより、ゼロバランスを実行します。

C++

```
bool  SetZeroBalance(
    Channel channel,
    out  List<Problem> problems,
    int numberOfMeasurementValuesToUse = 1
);
```

C#

```
public bool SetZeroBalance(
    Channel channel,
    out  List<Problem> problems,
    int numberOfMeasurementValuesToUse
);
```

Visual Basic

```
Public Function SetZeroBalance(
    channel As Channel,
    problems As out List<Problem>,
    numberOfMeasurementValuesToUse  As Integer = 1
) As bool
```

引数

引数	説明
channel	ゼロバランスを設定するチャネル。
problems	ゼロバランスの実行中に発生した警告とエラー。
numberOfMeasurementValuesToUse	ゼロバランス(平均化)に使用する連続測定の数(>0、デフォルトは1)。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

備考

ゼロバランスを取ることができるのは、有効な測定値を持つチャネルのみです。

例外

引数	説明
System.ArgumentNullException	チャネルが`null`です。
System.ArgumentException	numberOfMeasurementValuesToUse が0より小さいまたは同じです。

Device.SetZeroBalance Method (List<Channel>, out List<Problem>, int)

指定したチャネルの現在のZero.Targetを使用して、Zero.Offsetを新たに計算することにより、ゼロバランスを実行します。

C++

```
bool  SetZeroBalance(
    List<Channel> channels,
    out  List<Problem> problems,
    int numberOfMeasurementValuesToUse = 1
);
```

C#

```
public bool SetZeroBalance(
    List<Channel> channels,
    out  List<Problem> problems,
    int numberOfMeasurementValuesToUse
);
```

Visual Basic

```
Public Function SetZeroBalance(
    channels As List<Channel>,
    problems As out List<Problem>,
    numberOfMeasurementValuesToUse  As Integer = 1
) As bool
```

引数

引数	説明
channel	ゼロバランスを設定するチャネル。
problems	ゼロバランスの実行中に発生した警告とエラー。
numberOfMeasurementValuesToUse	ゼロバランス(平均化)に使用する連続測定の数(>0、デフォルトは1)。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

備考

ゼロバランスを取ることができるのは、有効な測定値を持つチャネルのみです。

例外

引数	説明
System.ArgumentNullException	チャネルが`null`です。
System.ArgumentException	numberOfMeasurementValuesToUse が`0`より小さいまたは同じです。

Device.StoreStatusIfChanged Method

ステータスがリストに保存されていない場合は保存し、変更された場合は置き換えます。

C++

```
bool StoreStatusIfChanged(
    DeviceStatus status
);
```

C#

```
protected internal bool StoreStatusIfChanged(
    DeviceStatus status
);
```

Visual Basic

```
Protected Friend Function StoreStatusIfChanged(
    status As DeviceStatus,
) As bool
```

引数

引数	説明
status	ステータス。

戻り値

ステータスが新しいか変更された場合は`true`、それ以外の場合は`false`。

Device.SubscribeToStatusMonitoring Method

指定したチャネルをステータス監視に登録します。

C++

```
bool SubscribeToStatusMonitoring(
    StatusType monitoredStatusTypes,
    List<Channel> channels = null
);
```

C#

```
public bool SubscribeToStatusMonitoring(
    StatusType monitoredStatusTypes,
    List<Channel> channels
);
```

Visual Basic

```
Public Function SubscribeToStatusMonitoring(
    monitoredStatusTypes As StatusType,
    channels  As List<Channel> = null
) As bool
```

引数

引数	説明
monitoredStatusTypes	監視対象のステータスの種類。
channels	アナログ入力またはFBGチャネル、またはすべてのチャネルを予約する場合は`null`。

戻り値

登録が成功したは`true`、それ以外の場合は`false`(デバイス接続されていないなど)。

備考

連続する登録の呼び出しで異なるmonitoredStatusTypesを使用すると、このデバイスのすべての登録済みチャネルに対して発生するステータスタイプイベントのリストに追加されるだけです。

Device.UnsubscribeFromStatusMonitoring Method

指定したチャネルをステータス監視から登録解除します。

C++

```
void UnsubscribeFromStatusMonitoring(
    List<Channel> channels = null
);
```

C#

```
public void UnsubscribeFromStatusMonitoring(
    List<Channel> channels
);
```

Visual Basic

```
Public Function UnsubscribeFromStatusMonitoring(
    channels  As List<Channel> = null
) As void
```

引数

引数	説明
channels	登録を解除するチャネル、またはデバイス(およびそのすべてのチャネル)を解除する場合は`null`。

Device.WriteTeds Method

指定したコネクタに接続されたTEDSに指定したセンサを書き込みます。

C++

```
bool WriteTeds(
    Connector connector,
    out List<Problem> problems,
    Sensor sensor,
    string tid = null,
    string channelName = null
);
```

C#

```
public abstract bool WriteTeds(
    Connector connector,
    out List<Problem> problems,
    Sensor sensor,
    string tid,
    string channelName
);
```

Visual Basic

```
Public abstract Function WriteTeds(
    connector As Connector,
    problems As out List<Problem>,
    sensor  As Sensor,
    tid  As string = null,
    channelName  As string = null
) As bool
```

引数

引数	説明
connector	コネクタ。
problems	問題。
sensor	センサ(sensor.Descriptionは使用しません。パラメータchannelNameを参照)。
tid	TEDS IDはオプションです。
channelName	オプション。TEDSでHBM Channel Nameテンプレートを作成するチャネルの名前。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

Device Properties

これは、Deviceクラスのプロパティの一覧です。

Public Properties

	名前	説明
	CalCertSyncLock	基礎となるデバイスの校正証明書の伝達への並列アクセスを同期するために使用します。
	ConnectionInfo	物理デバイスへの接続方法に関する情報。
	Connectors	すべてのコネクタのリスト。
	DataAcquisitionMode	測定に使用するデータ収集モードを保持します。このモードは通常、DaqMeasurement.StartDaq(Enums.DataAcquisitionMode, int)中に呼び出されるDeviceFamily.PrepareDataAcquisition(Enums.DataAcquisitionMode,List{Device})によって設定されます。特にEnums.DataAcquisitionMode.Autoの場合、このプロパティは、測定の開始後に選択したデータ取得モードを返します。このプロパティを手動で設定しないでください! 内部使用のみを目的としています。
	DataRateDomain	データレートドメインを取得または設定します。
A	FamilyName	このデバイスが属するファミリの名前。
	FbgFiberInfo	ファイバセンシングベースのデバイスの帯域情報を保持します。これは、ファイバセンシングベースのデバイスの場合にのみ入力します。
	FirmwareVersion	ファームウェアのバージョン。
	HardwareVersion	ハードウェアのバージョン。
R V	HasAdditionalTedsFieldsSupport	デバイスファミリがキャリブレーションおよびシリアル番号のTEDSフィールドをサポートしているかどうかを示す値を取得します。
	IsAnalogOutChannelDisablingSupported	デバイスがアナログ出力チャネルの無効化をサポートしているかどうかを示す値を取得または設定します。
R V	IsAutomaticTedsMonitoringSupported	このインスタンスが自動TEDS監視をサポートしているかどうかを示す値を取得します。
	IsCanBusSupplyVoltageSupported	デバイスがCAN busの供給電圧をサポートしているかどうかを示す値を取得または設定します。
	IsCanBusTerminationSupported	デバイスがCAN busターミネーションをサポートしているかどうかを示す値を取得または設定します。
	IsConnectable	デバイスが接続可能かどうかを示します。
	IsConnected	デバイスが接続されているかどうかを示します。
R V	IsElectricalSnapshotSupported	ReadSingleElectricalValue(AnalogInSignal)関数を使用して、このインスタンスが電気値の要求をサポートしているかどうかを示す値を取得します。
	IsEnablingRoutingSupported	デバイスがルーティング機能の有効化/無効化をサポートしているかどうかを示す値を取得または設定します。
V	IsFirmwareUpdateSupported	このインスタンスがファームウェアの更新をサポートしているかどうかを示す値を取得または設定します。
	IsIpChangeSupported	デバイスがIPアドレスの変更をサポートしているかどうかを示す値を取得または設定します。
	IsLedFlashSupported	デバイスがLEDの点滅をサポートしているかどうかを示す値を取得または設定します。
	IsReadingDaqValues	デバイスが測定値を更新中かどうかを示します。
R V	IsReadingFbgSpectrumSupported	このインスタンスがスペクトルの読み取りをサポートしているかどうかを示す値を取得します。
	IsRoutingEnabled	デバイスのルーティング機能が現在有効になっているかどうかを示す値を取得または設定します。
	IsUnsupportedModel	特定のモデルがAPIでサポートされているかどうかを示します。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> Device Class

	名前	説明
	IsUsingHbmClassicSampleRates	デバイスが10進数のサンプル レート(10Hz、100Hz、1000Hz など)ではなく、従来のHBMサンプル レート(1200Hz、2400Hz、4800Hz)など) を使用しているかどうかを示します。このプロパティはデバイスの接続後に有効になります。現在、QuantumXデバイス ファミリのみが10進数のサンプルレートをサポートしています。
	MinimumFirmwareVersionSupported	ドライバがサポートする最小ファームウェア バージョン。
	Model	デバイスモデルのタイプ。
R	MonitoredStatusTypes	監視対象のステータスの種類を取得します。
	Name	デバイス名。
R	NtpStartDate	秒数のカウントの開始日を協定世界時(UTC)で決定する日付。タイムスタンプは、この日付/時刻からの秒数で表されます。
S	Overflow	無効な測定値(オーバーフロー)の代わりに使用する値を定義します。ステータスが無効な各測定値については、対応する測定値がこのオーバーフロー値に設定されます。DaqEnvironment.Overflow を使用してこの値を設定します(デフォルトは1000000)。
	ProtectionType	デバイスの保護等級タイプ(例: IP55)。
	SerialNo	デバイスのシリアル番号。
	SupportedDataRateDomains	サポートしているデータレートドメインを取得または設定します。
	SyncLock	基礎となるデバイスへの並列アクセスを同期するために使用します。
	SyncMode	測定開始時のデバイスの役割を定義するモード。
	Timeout	デバイスとの通信のタイムアウト。
	TimeSource	物理デバイスの時刻を同期する方法に関する情報。

Device.CalCertSyncLock Property

基礎となるデバイスの校正証明書の伝達への並列アクセスを同期するために使用します。

C++

```
[JsonIgnore]
__property object CalCertSyncLock;
```

C#

```
[JsonIgnore]
public object CalCertSyncLock;
```

Visual Basic

```
<JsonIgnore>
Public Property CalCertSyncLock() As object
```

Device.ConnectionInfo Property

物理デバイスへの接続方法に関する情報。

C++

```
__property ConnectionInfo ConnectionInfo;
```

C#

```
public ConnectionInfo ConnectionInfo;
```

Visual Basic

```
Public Property ConnectionInfo() As ConnectionInfo
```

Device.Connectors Property

すべてのコネクタのリスト。

C++

```
[IgnoreInTms]
__property List<Connector> Connectors;
```

C#

```
[IgnoreInTms]
public List<Connector> Connectors;
```

Visual Basic

```
<IgnoreInTms>
Public Property Connectors() As List<Connector>
```

Device.DataAcquisitionMode Property

測定に使用するデータ収集モードを保持します。このモードは通常、DaqMeasurement.StartDaq(Enums.DataAcquisitionMode, int)中に呼び出されるDeviceFamily.PrepareDataAcquisition(Enums.DataAcquisitionMode,List{Device})によって設定されます。特にEnums.DataAcquisitionMode.Autoの場合、このプロパティは、測定の開始後に選択したデータ取得モードを返します。このプロパティを手動で設定しないでください! 内部使用のみを目的としています。

C++

```
__property DataAcquisitionMode DataAcquisitionMode;
```

C#

```
public DataAcquisitionMode DataAcquisitionMode;
```

Visual Basic

```
Public Property DataAcquisitionMode() As DataAcquisitionMode
```

Device.DataRateDomain Property

データレートドメインを取得または設定します。

C++

```
__property DataRateDomainType DataRateDomain;
```

C#

```
public DataRateDomainType DataRateDomain;
```

Visual Basic

```
Public Property DataRateDomain() As DataRateDomainType
```

説明

データレートドメイン。

Device.FamilyName Property

このデバイスが属するファミリの名前。

C++

```
__property string FamilyName;
```

C#

```
public abstract string FamilyName;
```

Visual Basic

```
Public abstract Property FamilyName() As string
```

備考

これは、DeviceFamilyクラスのFamilyNameプロパティとまったく同じ文字列でなければなりません。

Device.FbgFiberInfo Property

ファイバセンシングベースのデバイスの帯域情報を保持します。これは、ファイバセンシングベースのデバイスの場合にのみ入力します。

C++

```
__property FbgFiberInfo FbgFiberInfo;
```

C#

```
public FbgFiberInfo FbgFiberInfo;
```

Visual Basic

```
Public Property FbgFiberInfo() As FbgFiberInfo
```

Device.FirmwareVersion Property

ファームウェアのバージョン。

C++

```
__property string FirmwareVersion;
```

C#

```
public string FirmwareVersion;
```

Visual Basic

```
Public Property FirmwareVersion() As string
```

Device.HardwareVersion Property

ハードウェアのバージョン。

C++

```
__property string HardwareVersion;
```

C#

```
public string HardwareVersion;
```

Visual Basic

```
Public Property HardwareVersion() As string
```

Device.HasAdditionalTedsFieldsSupport Property

デバイスファミリがキャリブレーションおよびシリアル番号のTEDSフィールドをサポートしているかどうかを示す値を取得します。

C++

```
[JsonIgnore]
__property bool HasAdditionalTedsFieldsSupport;
```

C#

```
[JsonIgnore]
public virtual bool HasAdditionalTedsFieldsSupport;
```

Visual Basic

```
<JsonIgnore>
Public virtual Property HasAdditionalTedsFieldsSupport() As bool
```

Device.IsAnalogOutChannelDisablingSupported Property

デバイスがアナログ出力チャネルの無効化をサポートしているかどうかを示す値を取得または設定します。

C++

```
__property bool IsAnalogOutChannelDisablingSupported;
```

C#

```
public bool IsAnalogOutChannelDisablingSupported;
```

Visual Basic

```
Public Property IsAnalogOutChannelDisablingSupported() As bool
```

説明

このインスタンスがアナログ出力の無効化をサポートしている場合は`true`。それ以外の場合は`false`。

備考

手動で設定しないでください。このプロパティは、デバイスの接続時に設定されます。

Device.IsAutomaticTedsMonitoringSupported Property

このインスタンスが自動TEDS監視をサポートしているかどうかを示す値を取得します。

C++

```
[JsonIgnore]
__property bool IsAutomaticTedsMonitoringSupported;
```

C#

```
[JsonIgnore]
public virtual bool IsAutomaticTedsMonitoringSupported;
```

Visual Basic

```
<JsonIgnore>
Public virtual Property IsAutomaticTedsMonitoringSupported() As bool
```

Device.IsCanBusSupplyVoltageSupported Property

デバイスがCAN busの供給電圧をサポートしているかどうかを示す値を取得または設定します。

C++

```
__property bool IsCanBusSupplyVoltageSupported;
```

C#

```
public bool IsCanBusSupplyVoltageSupported;
```

Visual Basic

```
Public Property IsCanBusSupplyVoltageSupported() As bool
```

説明

このインスタンスがCAN busの電源電圧をサポートしている場合は`true`。それ以外の場合は`false`。

備考

手動で設定しないでください。このプロパティは、デバイスの接続時に設定されます。

Device.IsCanBusTerminationSupported Property

デバイスがCAN busターミネーションをサポートしているかどうかを示す値を取得または設定します。

C++

```
__property bool IsCanBusTerminationSupported;
```

C#

```
public bool IsCanBusTerminationSupported;
```

Visual Basic

```
Public Property IsCanBusTerminationSupported() As bool
```

説明

このインスタンスがCAN busターミネーションをサポートしている場合は`true`。それ以外の場合は`false`。

備考

手動で設定しないでください。このプロパティは、デバイスの接続時に設定されます。

Device.IsConnectable Property

デバイスが接続可能かどうかを示します。

C++

```
__property bool IsConnectable;
```

C#

```
public bool IsConnectable;
```

Visual Basic

```
Public Property IsConnectable() As bool
```

Device.IsConnected Property

デバイスが接続されているかどうかを示します。

C++

```
[JsonIgnore]
__property bool IsConnected;
```

C#

```
[JsonIgnore]
public bool IsConnected;
```

Visual Basic

```
<JsonIgnore>
Public Property IsConnected() As bool
```

Device.IsElectricalSnapshotSupported Property

ReadSingleElectricalValue(AnalogInSignal)関数を使用して、このインスタンスが電気値の要求をサポートしているかどうかを示す値を取得します。

C++

```
[JsonIgnore]
[Obsolete("Do not use this property- it is for temporary, internal use only and is subject to change without notice!")]
[Browsable(false)]
[EditorBrowsable(EditorBrowsableState.Never)]
__property bool IsElectricalSnapshotSupported;
```

C#

```
[JsonIgnore]
[Obsolete("Do not use this property- it is for temporary, internal use only and is subject to change without notice!")]
[Browsable(false)]
[EditorBrowsable(EditorBrowsableState.Never)]
public virtual bool IsElectricalSnapshotSupported;
```

Visual Basic

```
<JsonIgnore>
<Obsolete("Do not use this property- it is for temporary, internal use only and is subject to change without notice!")>
<Browsable(false)>
<EditorBrowsable(EditorBrowsableState.Never)>
Public virtual Property IsElectricalSnapshotSupported() As bool
```

Device.IsEnablingRoutingSupported Property

デバイスがルーティング機能の有効化/無効化をサポートしているかどうかを示す値を取得または設定します。

C++

```
__property bool IsEnablingRoutingSupported;
```

C#

```
public bool IsEnablingRoutingSupported;
```

Visual Basic

```
Public Property IsEnablingRoutingSupported() As bool
```

説明

このインスタンスがルーティング機能を有効または無効にできる場合は`true`。それ以外の場合は`false`。

備考

手動で設定しないでください。このプロパティは、デバイスの接続時に設定されます。

Device.IsFirmwareUpdateSupported Property

このインスタンスがファームウェアの更新をサポートしているかどうかを示す値を取得または設定します。

C++

```
__property bool IsFirmwareUpdateSupported;
```

C#

```
public virtual bool IsFirmwareUpdateSupported;
```

Visual Basic

```
Public virtual Property IsFirmwareUpdateSupported() As bool
```

説明

このインスタンスがファームウェアのアップデートをサポートする場合は`true`。それ以外の場合は`false`。

Device.IsIpChangeSupported Property

デバイスがIPアドレスの変更をサポートしているかどうかを示す値を取得または設定します。

C++

```
__property bool IsIpChangeSupported;
```

C#

```
public bool IsIpChangeSupported;
```

Visual Basic

```
Public Property IsIpChangeSupported() As bool
```

説明

このインスタンスがIP変更をサポートしている場合は`true`。それ以外の場合は`false`。

Device.IsLedFlashSupported Property

デバイスがLEDの点滅をサポートしているかどうかを示す値を取得または設定します。

C++

```
__property bool IsLedFlashSupported;
```

C#

```
public bool IsLedFlashSupported;
```

Visual Basic

```
Public Property IsLedFlashSupported() As bool
```

Device.IsReadingDaqValues Property

デバイスが測定値を更新中かどうかを示します。

C++

```
[JsonIgnore]
__property bool IsReadingDaqValues;
```

C#

```
[JsonIgnore]
public bool IsReadingDaqValues;
```

Visual Basic

```
<JsonIgnore>
Public Property IsReadingDaqValues() As bool
```

備考

これは、デバイスから値を読み取る新しいタスクを開始するかどうかを決定するために内部的に使用します。デバイスがDAQ値を読み取り中の場合、新しいタスクは開始されません。

Device.IsReadingFbgSpectrumSupported Property

このインスタンスがスペクトルの読み取りをサポートしているかどうかを示す値を取得します。

C++

```
[JsonIgnore]
__property bool IsReadingFbgSpectrumSupported;
```

C#

```
[JsonIgnore]
public virtual bool IsReadingFbgSpectrumSupported;
```

Visual Basic

```
<JsonIgnore>
Public virtual Property IsReadingFbgSpectrumSupported() As bool
```

Device.IsRoutingEnabled Property

デバイスのルーティング機能が現在有効になっているかどうかを示す値を取得または設定します。

C++

```
__property bool IsRoutingEnabled;
```

C#

```
public bool IsRoutingEnabled;
```

Visual Basic

```
Public Property IsRoutingEnabled() As bool
```

説明

このインスタンスで現在ルーティング機能が有効になっている場合は`true`。それ以外の場合は`false`。

備考

手動で設定しないでください。このプロパティは、デバイスの接続時に設定されます。

Device.IsUnsupportedModel Property

特定のモデルがAPIでサポートされているかどうかを示します。

C++

```
__property bool IsUnsupportedModel;
```

C#

```
public bool IsUnsupportedModel;
```

Visual Basic

```
Public Property IsUnsupportedModel() As bool
```

Device.IsUsingHbmClassicSampleRates Property

デバイスが10進数のサンプル レート(10Hz、100Hz、1000Hzなど)ではなく、従来のHBMサンプル レート(1200Hz、2400Hz、4800Hz)など) を使用しているかどうかを示します。このプロパティはデバイスの接続後に有効になります。現在、QuantumXデバイスファミリのみが10進数のサンプルレートをサポートしています。

C++

```
[JsonIgnore]
[Obsolete("Please use SupportedDataRateDomains instead!")]
__property bool IsUsingHbmClassicSampleRates;
```

C#

```
[JsonIgnore]
[Obsolete("Please use SupportedDataRateDomains instead!")]
public bool IsUsingHbmClassicSampleRates;
```

Visual Basic

```
<JsonIgnore>
<Obsolete("Please use SupportedDataRateDomains instead!")>
Public Property IsUsingHbmClassicSampleRates() As bool
```

Device.MinimumFirmwareVersionSupported Property

ドライバがサポートする最小ファームウェア バージョン。

C++

```
[JsonIgnore]
__property string MinimumFirmwareVersionSupported;
```

C#

```
[JsonIgnore]
public string MinimumFirmwareVersionSupported;
```

Visual Basic

```
<JsonIgnore>
Public Property MinimumFirmwareVersionSupported() As string
```

Device.Model Property

デバイスモデルのタイプ。

C++

```
__property string Model;
```

C#

```
public string Model;
```

Visual Basic

```
Public Property Model() As string
```

Device.MonitoredStatusTypes Property

監視対象のステータスの種類を取得します。

C++

```
[JsonIgnore]
__property StatusType MonitoredStatusTypes;
```

C#

```
[JsonIgnore]
public StatusType MonitoredStatusTypes;
```

Visual Basic

```
<JsonIgnore>
Public Property MonitoredStatusTypes() As StatusType
```

Device.Name Property

デバイス名。

C++

```
__property string Name;
```

C#

```
public string Name;
```

Visual Basic

```
Public Property Name() As string
```

Device.NtpStartDate Property

秒数のカウントの開始日を協定世界時(UTC)で決定する日付。タイムスタンプは、この日付/時刻からの秒数で表されます。

C++

```
__property DateTime NtpStartDate;
```

C#

```
public DateTime NtpStartDate;
```

Visual Basic

```
Public Property NtpStartDate() As DateTime
```

Device.Overflow Property

無効な測定値(オーバーフロー)の代わりに使用する値を定義します。ステータスが無効な各測定値については、対応する測定値がこのオーバーフロー値に設定されます。DaqEnvironment.Overflowを使用してこの値を設定します(デフォルトは1000000)。

C++

```
[JsonIgnore]
__property double Overflow;
```

C#

```
[JsonIgnore]
public static double Overflow;
```

Visual Basic

```
<JsonIgnore>
Public static Property Overflow() As double
```

備考

この値は、DaqEnvironmentのコンストラクタによってデフォルト値に設定されます。

Device.ProtectionType Property

デバイスの保護等級タイプ(例: IP55)。

C++

```
__property string ProtectionType;
```

C#

```
public string ProtectionType;
```

Visual Basic

```
Public Property ProtectionType() As string
```

Device.SerialNo Property

デバイスのシリアル番号。

C++

```
__property string SerialNo;
```

C#

```
public string SerialNo;
```

Visual Basic

```
Public Property SerialNo() As string
```

Device.SupportedDataRateDomains Property

サポートしているデータレートドメインを取得または設定します。

C++

```
[JsonIgnore]
__property List<DataRateDomainType> SupportedDataRateDomains;
```

C#

```
[JsonIgnore]
public List<DataRateDomainType> SupportedDataRateDomains;
```

Visual Basic

```
<JsonIgnore>
Public Property SupportedDataRateDomains() As List<DataRateDomainType>
```

説明

サポートしているデータレートドメイン。

Device.SyncLock Property

基礎となるデバイスへの並列アクセスを同期するために使用します。

C++

```
[JsonIgnore]
__property object SyncLock;
```

C#

```
[JsonIgnore]
public object SyncLock;
```

Visual Basic

```
<JsonIgnore>
Public Property SyncLock() As object
```

Device.SyncMode Property

測定開始時のデバイスの役割を定義するモード。

C++

```
__property SyncModeType SyncMode;
```

C#

```
public SyncModeType SyncMode;
```

Visual Basic

```
Public Property SyncMode() As SyncModeType
```

Device.Timeout Property

デバイスとの通信のタイムアウト。

C++

```
__property int Timeout;
```

C#

```
public int Timeout;
```

Visual Basic

```
Public Property Timeout() As Integer
```

Device.TimeSource Property

物理デバイスの時刻を同期する方法に関する情報。

C++

```
__property TimeSource TimeSource;
```

C#

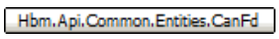
```
public TimeSource TimeSource;
```

Visual Basic

```
Public Property TimeSource() As TimeSource
```

CanFd Class

CAN FDの構成プロパティ。



C++

```
class CanFd;
```

C#

```
public class CanFd;
```

Visual Basic

```
Public Class CanFd
```

ファイル

CanFd.cs

CanFd Members

次の表は、CAN FDによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	BitRate	CAN FD Busのビットレート[bit/s]。従来のCANビットレートの倍数である必要があります。
	SamplePointRatio	サンプルポイント比を取得または設定します。
	SynchronizationJumpWidth	同期ジャンプ幅(SJW)を取得または設定します。

CanFd Properties

CanFdクラスのプロパティの一覧です。

Public Properties

	名前	説明
	BitRate	CAN FD Busのビットレート[bit/s]。従来のCANビットレートの倍数である必要があります。
	SamplePointRatio	サンプルポイント比を取得または設定します。
	SynchronizationJumpWidth	同期ジャンプ幅(SJW)を取得または設定します。

CanFd.BitRate Property

CAN FD Busのビットレート[bit/s]。従来のCANビットレートの倍数である必要があります。

C++

```
__property int BitRate;
```

C#

```
public int BitRate;
```

Visual Basic

```
Public Property BitRate() As Integer
```

CanFd.SamplePointRatio Property

サンプルポイント比を取得または設定します。

C++

```
__property decimal SamplePointRatio;
```

C#

```
public decimal SamplePointRatio;
```

Visual Basic

```
Public Property SamplePointRatio() As decimal
```

説明

サンプルポイント比。

CanFd.SynchronizationJumpWidth Property

同期ジャンプ幅(SJW)を取得または設定します。

C++

```
__property int SynchronizationJumpWidth;
```

C#

```
public int SynchronizationJumpWidth;
```

Visual Basic

```
Public Property SynchronizationJumpWidth() As Integer
```

DeviceFamily Class

すべてのDeviceFamily実装の抽象基底クラス。



C++

```
class DeviceFamily : public IDisposable;
```

C#

```
public abstract class DeviceFamily : public IDisposable;
```

Visual Basic

```
Public abstract Class DeviceFamily
Inherits IDisposable
```

ファイル

DeviceFamily.cs

DeviceFamily Members

次の表は、DeviceFamilyによって公開されるメンバーの一覧です。

Private Methods

	名前	説明
	~DeviceFamily	ファイナライザ。すべてのリソースをクリーンアップします。

Public Methods

	名前	説明
	DeviceFamily	コンストラクタ。新しいインスタンスを作成します。
	Dispose	すべてのリソースをクリーンアップします。
	Find	指定したconnectionInfosに従ってデバイスを検索します。
	GetPossibleDataAcquisitionModes	ファミリ内のすべての指定したデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSource は、ファミリのすべてのデバイスで使用するタイムソースに設定されます(指定されたデバイスで DataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関係します)。
	PrepareDataAcquisition	指定したDataAcquisitionModeに従ってデバイスの最終設定を行う関数。この機能を使用して、たとえばトリガマスタとトリガスレーブを設定したり、特定のデータ収集モードに関連する特定のタイムスタンプ設定などを選択したりできます。測定に参加する指定したデバイスファミリのすべてのデバイスにもこの機能が与えられるため、特定のデバイスを異なる方法で処理する可能性があります(例: トリガマスタがスレーブとは異なる設定/準備を取得するなど)。
	Scan	特定のデバイスファミリのデバイスのネットワークスキャンを実行します。派生クラスでオーバーライドする必要があります。

Public Properties

	名前	説明
	FamilyName	デバイスファミリの一意の名前。派生クラスでオーバーライドする必要があります。
	IsScanSupported	デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。
	SkillReader	SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

DeviceFamily.~DeviceFamily Destructor

ファイナライザ。すべてのリソースをクリーンアップします。

C++

```
~DeviceFamily();
```

C#

```
private ~DeviceFamily();
```

Visual Basic

```
Private Sub Finalize()
```

DeviceFamily.DeviceFamily Constructor

コンストラクタ。新しいインスタンスを作成します。

C++

```
DeviceFamily();
```

C#

```
public DeviceFamily();
```

Visual Basic

```
Public Sub DeviceFamily()
```

DeviceFamily Methods

DeviceFamilyクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Dispose	すべてのリソースをクリーンアップします。
	Find	指定したconnectionInfosに従ってデバイスを検索します。
	GetPossibleDataAcquisitionModes	ファミリ内のすべての指定したデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSource は、ファミリのすべてのデバイスで使用するタイムソースに設定されます(指定されたデバイスで DataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関係します)。
	PrepareDataAcquisition	指定したDataAcquisitionModeに従ってデバイスの最終設定を行う関数。この機能を使用して、たとえばトリガマスタとトリガスレーブを設定したり、特定のデータ収集モードに関連する特定のタイムスタンプ設定などを選択したりできます。測定に参加する指定したデバイスファミリのすべてのデバイスにもこの機能が与えられるため、特定のデバイスを異なる方法で処理する可能性があります(例: トリガマスタがスレーブとは異なる設定/準備を取得するなど)。
	Scan	特定のデバイスファミリのデバイスのネットワークスキャンを実行します。派生クラスでオーバーライドする必要があります。

DeviceFamily.Dispose Method

すべてのリソースをクリーンアップします。

C++

```
void Dispose(
);
```

C#

```
public void Dispose(
);
```

Visual Basic

```
Public Function Dispose() As void
```

DeviceFamily.Find Method

指定したconnectionInfosに従ってデバイスを検索します。

C++

```
List<Device> Find(
    List<ConnectionInfo> connectionInfos,
    out List<Problem> problems
);
```

C#

```
public abstract List<Device> Find(
    List<ConnectionInfo> connectionInfos,
    out List<Problem> problems
);
```

Visual Basic

```
Public abstract Function Find(
    connectionInfos As List<ConnectionInfo>,
    problems As out List<Problem>
) As List<Device>
```

引数

引数	説明
connectionInfos	接続情報。
problems	検索プロセス中に発生した警告とエラー。

戻り値

検出されたすべてのデバイスのリスト。

DeviceFamily.GetPossibleDataAcquisitionModes Method

ファミリ内のすべての指定したデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSource は、ファミリのすべてのデバイスで使用するタイムソースに設定されます(指定されたデバイスで DataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関係します)。

C++

```
List<DataAcquisitionMode> GetPossibleDataAcquisitionModes(
    List<Device> devices,
    out TimeSource commonTimeSource
);
```

C#

```
public abstract List<DataAcquisitionMode> GetPossibleDataAcquisitionModes(
    List<Device> devices,
    out TimeSource commonTimeSource
);
```

Visual Basic

```
Public abstract Function GetPossibleDataAcquisitionModes(
    devices As List<Device>,
    commonTimeSource As out TimeSource
) As List<DataAcquisitionMode>
```

引数

引数	説明
devices	ファミリに属しており、測定に参加するデバイス。
commonTimeSource	すべてのデバイスでサポートされているTimeSource(通常はNtpTimeSourceなど)。

戻り値

指定したデバイスを考慮した、可能なデータ取得モード。

DeviceFamily.PrepareDataAcquisition Method

指定したDataAcquisitionModeに従ってデバイスの最終設定を行う関数。この機能を使用して、たとえばトリガマスタとトリガスレーブを設定したり、特定のデータ収集モードに関連する特定のタイムスタンプ設定などを選択したりできます。測定に参加する指定したデバイスファミリのすべてのデバイスにもこの機能が与えられるため、特定のデバイスを異なる方法で処理する可能性があります(例: トリガマスタがスレーブとは異なる設定/準備を取得するなど)。

C++

```
void PrepareDataAcquisition(
    DataAcquisitionMode dataAcquisitionMode,
    List<Device> devicesToPrepare
);
```

C#

```
public abstract void PrepareDataAcquisition(
    DataAcquisitionMode dataAcquisitionMode,
    List<Device> devicesToPrepare
);
```

Visual Basic

```
Public abstract Function PrepareDataAcquisition(
    dataAcquisitionMode As DataAcquisitionMode,
    devicesToPrepare As List<Device>
) As void
```

引数

引数	説明
dataAcquisitionMode	データ取得モードは、測定の開始方法を表します。
devicesToPrepare	測定に参加するファミリのデバイス。

備考

DataAcquisitionMode.Autoではなく、関数GetPossibleDataAcquisitionModes(List{Device}, out TimeSource)によって返されるDataAcquisitionModesの1つを使用してください。

DeviceFamily.Scan Method

特定のデバイスファミリのデバイスのネットワークスキャンを実行します。派生クラスでオーバーライドする必要があります。

C++

```
List<Device> Scan();
```

C#

```
public abstract List<Device> Scan();
```

Visual Basic

```
Public abstract Function Scan() As List<Device>
```

戻り値

見つかったデバイスのリスト。

例外

例外	説明
Hbm.Api.Scan.Entities. ScanFailedException	スキャン中にエラーが発生した場合にスローされます。

DeviceFamily Properties

DeviceFamilyクラスのプロパティの一覧です。

Public Properties

	名前	説明
	FamilyName	デバイスファミリの一意の名前。派生クラスでオーバーライドする必要があります。
	IsScanSupported	デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。
	SkillReader	SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

DeviceFamily.FamilyName Property

デバイスファミリの一意の名前。派生クラスでオーバーライドする必要があります。

C++

```
__property string FamilyName;
```

C#

```
public abstract string FamilyName;
```

Visual Basic

```
Public abstract Property FamilyName() As string
```


現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> DeviceFamily Class -> DeviceFamily Properties

DeviceFamily.IsScanSupported Property

デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。

C++

```
__property bool IsScanSupported;
```

C#

```
public abstract bool IsScanSupported;
```

Visual Basic

```
Public abstract Property IsScanSupported() As bool
```

DeviceFamily.SkillReader Property

SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

C++

```
__property ISkillReader SkillReader;
```

C#

```
public abstract ISkillReader SkillReader;
```

Visual Basic

```
Public abstract Property SkillReader() As ISkillReader
```

DetectionRange Class

検出されたピークが有効となる低波長と高波長(帯域)をnm単位で定義します。



C++

```
class DetectionRange;
```

C#

```
public class DetectionRange;
```

Visual Basic

```
Public Class DetectionRange
```

ファイル

DetectionRange.cs

DetectionRange Members

次の表は、DetectionRangeによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	WavelengthHigh	高波長境界を取得または設定します。
	WavelengthLow	低波長境界を取得または設定します。

DetectionRange Properties

DetectionRangeクラスのプロパティの一覧です。

Public Properties

	名前	説明
	WavelengthHigh	高波長境界を取得または設定します。
	WavelengthLow	低波長境界を取得または設定します。

DetectionRange.WavelengthHigh Property

高波長境界を取得または設定します。

C++

```
__property double WavelengthHigh;
```

C#

```
public double string WavelengthHigh;
```

Visual Basic

```
Public Property WavelengthHigh() As double
```

説明

高波長の境界(nm)。

DetectionRange.WavelengthLow Property

低波長境界を取得または設定します。

C++

```
__property double WavelengthLow;
```

C#

```
public double string WavelengthLow;
```

Visual Basic

```
Public Property WavelengthLow() As double
```

説明

低波長の境界(nm)。

DeviceErrorStatus Class

デバイスの1つのエラーステータス情報を表します。

Hbm.Api.Common.Entities.DeviceErrorStatus

C++

```
class DeviceErrorStatus;
```

C#

```
public class DeviceErrorStatus;
```

Visual Basic

```
Public Class DeviceErrorStatus
```

ファイル

DeviceErrorStatus.cs

DeviceErrorStatus Members

次の表は、DeviceErrorStatusによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DeviceErrorStatus	これは、DeviceErrorStatusコンストラクタの多重定義の概要です。
	ToString	このインスタンスを表すSystem.Stringを返します。

Public Properties

	名前	説明
	Channel	このエラーステータスのチャネルを取得します。
	Connector	このエラーステータスのコネクタを取得します。
	ErrorCode	エラーコード(デバイス固有)を取得します。
	Message	エラーメッセージを取得します。
	Signal	このエラーステータスの信号を取得します。

DeviceErrorStatus Constructor

これは、DeviceErrorStatusコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	DeviceErrorStatus.DeviceErrorStatus ()	DeviceErrorStatusクラスの新しい(空の)インスタンスを初期化します。
	DeviceErrorStatus.DeviceErrorStatus (int, Connector, Channel, Signal, string)	DeviceErrorStatusクラスの新しいインスタンスを初期化します。

DeviceErrorStatus.DeviceErrorStatus Constructor ()

DeviceErrorStatusクラスの新しい(空の)インスタンスを初期化します。

C++

```
DeviceErrorStatus();
```

C#

```
public DeviceErrorStatus();
```

Visual Basic

```
Public Sub New()
```

DeviceErrorStatus.DeviceErrorStatus Constructor (int, Connector, Channel, Signal, string)

DeviceErrorStatusクラスの新しいインスタンスを初期化します。

C++

```
DeviceErrorStatus(
    int errorCode,
    Connector connector,
    Channel channel,
    Signal signal,
    string message
);
```

C#

```
public DeviceErrorStatus(
    int errorCode,
    Connector connector,
    Channel channel,
    Signal signal,
    string message
);
```

Visual Basic

```
Public Sub New(
    errorCode As Integer,
    connector As Connector,
    channel As Channel,
    signal As Signal,
    message As string
)
```

引数

引数	説明
errorCode	エラーコード。
connector	コネクタ(コネクタレベルでない場合はnull)。
channel	チャネル(チャネルレベルでない場合はnull)。
signal	信号(信号レベルでない場合はnull)。
message	エラーメッセージ。

DeviceErrorStatus Methods

DeviceErrorStatusクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ToString	このインスタンスを表すSystem.Stringを返します。

DeviceErrorStatus.ToString Method

このインスタンスを表すSystem.Stringを返します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

```
Public override Function ToString() As string
```

戻り値

このインスタンスを表すSystem.String(format "<class-name> (<hierachical-type> <unique-id>) {#<error-number> - <message-text>}")。

DeviceErrorStatus Properties

DeviceErrorStatusクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Channel	このエラーステータスのチャネルを取得します。
	Connector	このエラーステータスのコネクタを取得します。
	ErrorCode	エラーコード(デバイス固有)を取得します。
	Message	エラーメッセージを取得します。
	Signal	このエラーステータスの信号を取得します。

DeviceErrorStatus.Channel Property

このエラーステータスのチャネルを取得します。

C++

```
__property Channel Channel;
```

C#

```
public Channel Channel;
```

Visual Basic

```
Public Property Channel() As Channel
```

備考

別の階層レベルにある場合は`null`になる可能性があります。

DeviceErrorStatus.Connector Property

このエラーステータスのコネクタを取得します。

C++

```
__property Connector Connector;
```

C#

```
public Connector Connector;
```

Visual Basic

```
Public Property Connector() As Connector
```

備考

別の階層レベルにある場合は`null`になる可能性があります。

DeviceErrorStatus.ErrorCode Property

エラーコード(デバイス固有)を取得します。

C++

```
__property int ErrorCode;
```

C#

```
public int ErrorCode;
```

Visual Basic

```
Public Property ErrorCode() As Integer
```

DeviceErrorStatus.Message Property

エラーメッセージを取得します。

C++

```
__property string Message;
```

C#

```
public string Message;
```

Visual Basic

```
Public Property Message() As string
```

DeviceErrorStatus.Signal Property

このエラーステータスの信号を取得します。

C++

```
__property Signal Signal;
```

C#

```
public Signal Signal;
```

Visual Basic

```
Public Property Signal() As Signal
```

備考

別の階層レベルにある場合は`null`になる可能性があります。

MeasurementValue Class

ReSharperはInheritdocConsiderUsageを一度無効にします。



C++

```
class MeasurementValue : public MeasurementValueBase<double>;
```

C#

```
public class MeasurementValue : MeasurementValueBase<double>;
```

Visual Basic

```
Public Class MeasurementValue
Inherits MeasurementValueBase
```

ファイル

MeasurementValue.cs

MeasurementValue Members

次の表は、MeasurementValueによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	MeasurementValue	

MeasurementValue.MeasurementValue Constructor

C++

```
MeasurementValue(
    double value,
    double timeStamp,
    MeasurementValueState state
);
```

C#

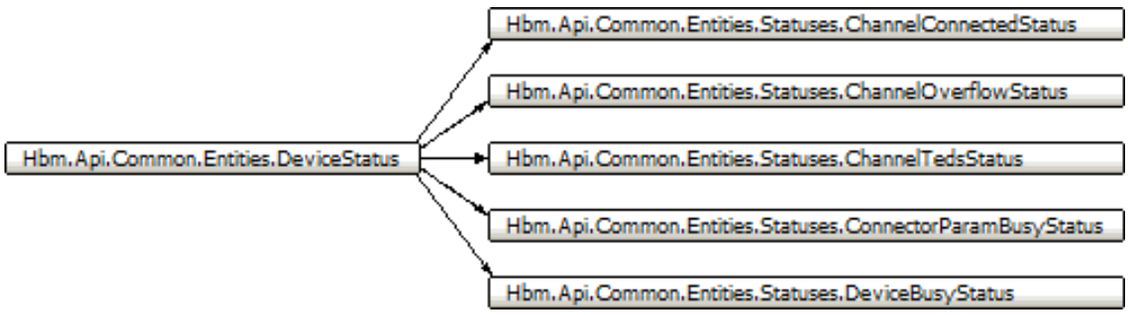
```
public MeasurementValue(
    double value,
    double timeStamp,
    MeasurementValueState state
);
```

Visual Basic

```
Public Sub New(
    value As double,
    timeStamp As double,
    channel As Channel,
    state As MeasurementValueState
)
```

DeviceStatus Class

デバイスの1つのステータス情報を表します。



C++

```
class DeviceStatus;
```

C#

```
public abstract class DeviceStatus;
```

Visual Basic

```
Public abstract Class DeviceStatus
```

ファイル

DeviceStatus.cs

備考

サブクラスは少なくともstatus.Furtherの値を含むプロパティ`Value`(通常は列挙型)を実現する必要があります。さらにサブクラス化ではプロパティ`DeviceSpecificValue`を実現し、基本クラス`Value = MyEnum.DeviceSpecific`をインジケータとして設定する必要があります(QuantumXドライバの`Hbm.Api.Common.Entities.Statuses.ChannelTedsStatus`とそのサブクラスを参照)。

DeviceStatus Members

次の表は、DeviceStatusによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DeviceStatus	これは、DeviceStatus Constructorの多重定義の概要です。
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。
	ToString	このインスタンスを表すSystem.Stringを返します。

Public Properties

	名前	説明
	Channel	このステータスのチャネルを取得します。
	Connector	このステータスのコネクタを取得します。
	OriginalID	このステータスの元の識別子(デバイス固有)を取得します。
	Signal	このステータスの信号を取得します。
	StatusType	このステータスの種類を取得します。

DeviceStatus Constructor

これは、DeviceStatus Constructorの多重定義の概要です。

多重定義リスト

	名前	説明
	DeviceStatus.DeviceStatus ()	DeviceStatusクラスの新しい(空の)インスタンスを初期化します。
	DeviceStatus.DeviceStatus (int, Connector, Channel, Signal)	DeviceStatusクラスの新しいインスタンスを初期化します。

DeviceStatus.DeviceStatus Constructor ()

DeviceStatusクラスの新しい(空の)インスタンスを初期化します。

C++

```
DeviceStatus();
```

C#

```
public DeviceStatus();
```

Visual Basic

```
Public Sub New()
```

DeviceStatus.DeviceStatus Constructor (int, Connector, Channel, Signal)

DeviceStatusクラスの新しいインスタンスを初期化します。

C++

```
DeviceStatus(
    int originalID,
    Connector connector,
    Channel channel,
    Signal signal
);
```

C#

```
public DeviceStatus(
    int originalID,
    Connector connector,
    Channel channel,
    Signal signal
);
```

Visual Basic

```
Public Sub New(
    originalID As Integer,
    connector As Connector,
    channel As Channel,
    signal As Signal
)
```

引数

引数	説明
originalID	このステータスの元の識別子。
connector	コネクタ(コネクタレベルでない場合はnull)。
channel	チャネル(チャネルレベルでない場合はnull)。
signal	信号(信号レベルでない場合はnull)。

DeviceStatus Methods

DeviceStatusクラスのメソッドの一覧です。

Public Methods

	名前	説明
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。
	ToString	このインスタンスを表すSystem.Stringを返します。

DeviceStatus.IsEqual Method

指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

C++

```
bool IsEqual(
    DeviceStatus status
);
```

C#

```
public abstract bool IsEqual(
    DeviceStatus status
);
```

Visual Basic

```
Public abstract Function IsEqual(
    status As DeviceStatus
) As bool
```

引数

引数	説明
status	ステータス。

DeviceStatus.ToString Method

このインスタンスを表すSystem.Stringを返します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

```
Public override Function ToString() As string
```

戻り値

このインスタンスを表すSystem.String(format "<class-name> (<hierachical-type> <unique-id>) {Value = <value-as-string>}")。

DeviceStatus Properties

DeviceStatusクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Channel	このステータスのチャネルを取得します。
	Connector	このステータスのコネクタを取得します。
	OriginalID	このステータスの元の識別子(デバイス固有)を取得します。
	Signal	このステータスの信号を取得します。
	StatusType	このステータスの種類を取得します。

DeviceStatus.Channel Property

このステータスのチャネルを取得します。

C++

```
__property Channel Channel;
```

C#

```
public Channel Channel;
```

Visual Basic

```
Public Property Channel() As Channel
```

備考

別の階層レベルにある場合は`null`になる可能性があります。

DeviceStatus.Connector Property

このステータスのコネクタを取得します。

C++

```
__property Connector Connector;
```

C#

```
public Connector Connector;
```

Visual Basic

```
Public Property Connector() As Connector
```

備考

別の階層レベルにある場合は`null`になる可能性があります。

DeviceStatus.OriginalID Property

このステータスの元の識別子(デバイス固有)を取得します。

C++

```
__property int OriginalID;
```

C#

```
public int OriginalID;
```

Visual Basic

```
Public Property OriginalID() As Integer
```

DeviceStatus.Signal Property

このステータスの信号を取得します。

C++

```
__property Signal Signal;
```

C#

```
public Signal Signal;
```

Visual Basic

```
Public Property Signal() As Signal
```

備考

別の階層レベルにある場合は`null`になる可能性があります。

DeviceStatus.StatusType Property

このステータスの種類を取得します。

C++

```
__property StatusType StatusType;
```

C#

```
public StatusType StatusType;
```

Visual Basic

```
Public Property StatusType() As StatusType
```

MeasurementValues Class

連続計測の計測値を保持するクラス。データ収集の実行中に計測値を読み取るには、Hbm.API.Common.DaqMeasurement.FillMeasurementValuesを呼び出す必要があります。これにより、信号の循環バッファから信号のContinuousMeasurementValueに一定数の計測値がコピーされます。



C++

```
class MeasurementValues : public MeasurementValuesBase<double>;
```

C#

```
public class MeasurementValues : MeasurementValuesBase<double>;
```

Visual Basic

```
Public Class MeasurementValues
Inherits MeasurementValuesBase
```

ファイル

MeasurementValues.cs

FirmwareUpdater Class

デバイスのファームウェア更新を処理するメソッドをカプセル化するクラス(HBMデバイスのみ)。



C++

```
class FirmwareUpdater : public IDisposable;
```

C#

```
public sealed class FirmwareUpdater : IDisposable;
```

Visual Basic

```
Public sealed Class FirmwareUpdater
Inherits IDisposable
```

ファイル

FirmwareUpdater.cs

FirmwareUpdater Members

次の表は、FirmwareUpdaterによって公開されるメンバーの一覧です。

Private Methods

	名前	説明
	FirmwareUpdater	コンストラクタ(HBMデバイスのみ)。

Public Methods

	名前	説明
	Dispose	インタフェースIDisposableのメソッド。
	RequestCollectionFileImageVersion	選択したファイルのイメージバージョンを要求(HBMデバイスのみ)。
	UpdateFirmware	選択したデバイスのファームウェアを更新(HBMデバイスのみ)。

FirmwareUpdater.FirmwareUpdater Constructor

コンストラクタ(HBMデバイスのみ)。

C++

```
FirmwareUpdater();
```

C#

```
private FirmwareUpdater();
```

Visual Basic

```
Private Sub New()
```

FirmwareUpdater Methods

FirmwareUpdaterクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Dispose	インタフェースIDisposableのメソッド。
	RequestCollectionFileImageVersion	選択したファイルのイメージバージョンを要求(HBMデバイスのみ)。
	UpdateFirmware	選択したデバイスのファームウェアを更新(HBMデバイスのみ)。

FirmwareUpdater.Dispose Method

インタフェースIDisposableのメソッド。

C++

```
void Dispose();
```

C#

```
public void Dispose();
```

Visual Basic

```
Public Function Dispose() As void
```

FirmwareUpdater.RequestCollectionFileImageVersion Method

選択したファイルのイメージバージョンを要求(HBMデバイスのみ)。

C++

```
void RequestCollectionFileImageVersion(
    string fileName,
    List<Device> devices,
    bool allowFirmwareVersionDowngrade,
    out Dictionary<Device, List<Problem>> devicesProblems
);
```

C#

```
public void RequestCollectionFileImageVersion(
    string fileName,
    List<Device> devices,
    bool allowFirmwareVersionDowngrade,
    out Dictionary<Device, List<Problem>> devicesProblems
);
```

Visual Basic

```
Public Function RequestCollectionFileImageVersion(
    fileName As string,
    devices As List<Device>,
    allowFirmwareVersionDowngrade As bool,
    devicesProblems As out Dictionary<Device, List<Problem>>
) As void
```

引数

引数	説明
fileName	更新するファームウェアが含まれるファイル名。
devices	更新するデバイスのリスト。
allowFirmwareVersionDowngrade	ダウングレードを許可するかどうか。
devicesProblems	発生した問題。

FirmwareUpdater.UpdateFirmware Method

選択したデバイスのファームウェアを更新(HBMデバイスのみ)。

C++

```
void UpdateFirmware(
    string fileName,
    List<Device> devices,
    bool allowFirmwareVersionDowngrade,
    out Dictionary<Device, List<Problem>> devicesProblems
);
```

C#

```
public void UpdateFirmware(
    string fileName,
    List<Device> devices,
    bool allowFirmwareVersionDowngrade,
    out Dictionary<Device, List<Problem>> devicesProblems
);
```

Visual Basic

```
Public Function UpdateFirmware(
    fileName As string,
    devices As List<Device>,
    allowFirmwareVersionDowngrade As bool,
    devicesProblems As out Dictionary<Device, List<Problem>>
) As void
```

引数

引数	説明
fileName	ファームウェアのファイル名。
devices	更新するデバイスのリスト。
allowFirmwareVersionDowngrade	ダウングレードを許可するかどうか。
devicesProblems	発生した問題。

Service Class

デバイスによって提供されるサービス(SCPIなど)を表します。



C++

```
class Service;
```

C#

```
public class Service;
```

Visual Basic

```
Public Class Service
```

ファイル

Service.cs

Service Members

次の表は、Serviceによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	Port	サービスが属しているポート。
	Protocol	サポートしているプロトコル(例: SCPI)。

Service Properties

FirmwareUpdaterクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Port	サービスが属しているポート。
	Protocol	サポートしているプロトコル(例: SCPI)。

Service.Port Property

サービスが属しているポート。

C++

```
__property int Port;
```

C#

```
public int Port;
```

Visual Basic

```
Public Property Port() As Integer
```

Service.Protocol Property

サポートしているプロトコル(例: SCPI)。

C++

```
__property string Protocol;
```

C#

```
public string Protocol;
```

Visual Basic

```
Public Property Protocol() As string
```

StreamingDevice Class

データ取得にHBMストリーミングを使用するデバイスの抽象基底クラス。



C++

```
class StreamingDevice : public Device;
```

C#

```
public abstract class StreamingDevice : Device;
```

Visual Basic

```
Public abstract Class StreamingDevice
Inherits Device
```

ファイル

StreamingDevice.cs

StreamingDevice Members

次の表は、StreamingDeviceによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	GetAndBroadcastDeviceStatusList	デバイスのステータスリストを取得し、サブスクリプションをフィルタリングして、変更された状態をブロードキャストします。

Protected internal Methods

	名前	説明
	StoreStatusIfChanged	ステータスがリストに保存されていない場合は保存し、変更された場合は置き換えます。

Protected Methods

	名前	説明
	BroadcastChannelStatusChanges	ステータスの変化をブロードキャストします。
	BroadcastTedsStatus	tedステータスを処理します。
	MakeChecks	複数のメソッド間で使用するいくつかのプロパティのチェックを集中管理するヘルパメソッド。
	SetAnalogOutOutputScaling	アナログ出力信号のデフォルトの出力テーブルのスケーリングを設定します。

Public Methods

	名前	説明
	ActivateTeds	指定したコネクタのTEDS設定をアクティブにし、新しいチャネルパラメータをロードします。
	AssignAutoCalibrationMode	指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignChannel	指定したチャネルの設定(センサ、信号などを含む)をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignConnector	指定したコネクタの設定(チャネル、センサ信号などを含む)をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSensor	指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSignal	指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignTimeSource	TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignZero	指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AutoDetectFbgBands	指定したFbgConnectorに対する帯域検出の結果を返します。
	ChangeIp	デバイスのipAddressを変更します。
	ClearZeroBalance	これは、ClearZeroBalance メソッドの多重定義の概要です。
	DisableRouting	ルーティング機能を無効にします。
	EnableRouting	ルーティング機能を有効にします。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Entities Namespace -> StreamingDevice Class

	名前	説明
	Find	指定した接続情報を使用してデバイスを検索します(実際には、コネクタ情報を読み取ることなくデバイス情報を読み取ります)。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	FindChannel	これは、FindChannelメソッドの多重定義の概要です。
	FindConnector	これは、FindConnectorメソッドの多重定義の概要です。
	FindSignal	指定したフィルタが配置されている信号を返します。
	FlashLed	これは、FlashLedメソッドの多重定義の概要です。
	GetAllChannels	デバイスのすべてのチャネルを取得します。
	GetAllSignals	デバイスのすべての信号を取得します。
	GetAmplifierSerialNumber	チャネルのアンプのシリアル番号を取得します。
	GetAmplifierType	チャネルのアンプタイプを取得します。
	GetBestFittingAutoFilter	信号のサンプルレートに応じて最適な自動フィルタを取得します。
	GetCalibrationCertificateFileInfos	要求したリリースの情報に応じて、校正証明ファイル情報のリストを返します。校正証明書を作成し、指定した出力ディレクトリにダウンロードします。
	GetChannelHardwareInfo	チャネルのハードウェア情報(アンプの種類など)を取得します。
	GetDeviceErrorStatusList	これは、GetDeviceErrorStatusListメソッドの多重定義の概要です。
	GetDeviceStatusList	これは、GetDeviceStatusListメソッドの多重定義の概要です。
	GetPossibleDataAcquisitionModes	デバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブル経由で確立)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedを含みます。デバイスが時間ベースの同期(PTPまたはNTPなど)を使用している場合、返されるリストにはDataAcquisitionMode.TimestampSynchronizedモードを含みます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはDataAcquisitionMode.Unsynchronizedモードを含む可能性が高くなります。デバイスが接続されていない場合、使用可能なデータ収集モードのリストは空になります。
	GetShunt	これは、GetShuntメソッドの多重定義の概要です。
	GetSupportedCanBitRates	サポートしているCANのビットレートを取得します。
	GetSupportedFilterFrequencies	指定したフィルタのサポートされているフィルタ周波数のリストを取得します。
	GetSupportedFilters	指定した信号のサポートされているフィルタのリストを取得します。
	GetSupportedJackType	指定した信号でサポートしているジャックの種類を取得します。
	GetSupportedSampleRates	サポートしているサンプルレートのリストを取得します。
	GetSupportedScalingTypes	指定したチャネルでサポートしているスケーリングタイプのリストを取得します。
	GetSupportedSensorTypes	指定したチャネルでサポートしているセンサタイプのリストを取得します。
	GetTimeSourceQuality	オフセット(同期タイムソースとデバイスの時間の間のタイムスパン)をミリ秒単位で返し、同期の品質に関する詳細情報を含むTimeSourceに依存する文字列を返します。NtpTimeSourceの場合、この文字列にはパラメータ(カンマ区切り)が含まれます: "remote、refid、st、t、when、pol、reach、delay、offset、jitter"。PtpTimeSourceの場合、この文字列には次のパラメータ (カンマ区切り)が含まれます: "GrandmasterID、SyncMode"、TimeScaleFlags、UtcOffset、UtcOffsetused、MasterOffset"
	GetUniqueID	すべてのデバイスファミリの中からデバイスを識別する一意の文字列を返します。
	GetUnit	指定したチャネルの単位を返します。
	IsChannelPartOfDevice	指定したチャネルがデバイスに属しているかどうかをチェックします。
	IsConnectorPartOfDevice	指定したコネクタがデバイスに属しているかどうかをチェックします。

	名前	説明
	IsFirmwareVersionSupported	指定したファームウェアのバージョンをサポートしているかどうかを判断します。
	IsSignalPartOfDevice	指定した信号がデバイスに属しているかどうかをチェックします。
	ReadDigitalIO	デジタル信号の現在の測定値を返します。
	ReadFbgSpectrum	指定したスペクトルタイプとコネクタに従ってFbgSpectrumを返します。この機能は、ファイバブラッググレーティング関連デバイスでのみサポートしいます。
	ReadSingleElectricalValue	指定したAnalogInSignalの現在の電気値を返します。
	ReadSingleMeasurementValue	DAQセッションを完全に初期化せずに、指定したSignalごとの単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadSingleMeasurementValueOfAllSignals	DAQセッションを完全に初期化せずに、デバイスのすべてのSignalに対して単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadTeds	指定したコネクタに接続されているTEDSからセンサを読み取ります。
	RemoveInputSignalReference	指定したチャネルから入力信号のリファレンスを削除します。
	Reset	デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数の呼び出し後に、デバイスは切断されます。これが必要なのはデバイスのパラメータ設定が変更され、デバイスがデバイスの関連オブジェクトモデルと一致しなくなったとき、工場出荷時のデフォルトをロードします(この関数を実装するときは、必ずデバイスを切断し、Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知します。
	ResetStatusMonitoringStates	ステータス監視状態をリセットして、次回のステータスポーリングですべてのイベントが再度発生できるようにします。
	SetAnalogOut	アナログ出力信号を指定した値に設定します。
	SetAnalogOutChannelActivation	提供されたチャネルのリストをアクティブまたは非アクティブにします。AnalogOutChannelのみをアクティブ化または非アクティブ化できます。
	SetCanBusParameters	指定したCanConnectorのすべてのCAN busパラメータ(CanConnector.BitRate、CanConnector.IsTerminated、CanConnector.SupplyVoltageなど)を割り当てますが、チャネルは割り当てません。
	SetCanChannelActivation	提供されたチャネルのリストをアクティブまたは非アクティブにします。CanInChannelとCanOutChannelのみをアクティブ化または非アクティブ化できます。
	SetChannelName	指定したチャネルにチャネル名を割り当てます。
	SetDataRateDomain	現在のモジュールに指定したdataRateDomainを設定します。重要:デバイスの種類によっては、再起動が自動的に行われます。この場合、デバイスはしばらくの間アクセスできなくなります。
	SetDeviceName	基になる物理デバイスの名前を設定します。派生クラスでオーバーライドする必要があります。
	SetDigitalIO	指定した値に従ってデジタル信号を設定します。
	SetFbgBandsDetectionResult	AutoDetectFbgBands経由で事前に取得した、指定されたFbgBandsDetectionResultに従って、指定されたFbgConnectorのFbgChannelのFBG関連プロパティ(DetectionRange、FbgSensor.ReferenceWavelength、FbgChannel.IsActivatedなど)を設定します。

	名前	説明
	SetFbgChannelActivation	指定したチャネルのリストをアクティブまたは非アクティブにします。FbgChannelのみをアクティブ化または非アクティブ化できます。
	SetFbgChannelDetectionRange	指定した値に従って、指定されたFbgChannelのFbgChannel.DetectionRangeを設定します。
	SetFbgDetectionThreshold	指定した値に従って、指定したFbgConnectorのFbgConnector.Detection Thresholdを設定します。
	SetInputSignalReference	入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。
	SetShunt	これは、SetShuntメソッドの多重定義の概要です。
	SetSignalName	指定した信号に信号名を割り当てます。
	SetTedsUsageMode	TEDS使用モードを設定し、コネクタ設定を再読み込みします。
	SetUnit	指定したチャネルの単位を割り当てます。
	SetZeroBalance	これは、SetZeroBalanceメソッドの多重定義の概要です。
	StreamingDevice	インスタンスを作成し、デバイスに一意のストリーミングIDを割り当てます。
	SubscribeToStatusMonitoring	指定したチャネルをステータス監視に登録します。
	UnsubscribeFromStatusMonitoring	指定したチャネルをステータス監視から登録解除します。
	WriteTeds	指定したコネクタに接続されたTEDSに指定したセンサを書き込みます。

Public Properties

	名前	説明
	CalCertSyncLock	基礎となるデバイスの校正証明書の伝達への並列アクセスを同期するために使用します。
	ConnectionInfo	物理デバイスへの接続方法に関する情報。
	Connectors	すべてのコネクタのリスト。
	DataAcquisitionMode	測定に使用するデータ収集モードを保持します。このモードは通常、DaqMeasurement.StartDaq(Enums.DataAcquisitionMode, int)中に呼び出されるDeviceFamily.PrepareDataAcquisition(Enums.DataAcquisitionMode,List{Device})によって設定されます。特にEnums.DataAcquisitionMode.Autoの場合、このプロパティは、測定の開始後に選択したデータ取得モードを返します。このプロパティを手動で設定しないでください! 内部使用のみを目的としています。
	DataRateDomain	データレートドメインを取得または設定します。
	FamilyName	このデバイスが属するファミリの名前。
	FbgFiberInfo	ファイバセンシングベースのデバイスの帯域情報を保持します。これは、ファイバセンシングベースのデバイスの場合にのみ入力します。
	FirmwareVersion	ファームウェアのバージョン。
	HardwareVersion	ハードウェアのバージョン。
	HasAdditionalTedsFieldsSupport	デバイスファミリがキャリブレーションおよびシリアル番号のTEDSフィールドをサポートしているかどうかを示す値を取得します。
	IsAnalogOutChannelDisablingSupported	デバイスがアナログ出力チャネルの無効化をサポートしているかどうかを示す値を取得または設定します。
	IsAutomaticTedsMonitoringSupported	このインスタンスが自動TEDS監視をサポートしているかどうかを示す値を取得します。
	IsCanBusSupplyVoltageSupported	デバイスがCAN busの供給電圧をサポートしているかどうかを示す値を取得または設定します。
	IsCanBusTerminationSupported	デバイスがCAN busターミネーションをサポートしているかどうかを示す値を取得または設定します。
	IsConnectable	デバイスが接続可能かどうかを示します。
	IsConnected	デバイスが接続されているかどうかを示します。

	名前	説明
R V	IsElectricalSnapshotSupported	ReadSingleElectricalValue(AnalogInSignal)関数を使用して、このインスタンスが電気値の要求をサポートしているかどうかを示す値を取得します。
	IsEnablingRoutingSupported	デバイスがルーティング機能の有効化/無効化をサポートしているかどうかを示す値を取得または設定します。
V	IsFirmwareUpdateSupported	このインスタンスがファームウェアの更新をサポートしているかどうかを示す値を取得または設定します。
	IsIpChangeSupported	デバイスがIPアドレスの変更をサポートしているかどうかを示す値を取得または設定します。
	IsLedFlashSupported	デバイスがLEDの点滅をサポートしているかどうかを示す値を取得または設定します。
	IsReadingDaqValues	デバイスが測定値を更新中かどうかを示します。
R V	IsReadingFbgSpectrumSupported	このインスタンスがスペクトルの読み取りをサポートしているかどうかを示す値を取得します。
	IsRoutingEnabled	デバイスのルーティング機能が現在有効になっているかどうかを示す値を取得または設定します。
	IsUnsupportedModel	特定のモデルがAPIでサポートされているかどうかを示します。
	IsUsingHbmClassicSampleRates	デバイスが10進数のサンプル レート(10Hz、100Hz、1000Hzなど)ではなく、従来のHBMサンプル レート(1200Hz、2400Hz、4800Hz)など) を使用しているかどうかを示します。このプロパティはデバイスの接続後に有効になります。現在、QuantumXデバイスファミリのみが10進数のサンプルレートをサポートしています。
	MinimumFirmwareVersionSupported	ドライバがサポートする最小ファームウェア バージョン。
	Model	デバイスモデルのタイプ。
R	MonitoredStatusTypes	監視対象のステータスの種類を取得します。
	Name	デバイス名。
R	NtpStartDate	秒数のカウントの開始日を協定世界時(UTC)で決定する日付。タイムスタンプは、この日付/時刻からの秒数で表されます。
S	Overflow	無効な測定値(オーバーフロー)の代わりに使用する値を定義します。ステータスが無効な各測定値については、対応する測定値がこのオーバーフロー値に設定されます。DaqEnvironment.Overflowを使用してこの値を設定します(デフォルトは1000000)。
	ProtectionType	デバイスの保護等級タイプ(例: IP55)。
	SerialNo	デバイスのシリアル番号。
	SupportedDataRateDomains	サポートしているデータレートドメインを取得または設定します。
	SyncLock	基礎となるデバイスへの並列アクセスを同期するために使用します。
	SyncMode	測定開始時のデバイスの役割を定義するモード。
	Timeout	デバイスとの通信のタイムアウト。
	TimeSource	物理デバイスの時刻を同期する方法に関する情報。

StreamingDevice.StreamingDevice Constructor

インスタンスを作成し、デバイスに一意のストリーミングIDを割り当てます。

C++

```
StreamingDevice();
```

C#

```
private StreamingDevice();
```

Visual Basic

```
Private Sub New()
```

Zero Class

アナログチャネルのゼロオフセット。



C++

```
class Zero;
```

C#

```
public class Zero;
```

Visual Basic

```
Public Class Zero
```

ファイル

Zero.cs

Zero Members

次の表は、Zeroによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Zero	これは、Zero Constructorの多重定義の概要です。

Public Properties

	名前	説明
	IsZeroBalancingInhibited	ゼロバランスの実行を許可しない場合にマークします(値が`true`の場合)。
	Offset	オフセット(測定値から差し引く値)。
	Target	ターゲット(ゼロ化後にゼロの代わりに使用する値)。

Zero Constructor

これは、Zero Constructorの多重定義の概要です。

多重定義リスト

	名前	説明
	Zero.Zero ()	Zeroクラスの新しい(空の)インスタンスを初期化します。
	Zero.Zero (double, double, bool)	Zeroクラスの新しいインスタンスを初期化します。

Zero.Zero Constructor ()

Zeroクラスの新しい(空の)インスタンスを初期化します。

C++

```
Zero();
```

C#

```
public Zero();
```

Visual Basic

```
Public Sub New()
```

Zero.Zero Constructor (double, double, bool)

Zeroクラスの新しいインスタンスを初期化します。

C++

```
Zero(
    double offset,
    double target,
    bool isZeroBalancingInhibited
);
```

C#

```
public Zero(
    double offset,
    double target,
    bool isZeroBal
);
```

Visual Basic

```
Public Sub New(
    offset As double,
    target As double,
    isZeroBalancingInhibited As bool
)
```

引数

引数	説明
offset	オフセット(測定値から差し引く値)。
target	ターゲット(ゼロ化後にゼロの代わりに使用する値)。
isZeroBalancingInhibited	ゼロバランスの実行を許可しない場合にマークします(値が`true`の場合)。

Zero Properties

Zeroクラスのプロパティの一覧です。

Public Properties

	名前	説明
	IsZeroBalancingInhibited	ゼロバランスの実行を許可しない場合にマークします(値が`true`の場合)。
	Offset	オフセット(測定値から差し引く値)。
	Target	ターゲット(ゼロ化後にゼロの代わりに使用する値)。

Zero.IsZeroBalancingInhibited Property

ゼロバランスの実行を許可しない場合にマークします(値がtrueの場合)。

C++

```
__property bool IsZeroBalancingInhibited;
```

C#

```
public bool IsZeroBalancingInhibited;
```

Visual Basic

```
Public Property IsZeroBalancingInhibited() As bool
```

備考

IsZeroBalancingInhibitedをtrueに設定している場合でも、OffsetとTargetが設定できることに注意してください。Hbm.Api.Common.Entities.Device.SetZeroBalance関数のみがこのプロパティを考慮します。

Zero.Offset Property

オフセット(測定値から差し引く値)。

C++

```
__property double Offset;
```

C#

```
public double Offset;
```

Visual Basic

```
Public Property Offset() As double
```

Zero.Target Property

ターゲット(ゼロ化後にゼロの代わりに使用する値)。

C++

```
__property double Target;
```

C#

```
public double Target;
```

Visual Basic

```
Public Property Target() As double
```

Hbm.Api.Common.Enums Namespace

これは名前空間Hbm.Api.Common.Enumsです。

クラス

	名前	説明
S	TedsValidationValueTypeExtensions	TedsValidationValueTypeの拡張関数。

列挙型

	名前	説明
	AutoCalibrationMode	チャネルを自動的に再校正する方法を記述するモード。
	ChangeReason	MessageBroker.FilterChangedなどのイベント変更の理由の列挙。
	ChannelConnectedValueType	ChannelConnectedStatusのvalueパラメータの値を定義。
	ChannelOverflowValueType	Entities.Statuses.ChannelOverflowStatusのvalueパラメータの値を定義。
	ConnectorParamBusyValueType	Hbm.Api.Common.Entities.Statuses.ConnectorParamBusyStatusのvalueパラメータの値を定義。
	DataAcquisitionMode	データ収集モード。
	DataRateDomainType	データレートドメインの種類。
	DelayMechanism	デバイスの時刻を同期するために使用するPTP遅延メカニズム。
	DeviceBusyValueType	Entities.Statuses.DeviceBusyStatusのvalueパラメータの値を定義。
	DigitalValueType	デジタル値の種類。
	DirectionType	デジタルIOコネクタの方向の種類。
	ErrorResetMode	エラーの解決後にCANエラーまたは警告を削除する方法を定義。
	FbgSpectrumType	ファイバーブラッググレーティング関連デバイスのスペクトルタイプの列挙。
	LedFlashMode	点滅させるLEDの色。
	MeasurementValueState	測定値のステータス。
	NetworkSettingsConfiguration	ネットワーク設定の構成。
	OutputScalingType	アナログ出力コネクタの出力スケーリングの種類。
	ReleaseType	要求した校正証明に必要な種類を決定します。
	SettingType	設定を適応したオブジェクトの種類。
	ShuntMode	シャント出力の構成を記述するモード。
	StatusType	デバイス/コネクタ/チャネルのステータスの種類(すべてのデバイスがすべてのステータスをサポートしているわけではありません)。
	SynchronousMode	同期モードは信号の測定配信の種類を表します。
	SyncModeType	同期モードの種類。
	TedsUsageMode	センサのTEDS使用モードの値を定義。
	TedsValidationValueType	Entities.Statuses.ChannelTedsStatusのvalueパラメータの値を定義。
	TransportMode	デバイスの同期に使用するPTPネットワークトランスポートプロトコル。

TedsValidationValueTypeExtensions Class

TedsValidationValueTypeの拡張関数。

Hbm.Api.Common.Enums.TedsValidationValueTypeExtensions

C++

```
class TedsValidationValueTypeExtensions;
```

C#

```
public static class TedsValidationValueTypeExtensions;
```

Visual Basic

```
Public static Class TedsValidationValueTypeExtensions
```

ファイル

TedsValidationValueTypeExtensions.cs

TedsValidationValueTypeExtensions Members

次の表は、TedsValidationValueTypeExtensionsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
S	IsTedsConnected	TEDSが接続されているかどうかを判断します。
S	IsTedsError	TEDSがエラー状態かどうかを判断します。
S	IsTedsUsed	TEDSを使用中かどうかを判断します(したがって有効ですが、おそらく警告が表示されます)。
S	IsTedsWarning	TEDSが警告状態にあるかどうかを判断します。

TedsValidationValueTypeExtensions Methods

TedsValidationValueTypeExtensionsクラスのメソッドの一覧です。

Public Methods

	名前	説明
S	IsTedsConnected	TEDSが接続されているかどうかを判断します。
S	IsTedsError	TEDSがエラー状態かどうかを判断します。
S	IsTedsUsed	TEDSを使用中かどうかを判断します(したがって有効ですが、おそらく警告が表示されます)。
S	IsTedsWarning	TEDSが警告状態にあるかどうかを判断します。

TedsValidationValueTypeExtensions.IsTedsConnected Method

TEDSが接続されているかどうかを判断します。

C++

```
bool IsTedsConnected(
    this TedsValidationValueType tedsValidationValue
);
```

C#

```
public static bool IsTedsConnected(
    this TedsValidationValueType tedsValidationValue
);
```

Visual Basic

```
Public static Function IsTedsConnected(
    tedsValidationValue As TedsValidationValueType
) As bool
```

引数

引数	説明
tedsValidationValue	TEDS検証値。

戻り値

TEDSが接続されている場合は`true`。それ以外の場合は`false`。

TedsValidationValueTypeExtensions.IsTedsError Method

TEDSがエラー状態かどうかを判断します。

C++

```
bool IsTedsError(
    this TedsValidationValueType tedsValidationValue
);
```

C#

```
public static bool IsTedsError(
    this TedsValidationValueType tedsValidationValue
);
```

Visual Basic

```
Public static Function IsTedsError(
    tedsValidationValue As TedsValidationValueType
) As bool
```

引数

引数	説明
tedsValidationValue	TEDS検証値。

戻り値

TEDSが接続されている場合は`true`。それ以外の場合は`false`。

TedsValidationValueTypeExtensions.IsTedsUsed Method

TEDSを使用中かどうかを判断します(したがって有効ですが、おそらく警告が表示されます)。

C++

```
bool IsTedsUsed(
    this TedsValidationValueType tedsValidationValue
);
```

C#

```
public static bool IsTedsUsed(
    this TedsValidationValueType tedsValidationValue
);
```

Visual Basic

```
Public static Function IsTedsUsed(
    tedsValidationValue As TedsValidationValueType
) As bool
```

引数

引数	説明
tedsValidationValue	TEDS検証値。

戻り値

TEDSが接続されている場合は`true`。それ以外の場合は`false`。

TedsValidationValueTypeExtensions.IsTedsWarning Method

TEDSが警告状態にあるかどうかを判断します。

C++

```
bool IsTedsWarning(
    this TedsValidationValueType tedsValidationValue
);
```

C#

```
public static bool IsTedsWarning(
    this TedsValidationValueType tedsValidationValue
);
```

Visual Basic

```
Public static Function IsTedsWarning(
    tedsValidationValue As TedsValidationValueType
) As bool
```

引数

引数	説明
tedsValidationValue	TEDS検証値。

戻り値

TEDSが接続されている場合は`true`。それ以外の場合は`false`。

Hbm.Api.Common.Enums.AutoCalibrationMode Enumeration

チャネルを自動的に再校正する方法を記述するモード。

C++

```
enum AutoCalibrationMode {
  NotSupported = -1,
  Default = 0,
  AutoAdjust = 1,
  AutoCalibrate = 2,
  Off = 3
};
```

C#

```
public enum AutoCalibrationMode {
  NotSupported = -1,
  Default = 0,
  AutoAdjust = 1,
  AutoCalibrate = 2,
  Off = 3
}
```

Visual Basic

```
Public Enum AutoCalibrationMode
  NotSupported = -1
  Default = 0
  AutoAdjust = 1
  AutoCalibrate = 2
  Off = 3
End Enum
```

ファイル

AutoCalibrationMode.cs

メンバー

メンバー	説明
NotSupported	AutoCalibrationはチャネルでサポートしていません(別の値に設定すると、割り当て中にHbm.Api.Common.Entities.Problems.ValueNotSetErrorが発生します)。
Default	デフォルト(デバイス自体が使用するモードを決定します)。
AutoAdjust	自動調整を使用します(デバイスは並列アンプを使用してチャネルのアンプを調整するため、校正プロセス中に測定値がフリーズすることはありません)。
AutoCalibrate	自動校正を使用します(デバイスは同じチャネルのアンプを使用して校正を実行するため、校正プロセス中に測定値が数ミリ秒フリーズする可能性があります)。
Off	デバイスはチャネルの自動校正を実行しません。

Hbm.Api.Common.Enums.ChangeReason Enumeration

MessageBroker.FilterChangedなどのイベント変更の理由の列挙。

C++

```
enum ChangeReason {
  DependentChannel,
  Teds,
  Unspecific
};
```

C#

```
public enum ChangeReason {
  DependentChannel,
  Teds,
  Unspecific
}
```

Visual Basic

```
Public Enum AutoCalibrationMode
  DependentChannel
  Teds
  Unspecific
End Enum
```

ファイル

ChangeReason.cs

メンバー

メンバー	説明
DependentChannel	チャネルに依存する変更により発生。
Teds	TEDSの変更により発生。
Unspecific	他の理由により発生。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Enums Namespace

Hbm.Api.Common.Enums.ChannelConnectedValueType Enumeration

ChannelConnectedStatusのvalueパラメータの値を定義。

C++

```
enum ChannelConnectedValueType {
  NotSupported = -1,
  NotConnected = 0,
  IsConnected = 1
};
```

C#

```
public enum ChannelConnectedValueType {
  NotSupported = -1,
  NotConnected = 0,
  IsConnected = 1
}
```

Visual Basic

```
Public Enum ChannelConnectedValueType
  NotSupported = -1
  NotConnected = 0
  IsConnected = 1
End Enum
```

ファイル

ChannelConnectedValueType.cs

メンバー

メンバー	説明
NotSupported	サポートしていません。
NotConnected	チャンネルにセンサが接続されていません。
IsConnected	チャンネルにセンサが接続されています。

Hbm.Api.Common.Enums.ChannelOverflowValueType Enumeration

Entities.Statuses.ChannelOverflowStatusのvalueパラメータの値を定義。

C++

```
enum ChannelOverflowValueType {
  NotSupported = -1,
  NoMeasValOverflow = 0,
  MeasValOverflow = 1
};
```

C#

```
public enum ChannelOverflowValueType {
  NotSupported = -1,
  NoMeasValOverflow = 0,
  MeasValOverflow = 1
}
```

Visual Basic

```
Public Enum ChannelOverflowValueType
  NotSupported = -1
  NoMeasValOverflow = 0
  MeasValOverflow = 1
End Enum
```

ファイル

ChannelOverflowValueType.cs

メンバー

メンバー	説明
NotSupported	サポートしていません。
NoMeasValOverflow	チャネルはオーバーフロー状態ではありません(測定値)。
MeasValOverflow	チャネルはオーバーフロー状態です(測定値)。

Hbm.Api.Common.Enums.
ConnectorParamBusyValueType Enumeration

Hbm.Api.Common.Entities.Statuses.ConnectorParamBusyStatusのvalueパラメータの値を定義。

C++

```
enum ConnectorParamBusyValueType {
  NotSupported = -1,
  NotBusy = 0,
  Busy = 1
};
```

C#

```
public enum ConnectorParamBusyValueType {
  NotSupported = -1,
  NotBusy = 0,
  Busy = 1
}
```

Visual Basic

```
Public Enum ConnectorParamBusyValueType
  NotSupported = -1
  NotBusy = 0
  Busy = 1
End Enum
```

ファイル

ConnectorParamBusyValueType.cs

メンバー

メンバー	説明
NotSupported	サポートしていません。
NotBusy	コネクタはパラメータ化でビジーではありません。
Busy	コネクタはパラメータ化でビジーです。

Hbm.Api.Common.Enums.DataAcquisitionMode Enumeration

データ収集モード。

C++

```
enum DataAcquisitionMode {
  TimestampSynchronized = 0,
  HardwareSynchronized = 1,
  Unsynchronized = 2,
  Auto = 3
};
```

C#

```
public enum DataAcquisitionMode {
  TimestampSynchronized = 0,
  HardwareSynchronized = 1,
  Unsynchronized = 2,
  Auto = 3
}
```

Visual Basic

```
Public Enum DataAcquisitionMode
  TimestampSynchronized = 0
  HardwareSynchronized = 1
  Unsynchronized = 2
  Auto = 3
End Enum
```

ファイル

DataAcquisitionMode.cs

メンバー

メンバー	説明
TimestampSynchronized	配信した測定値のタイムスタンプを使用してデバイスの同期を実行する必要がある場合は、このモードを使用します(例: FireWireケーブル経由で接続している場合、複数のQuantumXデバイスの組み合わせは等しいタイムスタンプを配信します。複数のPMXデバイスの組み合わせは、同期したタイムスタンプを配信するために同じNTPサーバを使用する必要があります。)。等間隔信号と非等間隔信号(非同期信号)が混在する可能性があります。非等間隔信号は、`DaqMeasurement.StartDaq`で指定したタイムアウトが発生する前に値を送信する必要があります。
HardwareSynchronized	デバイスがハードウェア(ケーブル)によって同期し、すべての信号に対して同時に最初の測定値を提供できる場合、HardwareSynchronizedを使用します(例: 同期ケーブル経由で接続され、適切に構成(マスタ/スレーブ)されている複数のMGCデバイスの組み合わせは、すべての信号に対して同時に最初の測定値を提供します。)。
Unsynchronized	同期せずに測定を開始するには、この非同期を使用します。
Auto	このモードを使用すると、APIがどのモードを使用するかを決定します。

備考

データ収集モードは、測定の開始時にさまざまなデバイスを同期するために使用する手順(`DaqMeasurement.StartDaq`)と、測定値を提供する方法(`DaqMeasurement.FillMeasurementValues`)を定義します。

Hbm.Api.Common.Enums.DataRateDomainType Enumeration

データレートドメインの種類。

C++

```
enum DataRateDomainType {
  Classic = 0,
  Decimal = 2,
  Binary = 3
};
```

C#

```
public enum DataRateDomainType {
  Classic = 0,
  Decimal = 2,
  Binary = 3
}
```

Visual Basic

```
Public Enum DataRateDomainType
  Classic = 0
  Decimal = 2
  Binary = 3
End Enum
```

ファイル

DataRateDomainType.cs

メンバー

メンバー	説明
Classic	クラシックデータ レートドメイン。
Decimal	10進データ レートドメイン。
Binary	バイナリデータ レートドメイン。

備考

このモードはストリーミングデバイスに必須です。値はQuantumXドメインレートに従って設定されます。

Hbm.Api.Common.Enums.DelayMechanism Enumeration

デバイスの時刻を同期するために使用するPTP遅延メカニズム。

C++

```
enum DelayMechanism {
  Auto = 0,
  E2E = 1,
  P2P = 2
};
```

C#

```
public enum DelayMechanism {
  Auto = 0,
  E2E = 1,
  P2P = 2
}
```

Visual Basic

```
Public Enum DelayMechanism
  Auto = 0
  E2E = 1
  P2P = 2
End Enum
```

ファイル

DelayMechanism.cs

メンバー

メンバー	説明
Auto	自動遅延メカニズム。
E2E	リクエスト - レスポンス遅延。
P2P	ピア遅延メカニズム。

Hbm.Api.Common.Enums.DeviceBusyValueType Enumeration

Entities.Statuses.DeviceBusyStatusのvalueパラメータの値を定義。

C++

```
enum DeviceBusyValueType {
  NotSupported = -1,
  NotBusy = 0,
  Busy = 1
};
```

C#

```
public enum DeviceBusyValueType {
  NotSupported = -1,
  NotBusy = 0,
  Busy = 1
}
```

Visual Basic

```
Public Enum DeviceBusyValueType
  NotSupported = -1
  NotBusy = 0
  Busy = 1
End Enum
```

ファイル

DeviceBusyValueType.cs

メンバー

メンバー	説明
NotSupported	サポートしていません。
NotBusy	デバイスはビジーではありません。
Busy	デバイスはビジーです。

Hbm.Api.Common.Enums.DigitalValueType Enumeration

デジタル値の種類。

C++

```
enum DigitalValueType {
  Low = 0,
  High = 1
};
```

C#

```
public enum DigitalValueType {
  Low = 0,
  High = 1
}
```

Visual Basic

```
Public Enum DigitalValueType
  Low = 0
  High = 1
End Enum
```

ファイル

DigitalValueType.cs

メンバー

メンバー	説明
Low	Low(0またはオフ状態に相当)。
High	High(1またはオン状態に相当)。

Hbm.Api.Common.Enums.DirectionType Enumeration

デジタルIOコネクタの方向の種類。

C++

```
enum DirectionType {
  In = 0,
  Out = 1
};
```

C#

```
public enum DirectionType {
  In = 0,
  Out = 1
}
```

Visual Basic

```
Public Enum DirectionType
  In = 0
  Out = 1
End Enum
```

ファイル

DirectionType.cs

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Enums Namespace

Hbm.Api.Common.Enums.ErrorResetMode Enumeration

エラーの解決後にCANエラーまたは警告を削除する方法を定義。

C++

```
enum ErrorResetMode {
  NotSupported = -1,
  OnRead = 0,
  Delayed = 1
};
```

C#

```
public enum ErrorResetMode {
  NotSupported = -1,
  OnRead = 0,
  Delayed = 1
}
```

Visual Basic

```
Public Enum ErrorResetMode
  NotSupported = -1
  OnRead = 0
  Delayed = 1
End Enum
```

ファイル

ErrorResetMode.cs

メンバー

メンバー	説明
NotSupported	サポートしていません。
OnRead	エラーまたは警告は、ステータスが読み取られるまで持続します。
Delayed	エラーまたは警告は、エラーが解決された後、遅れて削除されます。

Hbm.Api.Common.Enums.FbgSpectrumType Enumeration

ファイバーブラッググレーティング関連デバイスのスペクトルタイプの列挙。

C++

```
enum FbgSpectrumType {
  Fiber = 1,
  Etalon = 2,
  GasCell = 3
};
```

C#

```
public enum FbgSpectrumType {
  Fiber = 1,
  Etalon = 2,
  GasCell = 3
}
```

Visual Basic

```
Public Enum FbgSpectrumType
  Fiber = 1
  Etalon = 2
  GasCell = 3
End Enum
```

ファイル

FbgSpectrumType.cs

メンバー

メンバー	説明
Fiber	ファイバのスペクトル。
Etalon	エタロンのスペクトル(内部使用のみ)。
GasCell	ガスセルのスペクトル(内部使用のみ)。

Hbm.Api.Common.Enums.LedFlashMode Enumeration

点滅させるLEDの色。

C++

```
enum LedFlashMode {
  Off = 0,
  Green = 1,
  Yellow = 2,
  Red = 3
};
```

C#

```
public enum LedFlashMode {
  Off = 0,
  Green = 1,
  Yellow = 2,
  Red = 3
}
```

Visual Basic

```
Public Enum LedFlashMode
  Off = 0
  Green = 1
  Yellow = 2
  Red = 3
End Enum
```

ファイル

LedFlashMode.cs

メンバー

メンバー	説明
Off	点滅をオフにします。
Green	緑色に点滅します。
Yellow	黄色に点滅します。
Red	赤色に点滅します。

Hbm.Api.Common.Enums.MeasurementValueState Enumeration

測定値のステータス。

C++

```
[Flags]
enum MeasurementValueState {
  Valid  = 0,
  Overflow  = 1
};
```

C#

```
[Flags]
public enum MeasurementValueState {
  Valid  = 0,
  Overflow  = 1
}
```

Visual Basic

```
<Flags>
Public Enum MeasurementValueState
  Valid  = 0
  Overflow  = 1
End Enum
```

ファイル

MeasurementValueState.cs

メンバー

メンバー	説明
Valid	測定値は有効です。
Overflow	測定値がオーバーフローしました。

Hbm.Api.Common.Enums.
NetworkSettingsConfiguration Enumeration

ネットワーク設定の構成。

C++

```
enum NetworkSettingsConfiguration {
  Dhcp = 1,
  Static = 2
};
```

C#

```
public enum NetworkSettingsConfiguration {
  Dhcp = 1,
  Static = 2
}
```

Visual Basic

```
Public Enum NetworkSettingsConfiguration
  Dhcp = 1
  Static = 2
End Enum
```

ファイル

NetworkSettingsConfiguration.cs

メンバー

メンバー	説明
Dhcp	DHCP 構成。
Static	静的構成。

Hbm.Api.Common.Enums.OutputScalingType Enumeration

アナログ出力コネクタの出力スケーリングの種類。

C++

```
enum OutputScalingType {
  NotSupported,
  Table,
  Off
};
```

C#

```
public enum OutputScalingType {
  NotSupported,
  Table,
  Off
}
```

Visual Basic

```
Public Enum OutputScalingType
  NotSupported
  Table
  Off
End Enum
```

ファイル

OutputScalingType.cs

メンバー

メンバー	説明
NotSupported	スケーリングをサポートしていません。
Table	テーブルスケーリング。
Off	スケーリングを使用しません(オフ)。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Enums Namespace

Hbm.Api.Common.Enums.ReleaseType Enumeration

要求した校正証明に必要な種類を決定します。

C++

```
enum ReleaseType {
  All,
  Latest
};
```

C#

```
public enum ReleaseType {
  All,
  Latest
}
```

Visual Basic

```
Public Enum ReleaseType
  All
  Latest
End Enum
```

ファイル

ReleaseType.cs

メンバー

メンバー	説明
All	すべてのリリース。
Latest	最新リリースのみ。

Hbm.Api.Common.Enums.SettingType Enumeration

設定を適応したオブジェクトの種類。

C++

```
[AdditionalValueType(typeof(string))]
enum SettingType {
  Device,
  Connector,
  Channel,
  Sensor,
  Signal,
  Filter,
  Zero,
  Scaling,
  AdditionalFeature,
  ConnectionInfo,
  TimeSource,
  DataRateDomain,
  TedsUsageMode,
  ExtendedSampleRateMode
};
```

C#

```
[AdditionalValueType(typeof(string))]
public enum SettingType {
  Device,
  Connector,
  Channel,
  Sensor,
  Signal,
  Filter,
  Zero,
  Scaling,
  AdditionalFeature,
  ConnectionInfo,
  TimeSource,
  DataRateDomain,
  TedsUsageMode,
  ExtendedSampleRateMode
}
```

Visual Basic

```
<AdditionalValueType(typeof(string))>
Public Enum SettingType
  Device
  Connector
  Channel
  Sensor
  Signal
  Filter
  Zero
  Scaling
  AdditionalFeature
  ConnectionInfo
  TimeSource
  DataRateDomain
  TedsUsageMode
  ExtendedSampleRateMode
End Enum
```

ファイル

SettingType.cs

Hbm.Api.Common.Enums.ShuntMode Enumeration

シャント出力の構成を記述するモード。

C++

```
enum ShuntMode {
  NotSupported = -1,
  Disabled = 0,
  Enabled = 1
};
```

C#

```
public enum ShuntMode {
  NotSupported = -1,
  Disabled = 0,
  Enabled = 1
}
```

Visual Basic

```
Public Enum ShuntMode
  NotSupported = -1
  Disabled = 0
  Enabled = 1
End Enum
```

ファイル

ShuntMode.cs

メンバー

メンバー	説明
NotSupported	シャント出力はこのコネクタではサポートしていません(別の値に設定すると、割り当て中にHbm.Api.Common.Entities.Problems.ValueNotSetErrorが発生します)。
Disabled	シャントはオフです。
Enabled	シャントはオンです。

Hbm.Api.Common.Enums.StatusType Enumeration

デバイス/コネクタ/チャネルのステータスの種類(すべてのデバイスがすべてのステータスをサポートしているわけではありません)。

C++

```
[Flags]
enum StatusType {
  None = 0x0,
  Overflow = 0x1,
  Teds = 0x2,
  Connected = 0x4,
  ParamBusy = 0x8,
  ConnectorShunt = 0x10,
  TestSignal = 0x20,
  DeviceBusy = 0x40,
  All = 0xffff
};
```

C#

```
[Flags]
public enum StatusType {
  None = 0x0,
  Overflow = 0x1,
  Teds = 0x2,
  Connected = 0x4,
  ParamBusy = 0x8,
  ConnectorShunt = 0x10,
  TestSignal = 0x20,
  DeviceBusy = 0x40,
  All = 0xffff
}
```

Visual Basic

```
<Flags>
Public Enum StatusType
  None = &H0
  Overflow = &H1
  Teds = &H2
  Connected = &H4
  ParamBusy = &H8
  ConnectorShunt = &H10
  TestSignal = &H20
  DeviceBusy = &H40
  All = &Hffff
End Enum
```

ファイル

StatusType.cs

メンバー

メンバー	説明
None	ステータスの種類を設定していません。
Overflow	オーバーフローステータス。
Teds	TEDSステータス。
Connected	接続ステータス(センサ接続)。
ParamBusy	パラメータのビジーステータス。
ConnectorShunt	コネクタシャント。
TestSignal	テスト信号。
DeviceBusy	デバイスのビジーステータス。
All	すべてのステータス。

Hbm.Api.Common.Enums.SynchronousMode Enumeration

同期モードは信号の測定配信の種類を表します。

C++

```
enum SynchronousMode {
  Equidistant = 0,
  NonEquidistant = 1
};
```

C#

```
public enum SynchronousMode {
  Equidistant = 0,
  NonEquidistant = 1
}
```

Visual Basic

```
Public Enum SynchronousMode
  Equidistant = 0
  NonEquidistant = 1
End Enum
```

ファイル

SynchronousMode.cs

メンバー

メンバー	説明
Equidistant	信号の測定値のタイムスタンプはすべて同じタイムデルタを持ちます。
NonEquidistant	信号の測定値のタイムスタンプには、さまざまな予測不可能なタイムデルタが含まれます。これは、非同期CANセンサの測定値の典型的な現象です。

Hbm.Api.Common.Enums.SyncModeType Enumeration

同期モードの種類。

C++

```
enum SyncModeType {
  Standalone = 0,
  Slave = 1,
  Master = 2
};
```

C#

```
public enum SyncModeType {
  Standalone = 0,
  Slave = 1,
  Master = 2
}
```

Visual Basic

```
Public Enum SyncModeType
  Standalone = 0
  Slave = 1
  Master = 2
End Enum
```

ファイル

SyncModeType.cs

メンバー

メンバー	説明
Standalone	スタンドアロン測定。デバイスは他のデバイスとハードウェア同期していません。
Slave	測定の開始は他のデバイス(マスタとして機能している)によってトリガされます。
Master	このデバイスは、このデバイスファミリの他のデバイスの測定の開始をトリガします。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Enums Namespace

Hbm.Api.Common.Enums.TedsUsageMode Enumeration

センサのTEDS使用モードの値を定義。

C++

```
enum TedsUsageMode {
  NotSupported = -1,
  Ignore = 0,
  UseIfAvailable = 1,
  UseRequired = 2,
  UseManually = 100
};
```

C#

```
public enum TedsUsageMode {
  NotSupported = -1,
  Ignore = 0,
  UseIfAvailable = 1,
  UseRequired = 2,
  UseManually = 100
}
```

Visual Basic

```
Public Enum TedsUsageMode
  NotSupported = -1
  Ignore = 0
  UseIfAvailable = 1
  UseRequired = 2
  UseManually = 100
End Enum
```

ファイル

TedsUsageMode.cs

メンバー

メンバー	説明
NotSupported	TEDSの使用をサポートしていません。
Ignore	TEDSは完全に無視されます。
UseIfAvailable	TEDSが利用可能な場合は自動的に使用します。
UseRequired	TEDSを自動的および強制的にに使用します。
UseManually	TEDSを手動で使用します(アンプが自動化をサポートしていない場合)。

Hbm.Api.Common.Enums.TedsValidationValueType Enumeration

Entities.Statuses.ChannelTedsStatusのvalueパラメータの値を定義。

C++

```
enum TedsValidationValueType {
  NotSupported = -1,
  TedsNotAvailable = 0,
  TedsConnected = 1,
  TedsValid = 2,
  TedsChecksumError = 3,
  TedsDataEmpty = 4,
  TedsDataNotSupportedByConnector = 5,
  TedsDataParamBusy = 6,
  TedsDataNotUsed = 7,
  TedsValidRequestPressureEqualAmp = 8,
  TedsValidRequestPressureBelowAmp = 9,
  TedsValidRequestPressureAboveAmp = 10
};
```

C#

```
public enum TedsValidationValueType {
  NotSupported = -1,
  TedsNotAvailable = 0,
  TedsConnected = 1,
  TedsValid = 2,
  TedsChecksumError = 3,
  TedsDataEmpty = 4,
  TedsDataNotSupportedByConnector = 5,
  TedsDataParamBusy = 6,
  TedsDataNotUsed = 7,
  TedsValidRequestPressureEqualAmp = 8,
  TedsValidRequestPressureBelowAmp = 9,
  TedsValidRequestPressureAboveAmp = 10
}
```

Visual Basic

```
Public Enum TedsValidationValueType
  NotSupported = -1
  TedsNotAvailable = 0
  TedsConnected = 1
  TedsValid = 2
  TedsChecksumError = 3
  TedsDataEmpty = 4
  TedsDataNotSupportedByConnector = 5
  TedsDataParamBusy = 6
  TedsDataNotUsed = 7
  TedsValidRequestPressureEqualAmp = 8
  TedsValidRequestPressureBelowAmp = 9
  TedsValidRequestPressureAboveAmp = 10
End Enum
```

ファイル

TedsUsageMode.cs

メンバー

メンバー	説明
NotSupported	サポートしていません。
TedsNotAvailable	TEDSチップは使用できません。
TedsConnected	TEDSチップが接続されています(TIDは読み取り可能ですが、センサのTEDSパラメータに関する情報は認識されていません!)。
TedsValid	TEDSチップは利用可能であり、問題ありません。
TedsChecksumError	TEDSチップは利用可能ですが、チェックサムエラーが発生しています(破損しているか、空である可能性があります)。
TedsDataEmpty	TEDSチップは利用可能ですが、空です(データがありません)。
TedsDataNotSupportedByConnector	TEDSチップは使用できますが、TEDSデータはコネクタでサポートされていません。
TedsDataParamBusy	TEDSチップは利用可能ですが、TEDSパラメータの検証がビジー状態です(お待ちください)。
TedsDataNotUsed	TEDSチップは存在し、空でもなく、破損していませんが、使用されていません。
TedsValidRequestPressureEqualAmp	MX590のみ: TEDSの圧力設定はセンサの仕様と同じです(有効)。
TedsValidRequestPressureBelowAmp	MX590のみ: TEDSの圧力設定がセンサの仕様を下回っています(使用中のアンプのサブレンジのみ)。
TedsValidRequestPressureAboveAmp	MX590のみ: TEDSの圧力設定がセンサの仕様を上回っています(アンプの損傷の可能性があります)。

Hbm.Api.Common.Enums.TransportMode Enumeration

デバイスの同期に使用するPTPネットワークトランスポートプロトコル。

C++

```
enum TransportMode {
  IEEE = 0,
  UdpIpV4 = 1,
  UdpIpV6 = 2
};
```

C#

```
public enum TransportMode {
  IEEE = 0,
  UdpIpV4 = 1,
  UdpIpV6 = 2
}
```

Visual Basic

```
Public Enum TransportMode
  IEEE = 0
  UdpIpV4 = 1
  UdpIpV6 = 2
End Enum
```

ファイル

TransportMode.cs

メンバー

メンバー	説明
IEEE	IEEEプロトコル。
UdpIpV4	UDP IPv4。
UdpIpV6	UDP IPv6。

Hbm.Api.Common.Exceptions Namespace

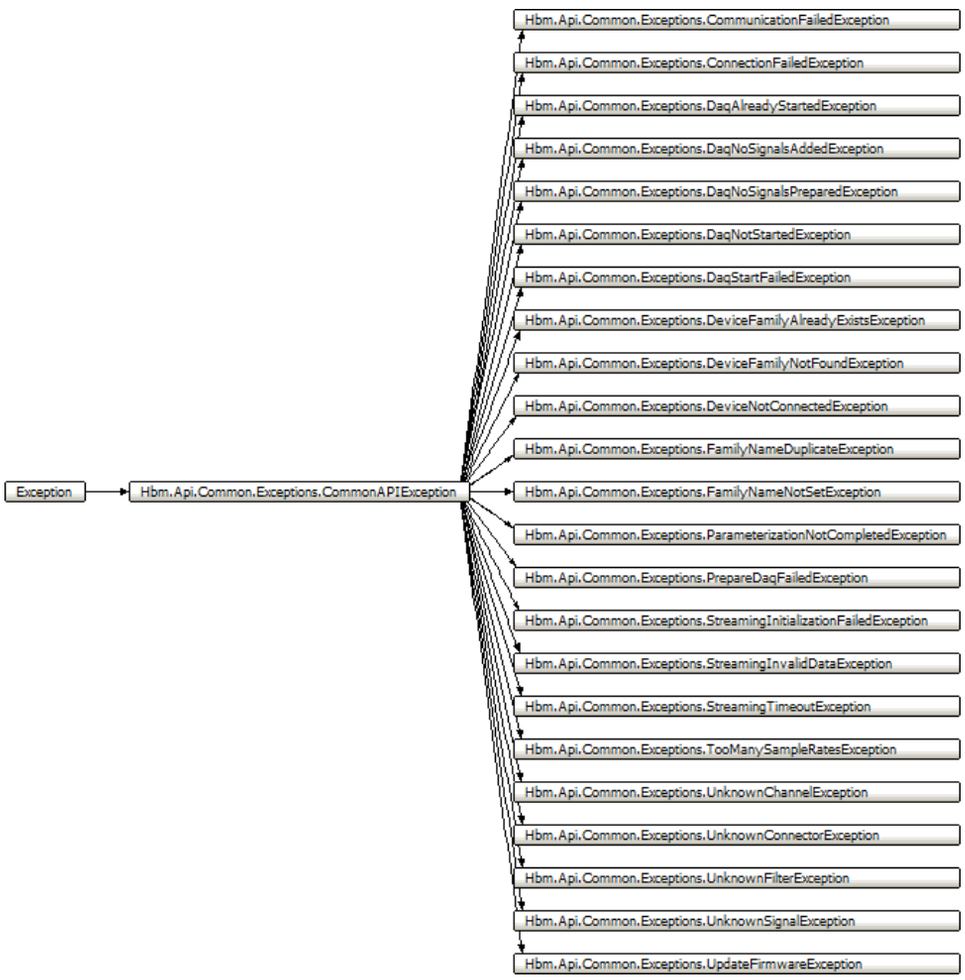
これは
Hbm.Api.Common.Exceptionsです。

クラス

	名前	説明
A	CommonAPIException	Common APIのカスタム例外の基本クラス。
	CommunicationFailedException	デバイスとの通信が失敗した場合にスローされます。
	ConnectionFailedException	デバイスへの接続を確立できなかった場合にスローされます。
	DaqAlreadyStartedException	データの取得中では実行できない関数をユーザが呼び出した場合にスローされます(例: DaqMeasurement.StartDaq、DaqMeasurement.PrepareDaq、DaqMeasurement.RegisterForMeasurementなど)。
	DaqNoSignalsAddedException	ユーザが信号を追加(`AddSignals`)せずにPrepareDaqを呼び出した場合にスローされます。
	DaqNoSignalsPreparedException	ユーザが登録(`RegisterForMeasurement`)および準備(`PrepareDaq`信号)を行わずに`StartDaq`を呼び出した場合にスローされます。
	DaqNotStartedException	データ取得の実行を必要とする関数(DaqMeasurement.StopDaqやDaqMeasurement.FillMeasurementValuesなど)をユーザが実行しようとした場合にスローされます。
	DaqStartFailedException	測定の開始中にエラーが発生した場合にスローされます。
	DeviceFamilyAlreadyExistsException	ユーザが既に登録されているデバイスファミリを登録しようとした場合にスローされます。
	DeviceFamilyNotFoundException	要求したデバイスファミリが見つからなかった場合にスローされます。
	DeviceNotConnectedException	アプリケーションが切断されたデバイスと通信したい場合にスローされます。
	FamilyNameDuplicateException	指定したデバイスファミリがすでに登録されている場合にスローされます。
	FamilyNameNotSetException	デバイスファミリを登録するためのファミリ名がない場合にスローされます。
	ParameterizationNotCompletedException	測定の開始時にデバイス内でパラメータ化がまだ進行中の場合にスローされます。
	PrepareDaqFailedException	DaqMeasurement.PrepareDaq()による測定の準備中にエラーが発生した場合にスローされます。
	StreamingInitializationFailedException	ストリーミングの初期化中にエラーが発生した場合にスローされます。
	StreamingInvalidDataException	ストリーミング中に無効なデータが返された場合にスローされます。
	StreamingTimeoutException	ストリーミング中にタイムアウトが発生した場合にスローされます。
	TooManySampleRatesException	単一デバイスに対して要求された異なるサンプルレートの数がサポートされていない場合にスローされます。
	UnknownChannelException	チャネルが特定のデバイスで認識されない場合にスローされます。
	UnknownConnectorException	コネクタが特定のデバイスで認識されない場合にスローされます。
	UnknownFilterException	フィルタが特定のデバイスで認識されない場合にスローされます。
	UnknownSignalException	信号が特定のデバイスで認識されない場合にスローされます。
	UpdateFirmwareException	デバイスのファームウェアを更新するときに発生した例外を処理するクラス。

CommonAPIException Class

Common APIのカスタム例外の基本クラス。



C++

```
[Serializable]
class CommonAPIException : public Exception;
```

C#

```
[Serializable]
public abstract class CommonAPIException : Exception;
```

Visual Basic

```
<Serializable>
Public abstract Class CommonAPIException
Inherits Exception
```

ファイル

CommonAPIException.cs

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Exceptions Namespace

CommonAPIException Members

次の表は、CommonAPIExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。

CommonAPIException Constructor

これは、CommonAPIExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	CommonAPIException.CommonAPIException ()	新しいインスタンスを作成します。
	CommonAPIException.CommonAPIException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	CommonAPIException.CommonAPIException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

CommonAPIException.CommonAPIException Constructor ()

新しいインスタンスを作成します。

C++

```
CommonAPIException();
```

C#

```
public CommonAPIException();
```

Visual Basic

```
Public Sub New()
```

CommonAPIException.CommonAPIException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
CommonAPIException(
    string message
);
```

C#

```
public CommonAPIException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

CommonAPIException.CommonAPIException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
CommonAPIException(
    string message,
    System.Exception inner
);
```

C#

```
public CommonAPIException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

CommunicationFailedException Class

デバイスとの通信が失敗した場合にスローされます。



C++

```
[Serializable]
class CommunicationFailedException : public CommonAPIException;
```

C#

```
[Serializable]
public class CommunicationFailedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class CommunicationFailedException
Inherits CommonAPIException
```

ファイル

CommunicationFailedException.cs

CommunicationFailedException Members

次の表は、CommunicationFailedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	CommunicationFailedException	これは、CommunicationFailedExceptionコンストラクタの多重定義の概要です。

CommunicationFailedException Constructor

これは、CommunicationFailedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	CommunicationFailedException. CommunicationFailedException ()	新しいインスタンスを作成します。
	CommunicationFailedException. CommunicationFailedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	CommunicationFailedException. CommunicationFailedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

CommunicationFailedException.CommunicationFailedException Constructor ()

新しいインスタンスを作成します。

C++

```
CommunicationFailedException();
```

C#

```
public CommunicationFailedException();
```

Visual Basic

```
Public Sub New()
```

CommunicationFailedException.CommunicationFailedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
CommunicationFailedException(
    string message
);
```

C#

```
public CommunicationFailedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

CommunicationFailedException.CommunicationFailedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
CommunicationFailedException(
    string message,
    Exception inner
);
```

C#

```
public CommunicationFailedException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

ConnectionFailedException Class

デバイスへの接続を確立できなかった場合にスローされます。



C++

```
[Serializable]
class ConnectionFailedException : public CommonAPIException;
```

C#

```
[Serializable]
public class ConnectionFailedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class ConnectionFailedException
Inherits CommonAPIException
```

ファイル

ConnectionFailedException.cs

ConnectionFailedException Members

次の表は、ConnectionFailedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	ConnectionFailedException	これは、ConnectionFailedExceptionコンストラクタの多重定義の概要です。

ConnectionFailedException Constructor

これは、ConnectionFailedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	ConnectionFailedException. ConnectionFailedException ()	新しいインスタンスを作成します。
	ConnectionFailedException. ConnectionFailedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	ConnectionFailedException. ConnectionFailedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

ConnectionFailedException.ConnectionFailedException Constructor ()

新しいインスタンスを作成します。

C++

```
ConnectionFailedException();
```

C#

```
public ConnectionFailedException();
```

Visual Basic

```
Public Sub New()
```

ConnectionFailedException.ConnectionFailedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
ConnectionFailedException(
    string message
);
```

C#

```
public ConnectionFailedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

ConnectionFailedException.ConnectionFailedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
ConnectionFailedException(
    string message,
    Exception inner
);
```

C#

```
public ConnectionFailedException(
    string message,
    Exception inner
);
```

Visual Basic

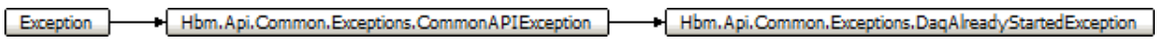
```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

DaqAlreadyStartedException Class

データの取得中では実行できない関数をユーザが呼び出した場合にスローされます(例: DaqMeasurement.StartDaq、DaqMeasurement.PrepareDaq、DaqMeasurement.RegisterForMeasurementなど)。



C++

```
[Serializable]
class DaqAlreadyStartedException : public CommonAPIException;
```

C#

```
[Serializable]
public class DaqAlreadyStartedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class DaqAlreadyStartedException
Inherits CommonAPIException
```

ファイル

DaqAlreadyStartedException.cs

DaqAlreadyStartedException Members

次の表は、DaqAlreadyStartedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	DaqAlreadyStartedException	これは、DaqAlreadyStartedExceptionコンストラクタの多重定義の概要です。

DaqAlreadyStartedException Constructor

これは、ConnectionFailedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqAlreadyStartedException. DaqAlreadyStartedException ()	新しいインスタンスを作成します。
	DaqAlreadyStartedException. DaqAlreadyStartedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	DaqAlreadyStartedException. DaqAlreadyStartedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

DaqAlreadyStartedException.DaqAlreadyStartedException Constructor ()

新しいインスタンスを作成します。

C++

```
DaqAlreadyStartedException();
```

C#

```
public DaqAlreadyStartedException();
```

Visual Basic

```
Public Sub New()
```

DaqAlreadyStartedException.DaqAlreadyStartedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
DaqAlreadyStartedException(
    string message
);
```

C#

```
public DaqAlreadyStartedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

DaqAlreadyStartedException.DaqAlreadyStartedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
DaqAlreadyStartedException(
    string message,
    Exception inner
);
```

C#

```
public DaqAlreadyStartedException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

DaqNoSignalsAddedException Class

ユーザが信号を追加(AddSignals)せずにPrepareDaqを呼び出した場合にスローされます。



C++

```
[Serializable]
class DaqNoSignalsAddedException : public CommonAPIException;
```

C#

```
[Serializable]
public class DaqNoSignalsAddedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class DaqNoSignalsAddedException
Inherits CommonAPIException
```

ファイル

DaqNoSignalsAddedException.cs

DaqNoSignalsAddedException Members

次の表は、DaqNoSignalsAddedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	DaqNoSignalsAddedException	これは、DaqNoSignalsAddedExceptionコンストラクタの多重定義の概要です。

DaqNoSignalsAddedException Constructor

これは、DaqNoSignalsAddedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqNoSignalsAddedException. DaqNoSignalsAddedException ()	新しいインスタンスを作成します。
	DaqNoSignalsAddedException. DaqNoSignalsAddedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	DaqNoSignalsAddedException. DaqNoSignalsAddedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

DaqNoSignalsAddedException.DaqNoSignalsAddedException Constructor ()

新しいインスタンスを作成します。

C++

```
DaqNoSignalsAddedException();
```

C#

```
public DaqNoSignalsAddedException();
```

Visual Basic

```
Public Sub New()
```

DaqNoSignalsAddedException.DaqNoSignalsAddedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
DaqNoSignalsAddedException(
    string message
);
```

C#

```
public DaqNoSignalsAddedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

DaqNoSignalsAddedException.DaqNoSignalsAddedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
DaqNoSignalsAddedException(
    string message,
    Exception inner
);
```

C#

```
public DaqNoSignalsAddedException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

DaqNoSignalsPreparedException Class

ユーザが登録(RegisterForMeasurement)および準備(PrepareDaq信号)を行わずにStartDaqを呼び出した場合にスローされます。



C++

```
[Serializable]
class DaqNoSignalsPreparedException : public CommonAPIException;
```

C#

```
[Serializable]
public class DaqNoSignalsPreparedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class DaqNoSignalsPreparedException
Inherits CommonAPIException
```

ファイル

DaqNoSignalsPreparedException.cs

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Exceptions Namespace

DaqNoSignalsPreparedException Members

次の表は、DaqNoSignalsPreparedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	DaqNoSignalsPreparedException	これは、DaqNoSignalsPreparedExceptionコンストラクタの多重定義の概要です。

DaqNoSignalsPreparedException Constructor

これは、DaqNoSignalsPreparedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqNoSignalsPreparedException. DaqNoSignalsPreparedException ()	新しいインスタンスを作成します。
	DaqNoSignalsPreparedException. DaqNoSignalsPreparedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	DaqNoSignalsPreparedException. DaqNoSignalsPreparedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

DaqNoSignalsPreparedException.DaqNoSignalsPreparedException Constructor ()

新しいインスタンスを作成します。

C++

```
DaqNoSignalsPreparedException();
```

C#

```
public DaqNoSignalsPreparedException();
```

Visual Basic

```
Public Sub New()
```

DaqNoSignalsPreparedException.DaqNoSignalsPreparedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
DaqNoSignalsPreparedException(
    string message
);
```

C#

```
public DaqNoSignalsPreparedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

DaqNoSignalsPreparedException.DaqNoSignalsPreparedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
DaqNoSignalsPreparedException(
    string message,
    Exception inner
);
```

C#

```
public DaqNoSignalsPreparedException(
    string message,
    Exception inner
);
```

Visual Basic

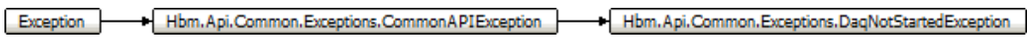
```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

DaqNotStartedException Class

データ取得の実行を必要とする関数(DaqMeasurement.StopDaqやDaqMeasurement.FillMeasurementValuesなど)をユーザが実行しようとした場合にスローされます。



C++

```
[Serializable]
class DaqNotStartedException : public CommonAPIException;
```

C#

```
[Serializable]
public class DaqNotStartedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class DaqNotStartedException
Inherits CommonAPIException
```

ファイル

DaqNotStartedException.cs

DaqNotStartedException Members

次の表は、DaqNotStartedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	DaqNotStartedException	これは、DaqNotStartedExceptionコンストラクタの多重定義の概要です。

DaqNotStartedException Constructor

これは、DaqNotStartedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqNotStartedException. DaqNotStartedException ()	新しいインスタンスを作成します。
	DaqNotStartedException. DaqNotStartedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	DaqNotStartedException. DaqNotStartedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

DaqNotStartedException.DaqNotStartedException Constructor ()

新しいインスタンスを作成します。

C++

```
DaqNotStartedException();
```

C#

```
public DaqNotStartedException();
```

Visual Basic

```
Public Sub New()
```

DaqNotStartedException.DaqNotStartedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
DaqNotStartedException(
    string message
);
```

C#

```
public DaqNotStartedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

DaqNotStartedException.DaqNotStartedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
DaqNotStartedException(
    string message,
    Exception inner
);
```

C#

```
public DaqNotStartedException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

DaqStartFailedException Class

測定の開始中にエラーが発生した場合にスローされます。



C++

```
[Serializable]
class DaqStartFailedException : public CommonAPIException;
```

C#

```
[Serializable]
public class DaqStartFailedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class DaqStartFailedException
Inherits CommonAPIException
```

ファイル

DaqStartFailedException.cs

DaqStartFailedException Members

次の表は、DaqStartFailedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	DaqStartFailedException	これは、DaqStartFailedExceptionコンストラクタの多重定義の概要です。

DaqStartFailedException Constructor

これは、DaqStartFailedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	DaqStartFailedException. DaqStartFailedException ()	新しいインスタンスを作成します。
	DaqStartFailedException. DaqStartFailedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	DaqStartFailedException. DaqStartFailedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

DaqStartFailedException.DaqStartFailedException Constructor ()

新しいインスタンスを作成します。

C++

```
DaqStartFailedException();
```

C#

```
public DaqStartFailedException();
```

Visual Basic

```
Public Sub New()
```

DaqStartFailedException.DaqStartFailedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
DaqStartFailedException(
    string message
);
```

C#

```
public DaqStartFailedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Exceptions Namespace -> DaqStartFailedException Class -> DaqStartFailedException Constructor

DaqStartFailedException.DaqStartFailedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
DaqStartFailedException(
    string message,
    Exception inner
);
```

C#

```
public DaqStartFailedException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

DeviceFamilyAlreadyExistsException Class

ユーザが既に登録されているデバイスファミリを登録しようとした場合にスローされます。



C++

```
[Serializable]
class DeviceFamilyAlreadyExistsException : public CommonAPIException;
```

C#

```
[Serializable]
public class DeviceFamilyAlreadyExistsException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class DeviceFamilyAlreadyExistsException
Inherits CommonAPIException
```

ファイル

DeviceFamilyAlreadyExistsException.cs

DeviceFamilyAlreadyExistsException Members

次の表は、DeviceFamilyAlreadyExistsExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	DeviceFamilyAlreadyExistsException	これは、DeviceFamilyAlreadyExistsExceptionコンストラクタの多重定義の概要です。

DeviceFamilyAlreadyExistsException Constructor

これは、DeviceFamilyAlreadyExistsExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	DeviceFamilyAlreadyExistsException.DeviceFamilyAlreadyExistsException ()	新しいインスタンスを作成します。
	DeviceFamilyAlreadyExistsException.DeviceFamilyAlreadyExistsException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	DeviceFamilyAlreadyExistsException.DeviceFamilyAlreadyExistsException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

DeviceFamilyAlreadyExistsException.DeviceFamilyAlreadyExistsException Constructor ()

新しいインスタンスを作成します。

C++

```
DeviceFamilyAlreadyExistsException();
```

C#

```
public DeviceFamilyAlreadyExistsException();
```

Visual Basic

```
Public Sub New()
```

DeviceFamilyAlreadyExistsException.DeviceFamilyAlreadyExistsException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
DeviceFamilyAlreadyExistsException(
    string message
);
```

C#

```
public DeviceFamilyAlreadyExistsException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

DeviceFamilyAlreadyExistsException.DeviceFamilyAlreadyExistsException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
DeviceFamilyAlreadyExistsException(
    string message,
    Exception inner
);
```

C#

```
public DeviceFamilyAlreadyExistsException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

DeviceFamilyNotFoundException Class

要求したデバイスファミリが見つからなかった場合にスローされます。



C++

```
[Serializable]
class DeviceFamilyNotFoundException : public CommonAPIException;
```

C#

```
[Serializable]
public class DeviceFamilyNotFoundException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class DeviceFamilyNotFoundException
Inherits CommonAPIException
```

ファイル

DeviceFamilyNotFoundException.cs

DeviceFamilyNotFoundException Members

次の表は、DeviceFamilyNotFoundExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	DeviceFamilyNotFoundException	これは、DeviceFamilyNotFoundExceptionコンストラクタの多重定義の概要です。

DeviceFamilyNotFoundException Constructor

次の表は、DeviceFamilyNotFoundExceptionによって公開されるメンバーの一覧です。

多重定義リスト

	名前	説明
	DeviceFamilyNotFoundException. DeviceFamilyNotFoundException ()	新しいインスタンスを作成します。
	DeviceFamilyNotFoundException. DeviceFamilyNotFoundException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	DeviceFamilyNotFoundException. DeviceFamilyNotFoundException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

DeviceFamilyNotFoundException.DeviceFamilyNotFoundException Constructor ()

新しいインスタンスを作成します。

C++

```
DeviceFamilyNotFoundException();
```

C#

```
public DeviceFamilyNotFoundException();
```

Visual Basic

```
Public Sub New()
```

DeviceFamilyNotFoundException.DeviceFamilyNotFoundException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
DeviceFamilyNotFoundException(
    string message
);
```

C#

```
public DeviceFamilyNotFoundException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

DeviceFamilyNotFoundException.DeviceFamilyNotFoundException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
DeviceFamilyNotFoundException(
    string message,
    Exception inner
);
```

C#

```
public DeviceFamilyNotFoundException(
    string message,
    Exception inner
);
```

Visual Basic

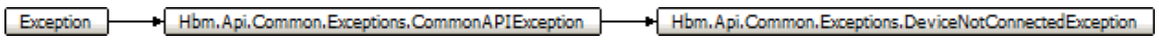
```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

DeviceNotConnectedException Class

アプリケーションが切断されたデバイスと通信したい場合にスローされます。



C++

```
[Serializable]
class DeviceNotConnectedException : public CommonAPIException;
```

C#

```
[Serializable]
public class DeviceNotConnectedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class DeviceNotConnectedException
Inherits CommonAPIException
```

ファイル

DeviceNotConnectedException.cs

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Exceptions Namespace

DeviceNotConnectedException Members

次の表は、DeviceNotConnectedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	DeviceNotConnectedException	これは、DeviceNotConnectedExceptionコンストラクタの多重定義の概要です。

DeviceNotConnectedException Constructor

これは、DeviceNotConnectedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	DeviceNotConnectedException. DeviceNotConnectedException ()	新しいインスタンスを作成します。
	DeviceNotConnectedException. DeviceNotConnectedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	DeviceNotConnectedException. DeviceNotConnectedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

DeviceNotConnectedException.DeviceNotConnectedException Constructor ()

新しいインスタンスを作成します。

C++

```
DeviceNotConnectedException();
```

C#

```
public DeviceNotConnectedException();
```

Visual Basic

```
Public Sub New()
```

DeviceNotConnectedException.DeviceNotConnectedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
DeviceNotConnectedException(
    string message
);
```

C#

```
public DeviceNotConnectedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

DeviceNotConnectedException.DeviceNotConnectedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
DeviceNotConnectedException(
    string message,
    Exception inner
);
```

C#

```
public DeviceNotConnectedException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

FamilyNameDuplicateException Class

指定したデバイスファミリがすでに登録されている場合にスローされます。



C++

```
[Serializable]
class FamilyNameDuplicateException : public CommonAPIException;
```

C#

```
[Serializable]
public class FamilyNameDuplicateException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class FamilyNameDuplicateException
Inherits CommonAPIException
```

ファイル

FamilyNameDuplicateException.cs

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Exceptions Namespace

FamilyNameDuplicateException Members

次の表は、FamilyNameDuplicateExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	FamilyNameDuplicateException	これは、FamilyNameDuplicateExceptionコンストラクタの多重定義の概要です。

FamilyNameDuplicateException Constructor

これは、FamilyNameDuplicateExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	FamilyNameDuplicateException. FamilyNameDuplicateException ()	新しいインスタンスを作成します。
	FamilyNameDuplicateException. FamilyNameDuplicateException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	FamilyNameDuplicateException. FamilyNameDuplicateException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

FamilyNameDuplicateException.FamilyNameDuplicateException Constructor ()

新しいインスタンスを作成します。

C++

```
FamilyNameDuplicateException();
```

C#

```
public FamilyNameDuplicateException();
```

Visual Basic

```
Public Sub New()
```

FamilyNameDuplicateException.FamilyNameDuplicateException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
FamilyNameDuplicateException(
    string message
);
```

C#

```
public FamilyNameDuplicateException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

FamilyNameDuplicateException.FamilyNameDuplicateException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
FamilyNameDuplicateException(
    string message,
    Exception inner
);
```

C#

```
public FamilyNameDuplicateException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

FamilyNameNotSetException Class

デバイスファミリを登録するためのファミリ名がない場合にスローされます。



C++

```
[Serializable]
class FamilyNameNotSetException : public CommonAPIException;
```

C#

```
[Serializable]
public class FamilyNameNotSetException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class FamilyNameNotSetException
Inherits CommonAPIException
```

ファイル

FamilyNameNotSetException.cs

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Exceptions Namespace

FamilyNameNotSetException Members

次の表は、FamilyNameNotSetExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	FamilyNameNotSetException	これは、FamilyNameNotSetExceptionコンストラクタの多重定義の概要です。

FamilyNameNotSetException Constructor

これは、FamilyNameNotSetExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	FamilyNameNotSetException. FamilyNameNotSetException ()	新しいインスタンスを作成します。
	FamilyNameNotSetException. FamilyNameNotSetException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	FamilyNameNotSetException. FamilyNameNotSetException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

FamilyNameNotSetException.FamilyNameNotSetException Constructor ()

新しいインスタンスを作成します。

C++

```
FamilyNameNotSetException();
```

C#

```
public FamilyNameNotSetException();
```

Visual Basic

```
Public Sub New()
```

FamilyNameNotSetException.FamilyNameNotSetException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
FamilyNameNotSetException(
    string message
);
```

C#

```
public FamilyNameNotSetException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

FamilyNameNotSetException.FamilyNameNotSetException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
FamilyNameNotSetException(
    string message,
    Exception inner
);
```

C#

```
public FamilyNameNotSetException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

ParameterizationNotCompletedException Class

測定の開始時にデバイス内でパラメータ化がまだ進行中の場合にスローされます。



C++

```
[Serializable]
class ParameterizationNotCompletedException : public CommonAPIException;
```

C#

```
[Serializable]
public class ParameterizationNotCompletedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class ParameterizationNotCompletedException
Inherits CommonAPIException
```

ファイル

ParameterizationNotCompletedException.cs

ParameterizationNotCompletedException Members

次の表は、ParameterizationNotCompletedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	ParameterizationNotCompletedException	これは、ParameterizationNotCompletedExceptionコンストラクタの多重定義の概要です。

ParameterizationNotCompletedException Constructor

これは、ParameterizationNotCompletedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	ParameterizationNotCompletedException.ParameterizationNotCompletedException ()	新しいインスタンスを作成します。
	ParameterizationNotCompletedException.ParameterizationNotCompletedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	ParameterizationNotCompletedException.ParameterizationNotCompletedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

ParameterizationNotCompletedException.ParameterizationNotCompletedException Constructor ()

新しいインスタンスを作成します。

C++

```
ParameterizationNotCompletedException();
```

C#

```
public ParameterizationNotCompletedException();
```

Visual Basic

```
Public Sub New()
```

ParameterizationNotCompletedException.ParameterizationNotCompletedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
ParameterizationNotCompletedException(
    string message
);
```

C#

```
public ParameterizationNotCompletedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

ParameterizationNotCompletedException.ParameterizationNotCompletedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
ParameterizationNotCompletedException(
    string message,
    Exception inner
);
```

C#

```
public ParameterizationNotCompletedException(
    string message,
    Exception inner
);
```

Visual Basic

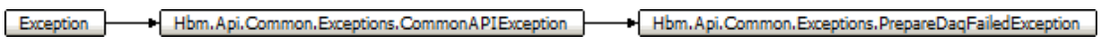
```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

PrepareDaqFailedException Class

DaqMeasurement.PrepareDaq()による測定の準備中にエラーが発生した場合にスローされます。



C++

```
[Serializable]
class PrepareDaqFailedException : public CommonAPIException;
```

C#

```
[Serializable]
public class PrepareDaqFailedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class PrepareDaqFailedException
Inherits CommonAPIException
```

ファイル

PrepareDaqFailedException.cs

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Exceptions Namespace

PrepareDaqFailedException Members

次の表は、PrepareDaqFailedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	PrepareDaqFailedException	これは、PrepareDaqFailedExceptionコンストラクタの多重定義の概要です。

PrepareDaqFailedException Constructor

これは、PrepareDaqFailedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	PrepareDaqFailedException. PrepareDaqFailedException ()	新しいインスタンスを作成します。
	PrepareDaqFailedException. PrepareDaqFailedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	PrepareDaqFailedException. PrepareDaqFailedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

PrepareDaqFailedException.PrepareDaqFailedException Constructor ()

新しいインスタンスを作成します。

C++

```
PrepareDaqFailedException();
```

C#

```
public PrepareDaqFailedException();
```

Visual Basic

```
Public Sub New()
```

PrepareDaqFailedException.PrepareDaqFailedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
PrepareDaqFailedException(
    string message
);
```

C#

```
public PrepareDaqFailedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

PrepareDaqFailedException.PrepareDaqFailedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
PrepareDaqFailedException(
    string message,
    Exception inner
);
```

C#

```
public PrepareDaqFailedException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

StreamingInitializationFailedException Class

ストリーミングの初期化中にエラーが発生した場合にスローされます。



C++

```
[Serializable]
class StreamingInitializationFailedException : public CommonAPIException;
```

C#

```
[Serializable]
public class StreamingInitializationFailedException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class StreamingInitializationFailedException
Inherits CommonAPIException
```

ファイル

StreamingInitializationFailedException.cs

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Exceptions Namespace

StreamingInitializationFailedException Members

次の表は、StreamingInitializationFailedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	StreamingInitializationFailedException	これは、StreamingInitializationFailedExceptionコンストラクタの多重定義の概要です。

StreamingInitializationFailedException Constructor

これは、StreamingInitializationFailedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	StreamingInitializationFailedException.StreamingInitializationFailedException ()	新しいインスタンスを作成します。
	StreamingInitializationFailedException.StreamingInitializationFailedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	StreamingInitializationFailedException.StreamingInitializationFailedException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

StreamingInitializationFailedException.StreamingInitializationFailedException Constructor ()

新しいインスタンスを作成します。

C++

```
StreamingInitializationFailedException();
```

C#

```
public StreamingInitializationFailedException();
```

Visual Basic

```
Public Sub New()
```

StreamingInitializationFailedException.StreamingInitializationFailedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
StreamingInitializationFailedException(
    string message
);
```

C#

```
public StreamingInitializationFailedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

StreamingInitializationFailedException.StreamingInitializationFailedException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
StreamingInitializationFailedException(
    string message,
    Exception inner
);
```

C#

```
public StreamingInitializationFailedException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

StreamingInvalidDataException Class

ストリーミング中に無効なデータが返された場合にスローされます。



C++

```
[Serializable]
class StreamingInvalidDataException : public CommonAPIException;
```

C#

```
[Serializable]
public class StreamingInvalidDataException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class StreamingInvalidDataException
Inherits CommonAPIException
```

ファイル

StreamingInvalidDataException.cs

StreamingInvalidDataException Members

次の表は、StreamingInvalidDataExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	StreamingInvalidDataException	これは、StreamingInvalidDataExceptionコンストラクタの多重定義の概要です。

StreamingInvalidDataException Constructor

これは、StreamingInvalidDataExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	StreamingInvalidDataException. StreamingInvalidDataException ()	新しいインスタンスを作成します。
	StreamingInvalidDataException. StreamingInvalidDataException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	StreamingInvalidDataException. StreamingInvalidDataException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

StreamingInvalidDataException.StreamingInvalidDataException Constructor ()

新しいインスタンスを作成します。

C++

```
StreamingInvalidDataException();
```

C#

```
public StreamingInvalidDataException();
```

Visual Basic

```
Public Sub New()
```

StreamingInvalidDataException.StreamingInvalidDataException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
StreamingInvalidDataException(
    string message
);
```

C#

```
public StreamingInvalidDataException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

StreamingInvalidDataException.StreamingInvalidDataException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
StreamingInvalidDataException(
    string message,
    Exception inner
);
```

C#

```
public StreamingInvalidDataException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

StreamingTimeoutException Class

ストリーミング中にタイムアウトが発生した場合にスローされます。



C++

```
[Serializable]
class StreamingTimeoutException : public CommonAPIException;
```

C#

```
[Serializable]
public class StreamingTimeoutException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class StreamingTimeoutException
Inherits CommonAPIException
```

ファイル

StreamingTimeoutException.cs

StreamingTimeoutException Members

次の表は、StreamingTimeoutExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	StreamingTimeoutException	これは、StreamingTimeoutExceptionコンストラクタの多重定義の概要です。

StreamingTimeoutException Constructor

これは、StreamingTimeoutExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	StreamingTimeoutException. StreamingTimeoutException ()	新しいインスタンスを作成します。
	StreamingTimeoutException. StreamingTimeoutException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	StreamingTimeoutException. StreamingTimeoutException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

StreamingTimeoutException.StreamingTimeoutException Constructor ()

新しいインスタンスを作成します。

C++

```
StreamingTimeoutException();
```

C#

```
public StreamingTimeoutException();
```

Visual Basic

```
Public Sub New()
```

StreamingTimeoutException.StreamingTimeoutException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
StreamingTimeoutException(
    string message
);
```

C#

```
public StreamingTimeoutException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

StreamingTimeoutException.StreamingTimeoutException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
StreamingTimeoutException(
    string message,
    Exception inner
);
```

C#

```
public StreamingTimeoutException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

TooManySampleRatesException Class

単一デバイスに対して要求された異なるサンプルレートの数がサポートされていない場合にスローされます。



C++

```
[Serializable]
class TooManySampleRatesException : public CommonAPIException;
```

C#

```
[Serializable]
public class TooManySampleRatesException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class TooManySampleRatesException
Inherits CommonAPIException
```

ファイル

TooManySampleRatesException.cs

TooManySampleRatesException Members

次の表は、TooManySampleRatesExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	TooManySampleRatesException	これは、TooManySampleRatesExceptionコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	DeviceFamily	デバイスファミリ名。
	DeviceName	デバイスの名前。
	DeviceUniqueId	デバイスの一意の識別子。
	MaxSupprtedSampleRates	サポートされている最大サンプルレートを取得または設定します。

TooManySampleRatesException Constructor

これは、TooManySampleRatesExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	TooManySampleRatesException. TooManySampleRatesException ()	新しいインスタンスを作成します。
	TooManySampleRatesException. TooManySampleRatesException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	TooManySampleRatesException. TooManySampleRatesException (string, string, string, string, int)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

TooManySampleRatesException.TooManySampleRatesException Constructor ()

新しいインスタンスを作成します。

C++

```
TooManySampleRatesException();
```

C#

```
public TooManySampleRatesException();
```

Visual Basic

```
Public Sub New()
```

TooManySampleRatesException.TooManySampleRatesException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
TooManySampleRatesException(
    string message
);
```

C#

```
public TooManySampleRatesException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

TooManySampleRatesException.TooManySampleRatesException Constructor (string, string, string, string, int)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
TooManySampleRatesException(
    string message,
    string familyName,
    string uniqueId,
    string deviceName,
    int maxSupportedSampleRates
);
```

C#

```
public TooManySampleRatesException(
    string message,
    string familyName,
    string uniqueId,
    string deviceName,
    int maxSupportedSampleRates
);
```

Visual Basic

```
Public Sub New(
    message As string,
    familyName As string,
    uniqueId As string,
    deviceName As string,
    maxSupportedSampleRates As Integer
)
```

引数

引数	説明
message	エラーメッセージ。
familyName	デバイスファミリ名。
uniqueId	デバイスの一意の 識別子。
deviceName	デバイスの名前。
maxSupportedSampleRates	サポートされている最大サンプルレート。

TooManySampleRatesException Properties

これは、TooManySampleRatesExceptionクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DeviceFamily	デバイスファミリ名。
	DeviceName	デバイスの名前。
	DeviceUniqueId	デバイスの一意の識別子。
	MaxSupprtedSampleRates	サポートされている最大サンプルレートを取得または設定します。

TooManySampleRatesException.DeviceFamily Property

デバイスファミリ名。

C++

```
__property string DeviceFamily;
```

C#

```
public string DeviceFamily;
```

Visual Basic

```
Public Property DeviceFamily() As string
```

TooManySampleRatesException.DeviceName Property

デバイスの名前。

C++

```
__property string DeviceName;
```

C#

```
public string DeviceName;
```

Visual Basic

```
Public Property DeviceName() As string
```

TooManySampleRatesException.DeviceUniqueId Property

デバイスの一意の識別子。

C++

```
__property string DeviceUniqueId;
```

C#

```
public string DeviceUniqueId;
```

Visual Basic

```
Public Property DeviceUniqueId() As string
```

TooManySampleRatesException.MaxSupprtedSampleRates Property

サポートされている最大サンプルレートを取得または設定します。

C++

```
__property int MaxSupprtedSampleRates;
```

C#

```
public int MaxSupprtedSampleRates;
```

Visual Basic

```
Public Property MaxSupprtedSampleRates() As Integer
```

UnknownChannelException Class

チャネルが特定のデバイスで認識されない場合にスローされます。



C++

```
[Serializable]
class UnknownChannelException : public CommonAPIException;
```

C#

```
[Serializable]
public class UnknownChannelException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class UnknownChannelException
Inherits CommonAPIException
```

ファイル

UnknownChannelException.cs

UnknownChannelException Members

次の表は、UnknownChannelExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	UnknownChannelException	これは、UnknownChannelExceptionコンストラクタの多重定義の概要です。

UnknownChannelException Constructor

これは、UnknownChannelExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	UnknownChannelException. UnknownChannelException ()	新しいインスタンスを作成します。
	UnknownChannelException. UnknownChannelException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	UnknownChannelException. UnknownChannelException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

UnknownChannelException.UnknownChannelException Constructor ()

新しいインスタンスを作成します。

C++

```
UnknownChannelException();
```

C#

```
public UnknownChannelException();
```

Visual Basic

```
Public Sub New()
```

UnknownChannelException.UnknownChannelException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
UnknownChannelException(
    string message
);
```

C#

```
public UnknownChannelException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

UnknownChannelException.UnknownChannelException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
UnknownChannelException(
    string message,
    Exception inner
);
```

C#

```
public UnknownChannelException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

UnknownConnectorException Class

コネクタが特定のデバイスで認識されない場合にスローされます。



C++

```
[Serializable]
class UnknownConnectorException : public CommonAPIException;
```

C#

```
[Serializable]
public class UnknownConnectorException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class UnknownConnectorException
Inherits CommonAPIException
```

ファイル

UnknownConnectorException.cs

UnknownConnectorException Members

次の表は、UnknownConnectorExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	UnknownConnectorException	これは、UnknownConnectorExceptionコンストラクタの多重定義の概要です。

UnknownConnectorException Constructor

これは、UnknownConnectorExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	UnknownConnectorException. UnknownConnectorException ()	新しいインスタンスを作成します。
	UnknownConnectorException. UnknownConnectorException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	UnknownConnectorException. UnknownConnectorException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

UnknownConnectorException.UnknownConnectorException Constructor ()

新しいインスタンスを作成します。

C++

```
UnknownConnectorException();
```

C#

```
public UnknownConnectorException();
```

Visual Basic

```
Public Sub New()
```

UnknownConnectorException.UnknownConnectorException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
UnknownConnectorException(
    string message
);
```

C#

```
public UnknownConnectorException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

UnknownConnectorException.UnknownConnectorException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
UnknownConnectorException(
    string message,
    Exception inner
);
```

C#

```
public UnknownConnectorException(
    string message,
    Exception inner
);
```

Visual Basic

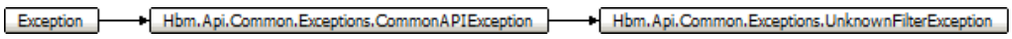
```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

UnknownFilterException Class

フィルタが特定のデバイスで認識されない場合にスローされます。



C++

```
[Serializable]
class UnknownFilterException : public CommonAPIException;
```

C#

```
[Serializable]
public class UnknownFilterException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class UnknownFilterException
Inherits CommonAPIException
```

ファイル

UnknownFilterException.cs

UnknownFilterException Members

次の表は、UnknownFilterExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	UnknownFilterException	これは、UnknownFilterExceptionコンストラクタの多重定義の概要です。

UnknownFilterException Constructor

これは、UnknownFilterExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	UnknownFilterException. UnknownFilterException ()	新しいインスタンスを作成します。
	UnknownFilterException. UnknownFilterException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	UnknownFilterException. UnknownFilterException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

UnknownFilterException.UnknownFilterException Constructor ()

新しいインスタンスを作成します。

C++

```
UnknownFilterException();
```

C#

```
public UnknownFilterException();
```

Visual Basic

```
Public Sub New()
```

UnknownFilterException.UnknownFilterException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
UnknownFilterException(
    string message
);
```

C#

```
public UnknownFilterException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

UnknownFilterException.UnknownFilterException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
UnknownFilterException(
    string message,
    Exception inner
);
```

C#

```
public UnknownFilterException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

UnknownSignalException Class

信号が特定のデバイスで認識されない場合にスローされます。



C++

```
[Serializable]
class UnknownSignalException : public CommonAPIException;
```

C#

```
[Serializable]
public class UnknownSignalException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public Class UnknownSignalException
Inherits CommonAPIException
```

ファイル

UnknownSignalException.cs

UnknownSignalException Members

次の表は、UnknownSignalExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	UnknownSignalException	これは、UnknownSignalExceptionコンストラクタの多重定義の概要です。

UnknownSignalException Constructor

これは、UnknownSignalExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	UnknownSignalException. UnknownSignalException ()	新しいインスタンスを作成します。
	UnknownSignalException. UnknownSignalException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	UnknownSignalException. UnknownSignalException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

UnknownSignalException.UnknownSignalException Constructor ()

新しいインスタンスを作成します。

C++

```
UnknownSignalException();
```

C#

```
public UnknownSignalException();
```

Visual Basic

```
Public Sub New()
```

UnknownSignalException.UnknownSignalException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
UnknownSignalException(
    string message
);
```

C#

```
public UnknownSignalException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

UnknownSignalException.UnknownSignalException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
UnknownSignalException(
    string message,
    Exception inner
);
```

C#

```
public UnknownSignalException(
    string message,
    Exception inner
);
```

Visual Basic

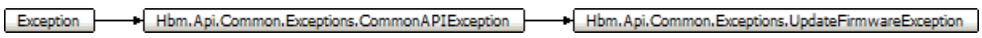
```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

UpdateFirmwareException Class

デバイスのファームウェアを更新するときに発生した例外を処理するクラス。



C++

```
class UpdateFirmwareException : public CommonAPIException;
```

C#

```
public class UpdateFirmwareException : CommonAPIException;
```

Visual Basic

```
Public Class UpdateFirmwareException
Inherits CommonAPIException
```

ファイル

UpdateFirmwareException.cs

UpdateFirmwareException Members

次の表は、UpdateFirmwareExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CommonAPIException	これは、CommonAPIExceptionコンストラクタの多重定義の概要です。
	UpdateFirmwareException	これは、UpdateFirmwareExceptionコンストラクタの多重定義の概要です。

UpdateFirmwareException Constructor

これは、UpdateFirmwareExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	UpdateFirmwareException. UpdateFirmwareException ()	新しいインスタンスを作成します。
	UpdateFirmwareException. UpdateFirmwareException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	UpdateFirmwareException. UpdateFirmwareException (string, Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

UpdateFirmwareException.UpdateFirmwareException Constructor ()

新しいインスタンスを作成します。

C++

```
UpdateFirmwareException();
```

C#

```
public UpdateFirmwareException();
```

Visual Basic

```
Public Sub New()
```

UpdateFirmwareException.UpdateFirmwareException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
UpdateFirmwareException(
    string message
);
```

C#

```
public UpdateFirmwareException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

UpdateFirmwareException.UpdateFirmwareException Constructor (string, Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
UpdateFirmwareException(
    string message,
    Exception inner
);
```

C#

```
public UpdateFirmwareException(
    string message,
    Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

Hbm.Api.Common.Messaging Namespace

Hbm.Api.Common.Messaging名前空間には、デバイスやチャネルの特定のステータス変更に関するさまざまなイベントを受信できるようにする関数が含まれています。

チャネルがオーバーフロー状態に切り替わったときなど、状態の変化について通知を受けるには、ステータスのタイプと監視するチャネルを指定する必要があります。

一部のデバイスでは、監視対象の特定の変更(例: PMXのTEDSステータスの変更)が発生した場合、全体的なパフォーマンスが大幅に低下する可能性があることに注意してください。

例

次のコードスニペットは、その方法を示しています:

- 特定のチャネルのステータス変更をサブスクライブする
- 受信したイベントを処理する
- ステータス監視のサブスクライブを解除する

```
...
// First scan and connect to any device
List<Problem> lastProblems;       // Used to return warnings and errors
DaqEnvironment daqEnvironment = DaqEnvironment.GetInstance(); // Determine all available
device families

List<Device> devices = daqEnvironment.Scan();          // Scan entire system for all
supported devices
daqEnvironment.Connect(devices[0], out lastProblems);   // Connect to first found device

// Check if the first channel of the device is an AnalogInChannel (status monitoring works
only for analog-in channels)
if (devices[0].Connectors[0].Channels[0] is AnalogInChannel)
{
  // Subscribe first  channel to monitor overflow state changes
  devices[0].SubscribeToStatusMonitoring(StatusType.Overflow, new List<AnalogInChannel>(){
devices[0].Connectors[0].Channels[0]as AnalogInChannel} );

  // also possible: subscribe more status or/and also more channels in one step
  // devices[0].SubscribeToStatusMonitoring(StatusType.Connected | StatusType.Overflow, new
List<AnalogInChannel>()
  //   { _commonDevice.Connectors[0].Channels[0]as AnalogInChannel,
  //   _commonDevice.Connectors[1].Channels[0]as AnalogInChannel,
  //   _commonDevice.Connectors[2].Channels[0]as AnalogInChannel} );


  ...
}
```

```
// Add an event handler for ChannelsStateChanged event
MessageBroker.ChannelsStateChanged += MessageBroker_ChannelsStateChanged;
```

```
...

// Event handler for ChannelsStateChanged events:
private void MessageBroker_ChannelsStateChanged(object sender, ChannelsStateChangedEventArgs
e)
{
  // Handle received event, e.g. here for first channel/status
  MessageBox.Show("Channel " + e.ChannelsStateChanges[0].ChannelUniqueId + ": Overflow = " +
e.ChannelsStateChanges[0].IsInOverflow.ToString());
}
```

```
...
// Anywhere later in your code
// Unsubscribe from status monitoring and remove event handler:
devices[0].UnsubscribeFromStatusMonitoring();
MessageBroker.ChannelsStateChanged -= MessageBroker_ChannelsStateChanged
...
```

クラス

名前	説明
ChannelsEventArgs	MessageBroker.ChannelNameChangedなどのチャネルステータスイベントのデータを提供します。
ChannelsStateChangedEventArgs	チャネルステータスイベントのデータを提供します。
ChannelsTedsStatusEventArgs	MessageBroker.ChannelTedsStatusChangedなど、チャネルのTEDSステータスイベントのデータを提供します。
DeviceBusyStateChangedEventArgs	デバイスのビジーステータスイベントのデータを提供します。
DeviceEventArgs	MessageBroker.DeviceConnectedなどのDeviceStatusイベントのデータを提供します。
DeviceNameEventArgs	DeviceNameChangedイベントにデータを提供します。
FiltersEventArgs	MessageBroker.FiltersChangedなどのフィルタイベントのデータを提供します。
FirmwareImageVersionCallbackEventArgs	イメージバージョンのコールバックがトリガされたときに発生するイベント。
FirmwareUpdateStatusEventArgs	デバイスの更新ステータスを通知するイベントクラス。
MessageBroker	完全なCommonApiのメッセージの中枢。
SampleRatesEventArgs	MessageBroker.SampleRatesChangedなどのサンプルレート変更イベントのデータを提供します。
SensorsEventArgs	MessageBroker.SensorsChangedなどのセンサイベントのデータを提供します。
SignalsEventArgs	MessageBroker.SignalNameChangedなどのシグナル ステータス イベントのデータを提供します。

名前空間

名前	説明
Hbm.Api.Common.Messaging.DataTransferObjects	これは、名前空間Hbm.Api.Common.Messaging.DataTransferObjectsです。

Hbm.Api.Common.Messaging.DataTransferObjects Namespace

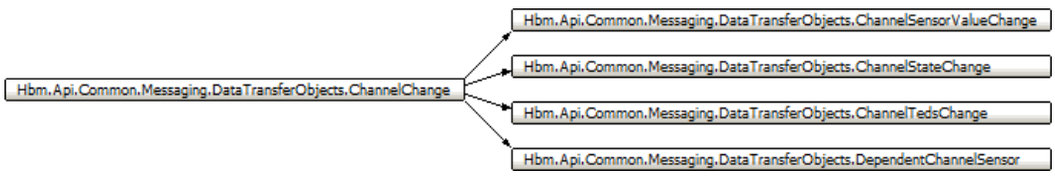
これは、名前空間Hbm.Api.Common.Messaging.DataTransferObjectsです。

クラス

	名前	説明
A	ChannelChange	チャネル状態のDtoクラス。
	ChannelSensorValueChange	チャネルセンサの変更を処理するデータ転送オブジェクトクラス。
	ChannelStateChange	チャネルステータスの変化を表すクラス。
	ChannelTedsChange	チャネルのTEDSステータス(独自のIDで識別)。
	DependentChannelSensor	センサ変更に依存するDtoクラス。
	DependentSignalFilter	チャネル依存のフィルタ。
	DependentSignalSampleRate	信号のサンプルレート依存を表すクラス。
	DeviceBusyStateChange	デバイスのビジーステータスの変化を表すクラス。
A	SignalChange	信号変更用のデータ転送オブジェクト クラス。
	SignalFilterChange	信号(およびすべての依存信号)のフィルタ変更を表すクラス。
	SignalSampleRateChange	信号リストのサンプルレートの変更を表すクラス。

ChannelChange Class

チャネル状態のDtoクラス。



C++

```
class ChannelChange;
```

C#

```
public abstract class ChannelChange;
```

Visual Basic

```
Public abstract Class ChannelChange
```

ファイル

ChannelChange.cs

ChannelChange Members

次の表は、ChannelChangeによって公開されるメンバーの一覧です。

Protected Methods

	名前	説明
	ChannelChange	ChannelChangeクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	ChannelUniqueId	チャネルの固有の識別子を取得します。

ChannelChange.ChannelChange Constructor

ChannelChangeクラスの新しいインスタンスを初期化します。

C++

```
ChannelChange(
    string channelId
);
```

C#

```
internal ChannelChange(
    string channelId
);
```

Visual Basic

```
Friend Sub New(
    channelId As string
)
```

引数

引数	説明
channelId	チャネルの識別子。

ChannelChange Properties

ChannelChangeクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ChannelUniqueId	チャネルの固有の識別子を取得します。

ChannelChange.ChannelUniqueId Property

チャネルの固有の識別子を取得します。

C++

```
__property string ChannelUniqueId;
```

C#

```
public string ChannelUniqueId;
```

Visual Basic

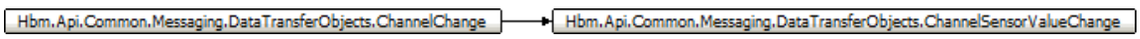
```
Public Property ChannelUniqueId() As string
```

説明

チャネル固有の識別子。

ChannelSensorValueChange Class

チャネルセンサの変更を処理するデータ転送オブジェクトクラス。



C++

```
class ChannelSensorValueChange : public ChannelChange;
```

C#

```
public class ChannelSensorValueChange : ChannelChange;
```

Visual Basic

```
Public Class ChannelSensorValueChange
Inherits ChannelChange
```

ファイル

ChannelSensorValueChange.cs

ChannelSensorValueChange Members

次の表は、ChannelSensorValueChangeによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ChannelSensorValueChange	ChannelSensorValueChangeクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	ChannelUniqueId	チャネルの固有の識別子を取得します。
	OldSensorsOfDependentChannels	チャネル依存の古いセンサを取得します。

ChannelSensorValueChange.ChannelSensorValueChange Constructor

ChannelSensorValueChangeクラスの新しいインスタンスを初期化します。

C++

```
ChannelSensorValueChange(
    string channelId,
    List<DependentChannelSensor> dependentChannelsOldSensors
);
```

C#

```
public ChannelSensorValueChange(
    string channelId,
    List<DependentChannelSensor> dependentChannelsOldSensors
);
```

Visual Basic

```
Public Sub New(
    channelId As string,
    dependentChannelsOldSensors As List<DependentChannelSensor>
)
```

引数

引数	説明
channelId	チャネル識別子。
dependentChannelsOldSensors	チャネル依存の古いセンサです。

ChannelSensorValueChange Properties

ChannelSensorValueChangeクラスのプロパティの一覧です。

Public Properties

	名前	説明
	OldSensorsOfDependentChannels	チャネル依存の古いセンサを取得します。

ChannelSensorValueChange.OldSensorsOfDependentChannels Property

チャネル依存の古いセンサを取得します。

C++

```
__property List<DependentChannelSensor> OldSensorsOfDependentChannels;
```

C#

```
public List<DependentChannelSensor> OldSensorsOfDependentChannels;
```

Visual Basic

```
Public Property OldSensorsOfDependentChannels() As List<DependentChannelSensor>
```

説明

チャネル依存の古いセンサ。

ChannelStateChange Class

チャネルステータスの変化を表すクラス。



C++

```
class ChannelStateChange : public ChannelChange;
```

C#

```
public class ChannelStateChange : ChannelChange;
```

Visual Basic

```
Public Class ChannelStateChange
Inherits ChannelChange
```

ファイル

ChannelStateChange.cs

ChannelStateChange Members

次の表は、ChannelStateChangeによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ChannelStateChange	T:Hbm.Api.Common.Messaging.DataTransferObjects.ChannelStateChangeクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	ChannelUniqueId	チャネルの固有の識別子を取得します。
	IsConnected	チャネルが接続されているかどうかを示す値を取得または設定します。
	IsInOverflow	チャネルがオーバーフローしているかどうかを示す値を取得または設定します。

ChannelStateChange.ChannelStateChange Constructor

T:Hbm.Api.Common.Messaging.DataTransferObjects.ChannelStateChangeクラスの新しいインスタンスを初期化します。

C++

```
ChannelStateChange(
    string channelId,
    bool? isConnected = null,
    bool? isInOverflow = null
);
```

C#

```
public ChannelStateChange(
    string channelId,
    bool? isConnected,
    bool? isInOverflow
);
```

Visual Basic

```
Public Sub New(
    channelId As string,
    isConnected As bool? = null,
    isInOverflow As bool? = null
)
```

引数

引数	説明
channelId	チャネル識別子。
isConnected	接続しているチャネルに対してtrueに設定します。
isInOverflow	オーバーフロー中のチャネルに対してtrueに設定します。

ChannelStateChange Properties

ChannelStateChangeクラスのプロパティの一覧です。

Public Properties

	名前	説明
	IsConnected	チャネルが接続されているかどうかを示す値を取得または設定します。
	IsInOverflow	チャネルがオーバーフローしているかどうかを示す値を取得または設定します。

ChannelStateChange.IsConnected Property

チャネルが接続されているかどうかを示す値を取得または設定します。

C++

```
__property bool? IsConnected;
```

C#

```
public bool? IsConnected;
```

Visual Basic

```
Public Property IsConnected() As bool?
```

説明

チャネルが接続されている場合は`true`。それ以外の場合は`false`。

現在位置: Hbm.Api.Common Namespace -> Hbm.Api.Common.Messaging Namespace -> Hbm.Api.Common.Messaging.DataTransferObjects Namespace -> ChannelStateChange Class -> ChannelStateChange.ChannelStateChange Constructor

ChannelStateChange.IsInOverflow Property

チャネルがオーバーフローしているかどうかを示す値を取得または設定します。

C++

```
__property bool? IsInOverflow;
```

C#

```
public bool? IsInOverflow;
```

Visual Basic

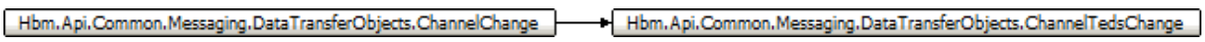
```
Public Property IsInOverflow() As bool?
```

説明

チャネルがオーバーフローしている場合は`true`。それ以外の場合は`false`。

ChannelTedsChange Class

チャネルのTEDSステータス(独自のIDで識別)。



C++

```
class ChannelTedsChange : public ChannelChange;
```

C#

```
public class ChannelTedsChange : ChannelChange;
```

Visual Basic

```
Public Class ChannelTedsChange
Inherits ChannelChange
```

ファイル

ChannelTedsChange.cs

ChannelTedsChange Members

次の表は、ChannelTedsChangeによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ChannelTedsChange	ChannelTedsChangeクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	ChannelUniqueId	チャネルの固有の識別子を取得します。
	TedsValidationValueType	TEDS検証値の型を取得します。

ChannelTedsChange.ChannelTedsChange Constructor

ChannelTedsChangeクラスの新しいインスタンスを初期化します。

C++

```
ChannelTedsChange(
    string channelId,
    TedsValidationValueType tedsValidationValueType
);
```

C#

```
public ChannelTedsChange(
    string channelId,
    TedsValidationValueType tedsValidationValueType
);
```

Visual Basic

```
Public Sub New(
    channelId As string,
    tedsValidationValueType As TedsValidationValueType
)
```

引数

引数	説明
channelId	チャネル識別子。
tedsValidationValueType	TEDS検証値の型。

ChannelTedsChange Properties

ChannelTedsChangeクラスのプロパティの一覧です。

Public Properties

	名前	説明
	TedsValidationValueType	TEDS検証値の型を取得します。

ChannelTedsChange.TedsValidationValueType Property

TEDS検証値の型を取得します。

C++

```
__property TedsValidationValueType TedsValidationValueType;
```

C#

```
public TedsValidationValueType TedsValidationValueType;
```

Visual Basic

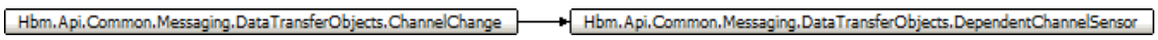
```
Public Property TedsValidationValueType() As TedsValidationValueType
```

説明

TEDS検証値の型。

DependentChannelSensor Class

センサ変更に依存するDtoクラス。



C++

```
class DependentChannelSensor : public ChannelChange;
```

C#

```
public class DependentChannelSensor : ChannelChange;
```

Visual Basic

```
Public Class DependentChannelSensor
Inherits ChannelChange
```

ファイル

DependentChannelSensor.cs

DependentChannelSensor Members

次の表は、DependentChannelSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DependentChannelSensor	DependentChannelSensorクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	ChannelSensor	サンプルレートを取得します。
	ChannelUniqueId	固有のチャネル識別子を取得します。

DependentChannelSensor.DependentChannelSensor Constructor

DependentChannelSensorクラスの新しいインスタンスを初期化します。

C++

```
DependentChannelSensor(
    string channelId,
    Sensor sensor
);
```

C#

```
public DependentChannelSensor(
    string channelId,
    Sensor sensor
);
```

Visual Basic

```
Public Sub New(
    channelId As string,
    sensor As Sensor
)
```

引数

引数	説明
channelId	チャネル識別子。
sensor	センサ。

DependentChannelSensor Properties

DependentChannelSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ChannelSensor	サンプルレートを取得します。

DependentChannelSensor.ChannelSensor Property

サンプルレートを取得します。

C++

```
__property Sensor ChannelSensor;
```

C#

```
public Sensor ChannelSensor;
```

Visual Basic

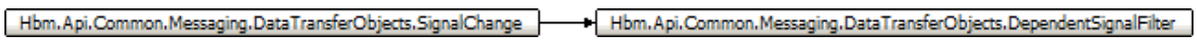
```
Public Property ChannelSensor() As Sensor
```

説明

サンプルレート。

DependentSignalFilter Class

チャネル依存のフィルタ。



C++

```
class DependentSignalFilter : public SignalChange;
```

C#

```
public class DependentSignalFilter : SignalChange;
```

Visual Basic

```
Public Class DependentSignalFilter
Inherits SignalChange
```

ファイル

DependentSignalFilter.cs

DependentSignalFilter Members

次の表は、DependentSignalFilterによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DependentSignalFilter	dependentSignalFilterクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	Filter	フィルタを取得します。
	UniqueSignalId	固有の信号識別子を取得します。

DependentSignalFilter.DependentSignalFilter Constructor

dependentSignalFilterクラスの新しいインスタンスを初期化します。

C++

```
DependentSignalFilter(
    string signalId,
    Filter signalFilter
);
```

C#

```
public DependentSignalFilter(
    string signalId,
    Filter signalFilter
);
```

Visual Basic

```
Public Sub New(
    signalId As string,
    signalFilter As Filter
)
```

引数

引数	説明
signalId	信号識別子。
signalFilter	信号フィルタ。

DependentSignalFilter Properties

DependentSignalFilterクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Filter	フィルタを取得します。

DependentSignalFilter.Filter Property

フィルタを取得します。

C++

```
__property Filter Filter;
```

C#

```
public Filter Filter;
```

Visual Basic

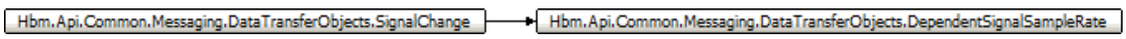
```
Public Property Filter() As Filter
```

説明

フィルタ。

DependentSignalSampleRate Class

信号のサンプルレート依存を表すクラス。



C++

```
class DependentSignalSampleRate : public SignalChange;
```

C#

```
public class DependentSignalSampleRate : SignalChange;
```

Visual Basic

```
Public Class DependentSignalSampleRate
Inherits SignalChange
```

ファイル

DependentSignalSampleRate.cs

DependentSignalSampleRate Members

次の表は、DependentSignalSampleRateによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DependentSignalSampleRate	DependentSignalSampleRateクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	SampleRate	サンプルレートを取得します。
	UniqueSignalId	固有の信号識別子を取得します。

DependentSignalSampleRate.DependentSignalSampleRate Constructor

DependentSignalSampleRateクラスの新しいインスタンスを初期化します。

C++

```
DependentSignalSampleRate(
    string signalId,
    decimal sampleRate
);
```

C#

```
public DependentSignalSampleRate(
    string signalId,
    decimal sampleRate
);
```

Visual Basic

```
Public Sub New(
    signalId As string,
    sampleRate As decimal
)
```

引数

引数	説明
signalId	信号識別子。
sampleRate	サンプルレート。

DependentSignalSampleRate Properties

DependentSignalSampleRateクラスのプロパティの一覧です。

Public Properties

	名前	説明
	SampleRate	サンプルレートを取得します。

DependentSignalSampleRate.SampleRate Property

サンプルレートを取得します。

C++

```
__property decimal SampleRate;
```

C#

```
public decimal SampleRate;
```

Visual Basic

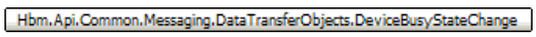
```
Public Property SampleRate() As decimal
```

説明

サンプルレート。

DeviceBusyStateChange Class

デバイスのビジーステータスの変化を表すクラス。



C++

```
class DeviceBusyStateChange;
```

C#

```
public class DeviceBusyStateChange;
```

Visual Basic

```
Public Class DeviceBusyStateChange
```

ファイル

DeviceBusyStateChange.cs

DeviceBusyStateChange Members

次の表は、DeviceBusyStateChangeによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DeviceBusyStateChange	T:Hbm.Api.Common.Messaging.DataTransferObjects. DeviceBusyStateChangeクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	DeviceUniqueId	デバイス固有の識別子を取得します。
	IsBusy	デバイスがビジーかどうかを示す値を取得または設定します。

DeviceBusyStateChange.DeviceBusyStateChange Constructor

T:Hbm.Api.Common.Messaging.DataTransferObjects.DeviceBusyStateChangeクラスの新しいインスタンスを初期化します。

C++

```
DeviceBusyStateChange(
    string deviceId,
    bool isBusy
);
```

C#

```
public DeviceBusyStateChange(
    string deviceId,
    bool isBusy
);
```

Visual Basic

```
Public Sub New(
    deviceId As string,
    isBusy  As bool
)
```

引数

引数	説明
deviceId	デバイス識別子。
isBusy	デバイスをビジーに設定する場合は`true`。

DeviceBusyStateChange Properties

DeviceBusyStateChangeクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DeviceUniqueId	デバイス固有の識別子を取得します。
	IsBusy	デバイスがビジーかどうかを示す値を取得または設定します。

DeviceBusyStateChange.DeviceUniqueId Property

デバイス固有の識別子を取得します。

C++

```
[PublicAPI]
__property string DeviceUniqueId;
```

C#

```
[PublicAPI]
public string DeviceUniqueId;
```

Visual Basic

```
<PublicAPI>
Public Property DeviceUniqueId() As string
```

説明

デバイス固有の識別子。

DeviceBusyStateChange.IsBusy Property

デバイスがビジーかどうかを示す値を取得または設定します。

C++

```
[PublicAPI]
__property bool IsBusy;
```

C#

```
[PublicAPI]
public bool IsBusy;
```

Visual Basic

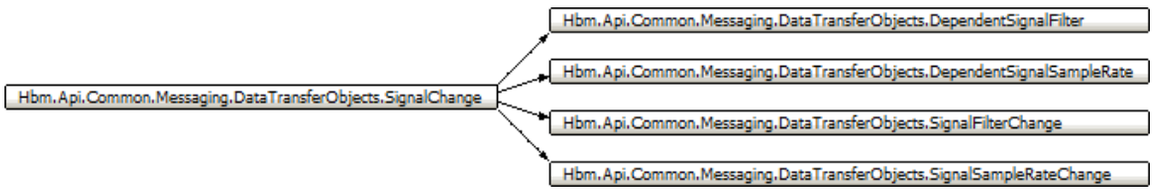
```
<PublicAPI>
Public Property IsBusy() As bool
```

説明

デバイスがビジー状態の場合は`true`。それ以外の場合は`false`。

SignalChange Class

信号変更用のデータ転送オブジェクトクラス。



C++

```
class SignalSampleRateChange : public SignalChange;
```

C#

```
public class SignalSampleRateChange : SignalChange;
```

Visual Basic

```
Public Class SignalSampleRateChange
Inherits SignalChange
```

ファイル

SignalSampleRateChange.cs

SignalChange Members

次の表は、SignalChangeによって公開されるメンバーの一覧です。

Protected Methods

	名前	説明
	SignalChange	SignalChangeクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	UniqueSignalId	固有の信号識別子を取得します。

SignalChange.SignalChange Constructor

SignalChangeクラスの新しいインスタンスを初期化します。

C++

```
SignalChange(
    string signalId
);
```

C#

```
protected SignalChange(
    string signalId
);
```

Visual Basic

```
Public Sub New(
    signalId  As string
)
```

引数

引数	説明
signalId	信号識別子。

SignalChange Properties

SignalChangeクラスのプロパティの一覧です。

Public Properties

	名前	説明
	UniqueSignalId	固有の信号識別子を取得します。

SignalChange.UniqueSignalId Property

固有の信号識別子を取得します。

C++

```
__property string UniqueSignalId;
```

C#

```
public string UniqueSignalId;
```

Visual Basic

```
Public Property UniqueSignalId() As string
```

説明

信号識別子。

SignalFilterChange Class

信号(およびすべての依存信号)のフィルタ変更を表すクラス。



C++

```
class SignalFilterChange : public SignalChange;
```

C#

```
public class SignalFilterChange : SignalChange;
```

Visual Basic

```
Public Class SignalFilterChange
Inherits SignalChange
```

ファイル

SignalFilterChange.cs

SignalFilterChange Members

次の表は、SignalFilterChangeによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	SignalFilterChange	SignalFilterChangeクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	DependentSignalsOldFilterValue	依存信号フィルタの変更を取得します。
	UniqueSignalId	固有の信号識別子を取得します。

SignalFilterChange.SignalFilterChange Constructor

SignalFilterChangeクラスの新しいインスタンスを初期化します。

C++

```
SignalFilterChange(
    string signalId,
    List<DependentSignalFilter> dependentSignalsOldFilterValue
);
```

C#

```
public SignalFilterChange(
    string signalId,
    List<DependentSignalFilter> dependentSignalsOldFilterValue
);
```

Visual Basic

```
Public Sub New(
    signalId As string,
    dependentSignalsOldFilterValue As List<DependentSignalFilter>
)
```

引数

引数	説明
signalId	信号識別子。
dependentSignalsOldFilterValue	変更する依存信号フィルタ。

SignalFilterChange Properties

SignalFilterChangeクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DependentSignalsOldFilterValue	依存信号フィルタの変更を取得します。

SignalFilterChange.DependentSignalsOldFilterValue Property

依存信号フィルタの変更を取得します。

C++

```
__property List<DependentSignalFilter> DependentSignalsOldFilterValue;
```

C#

```
public List<DependentSignalFilter> DependentSignalsOldFilterValue;
```

Visual Basic

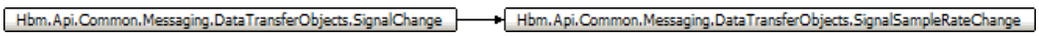
```
Public Property DependentSignalsOldFilterValue() As List<DependentSignalFilter>
```

説明

変更する依存信号フィルタ。

SignalSampleRateChange Class

信号リストのサンプルレートの変更を表すクラス。



C++

```
class SignalSampleRateChange : public SignalChange;
```

C#

```
public class SignalSampleRateChange : SignalChange;
```

Visual Basic

```
Public Class SignalSampleRateChange
Inherits SignalChange
```

ファイル

SignalSampleRateChange.cs

SignalSampleRateChange Members

次の表は、SignalSampleRateChangeによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	SignalSampleRateChange	SignalSampleRateChangeクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	DependentSignalsOldSampleRatesValues	依存信号のサンプルレートを取得します。
	UniqueSignalId	固有の信号識別子を取得します。

SignalSampleRateChange.SignalSampleRateChange Constructor

SignalSampleRateChangeクラスの新しいインスタンスを初期化します。

C++

```
SignalSampleRateChange(
    string signalId,
    List<DependentSignalSampleRate> dependentSignalsOldSampleRatesValues
);
```

C#

```
public SignalSampleRateChange(
    string signalId,
    List<DependentSignalSampleRate> dependentSignalsOldSampleRatesValues
);
```

Visual Basic

```
Public Sub New(
    signalId As string,
    dependentSignalsOldSampleRatesValues As List<DependentSignalSampleRate>
)
```

引数

引数	説明
signalId	信号識別子。
dependentSignalsOldSampleRatesValues	依存信号サンプルレート。

SignalSampleRateChange Properties

SignalSampleRateChangeクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DependentSignalsOldSampleRatesValues	依存信号のサンプルレートを取得します。

SignalSampleRateChange.DependentSignalsOldSampleRatesValues Property

依存信号のサンプルレートを取得します。

C++

```
__property List<DependentSignalSampleRate> DependentSignalsOldSampleRatesValues;
```

C#

```
public List<DependentSignalSampleRate> DependentSignalsOldSampleRatesValues;
```

Visual Basic

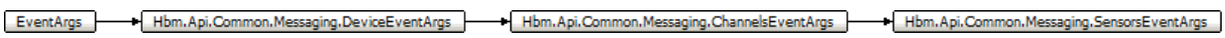
```
Public Property DependentSignalsOldSampleRatesValues() As List<DependentSignalSampleRate>
```

説明

依存信号のサンプルレート。

ChannelsEventArgs Class

MessageBroker.ChannelNameChangedなどのチャネルステータスイベントのデータを提供します。



C++

```
class SensorsEventArgs : public ChannelsEventArgs;
```

C#

```
public class SensorsEventArgs : ChannelsEventArgs;
```

Visual Basic

```
Public Class SensorsEventArgs
Inherits ChannelsEventArgs
```

ファイル

ChannelsEventArgs.cs

ChannelsEventArgs Members

次の表は、ChannelsEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ChannelsEventArgs	ChannelsEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	Reason	変更の原因となった理由。
	UniqueChannelIDs	固有のチャネルID(DeviceEventArgs.UniqueDeviceIDを持つデバイスに対してのみ固有)。
	UniqueDeviceID	固有のデバイスID。

ChannelsEventArgs.ChannelsEventArgs Constructor

ChannelsEventArgsクラスの新しいインスタンスを初期化します。

C++

```
ChannelsEventArgs(
    string uniqueDeviceID,
    List<string> uniqueChannelIDs,
    ChangeReason reason = ChangeReason.Unspecific
);
```

C#

```
public ChannelsEventArgs(
    string uniqueDeviceID,
    List<string> uniqueChannelIDs,
    ChangeReason reason
);
```

Visual Basic

```
Public Sub New(
    uniqueDeviceID As string,
    uniqueChannelIDs As List<string>,
    reason As ChangeReason = ChangeReason.Unspecific
)
```

引数

引数	説明
uniqueDeviceID	チャネルのデバイス固有の識別子。
uniqueChannelIDs	固有のチャネルID。
reason	変更の原因となった理由。

ChannelsEventArgs Properties

SignalSampleRateChangeクラスのプロパティの一覧です。

Public Properties

	名前	説明
	UniqueChannelIDs	固有のチャネルID(DeviceEventArgs.UniqueDeviceIDを持つデバイスに対してのみ固有)。

ChannelsEventArgs.UniqueChannelIDs Property

固有のチャネルID(DeviceEventArgs.UniqueDeviceIDを持つデバイスに対してのみ固有)。

C++

```
__property List<string> UniqueChannelIDs;
```

C#

```
public List<string> UniqueChannelIDs;
```

Visual Basic

```
Public Property UniqueChannelIDs() As List<string>
```

ChannelsStateChangedEventArgs Class

チャネルステータスイベントのデータを提供します。



C++

```
class ChannelsStateChangedEventArgs : public DeviceEventArgs;
```

C#

```
public class ChannelsStateChangedEventArgs : DeviceEventArgs;
```

Visual Basic

```
Public Class ChannelsStateChangedEventArgs
Inherits DeviceEventArgs
```

ファイル

ChannelsStateChangedEventArgs.cs

ChannelsStateChangedEventArgs Members

次の表は、ChannelsStateChangedEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ChannelsStateChangedEventArgs	ChannelsStateChangedEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	ChannelsStateChanges	チャネルのオーバーフローステータス
	Reason	変更の原因となった理由。
	UniqueDeviceID	固有のデバイスID。

ChannelsStateChangedEventArgs.ChannelsStateChangedEventArgs Constructor

ChannelsStateChangedEventArgsクラスの新しいインスタンスを初期化します。

C++

```
ChannelsStateChangedEventArgs(
    string uniqueDeviceId,
    List<ChannelStateChange> channelsStateChanges
);
```

C#

```
public ChannelsStateChangedEventArgs(
    string uniqueDeviceId,
    List<ChannelStateChange> channelsStateChanges,
);
```

Visual Basic

```
Public Sub New(
    uniqueDeviceId As string,
    channelsStateChanges As List<ChannelStateChange>,
)
```

引数

引数	説明
uniqueDeviceID	チャネルのデバイス固有の識別子。
channelsStateChanges	状態を変更するチャネル。

ChannelsStateChangedEventArgs Properties

ChannelsStateChangedEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ChannelsStateChanges	チャネルのオーバーフローステータス

ChannelsStateChangedEventArgs.ChannelsStateChanges Property

チャネルのオーバーフローステータス。

C++

```
__property List<ChannelStateChange> ChannelsStateChanges;
```

C#

```
public List<ChannelStateChange> ChannelsStateChanges;
```

Visual Basic

```
Public Property ChannelsStateChanges() As List<ChannelStateChange>
```

ChannelsTedsStatusEventArgs Class

MessageBroker.ChannelTedsStatusChangedなど、チャネルのTEDSステータスイベントのデータを提供します。



C++

```
class ChannelsTedsStatusEventArgs : public DeviceEventArgs;
```

C#

```
public class ChannelsTedsStatusEventArgs : DeviceEventArgs;
```

Visual Basic

```
Public Class ChannelsTedsStatusEventArgs
Inherits DeviceEventArgs
```

ファイル

ChannelsTedsStatusEventArgs.cs

ChannelsTedsStatusEventArgs Members

次の表は、ChannelsTedsStatusEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ChannelsTedsStatusEventArgs	ChannelsTedsStatusEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	Reason	変更の原因となった理由。
	TedsValidationValueTypes	TEDS検証値の型を取得します。
	UniqueDeviceID	固有のデバイスID。

ChannelsTedsStatusEventArgs.ChannelsTedsStatusEventArgs Constructor

ChannelsTedsStatusEventArgsクラスの新しいインスタンスを初期化します。

C++

```
ChannelsTedsStatusEventArgs(
    string uniqueDeviceID,
    List<ChannelTedsChange> tedsValidationValueTypes,
    ChangeReason reason = ChangeReason.Unspecific
);
```

C#

```
public ChannelsTedsStatusEventArgs(
    string uniqueDeviceID,
    List<ChannelTedsChange> tedsValidationValueTypes,
    ChangeReason reason
);
```

Visual Basic

```
Public Sub New(
    uniqueDeviceID As string,
    tedsValidationValueTypes As List<ChannelTedsChange>,
    reason As ChangeReason = ChangeReason.Unspecific
)
```

引数

引数	説明
uniqueDeviceID	チャネルのデバイス固有の識別子。
tedsValidationValueTypes	TEDS検証値の型。
reason	変更の原因となった理由。

ChannelsTedsStatusEventArgs Properties

ChannelsTedsStatusEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	TedsValidationValueTypes	TEDS検証値の型を取得します。

ChannelsTedsStatusEventArgs.TedsValidationValueTypes Property

TEDS検証値の型を取得します。

C++

```
__property List<ChannelTedsChange> TedsValidationValueTypes;
```

C#

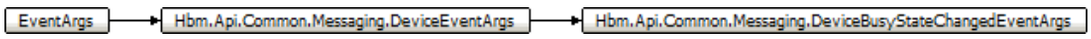
```
public List<ChannelTedsChange> TedsValidationValueTypes;
```

Visual Basic

```
Public Property TedsValidationValueTypes() As List<ChannelTedsChange>
```

DeviceBusyStateChangedEventArgs Class

デバイスのビジーステータスイベントのデータを提供します。



C++

```
class DeviceBusyStateChangedEventArgs : public DeviceEventArgs;
```

C#

```
public class DeviceBusyStateChangedEventArgs : DeviceEventArgs;
```

Visual Basic

```
Public Class DeviceBusyStateChangedEventArgs
Inherits DeviceEventArgs
```

ファイル

DeviceBusyStateChangedEventArgs.cs

DeviceBusyStateChangedEventArgs Members

次の表は、DeviceBusyStateChangedEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DeviceBusyStateChangedEventArgs	DeviceBusyStateChangedEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	DeviceBusyStateChange	変更したデバイスのビジー状態。
	Reason	変更の原因となった理由。
	UniqueDeviceID	固有のデバイスID。

DeviceBusyStateChangedEventArgs.DeviceBusyStateChangedEventArgs Constructor

DeviceBusyStateChangedEventArgsクラスの新しいインスタンスを初期化します。

C++

```
DeviceBusyStateChangedEventArgs(
    string uniqueDeviceID,
    DeviceBusyStateChange deviceBusyStateChange
);
```

C#

```
public DeviceBusyStateChangedEventArgs(
    string uniqueDeviceID,
    DeviceBusyStateChange deviceBusyStateChange
);
```

Visual Basic

```
Public Sub New(
    uniqueDeviceID As string,
    deviceBusyStateChange As DeviceBusyStateChange
)
```

引数

引数	説明
uniqueDeviceID	チャネルのデバイス固有の識別子。
deviceBusyStateChange	状態が変更したチャネル。

DeviceBusyStateChangedEventArgs Properties

DeviceBusyStateChangedEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DeviceBusyStateChange	変更したデバイスのビジー状態。

DeviceBusyStateChangedEventArgs.DeviceBusyStateChange Property

変更したデバイスのビジー状態。

C++

```
[PublicAPI]
__property DeviceBusyStateChange DeviceBusyStateChange;
```

C#

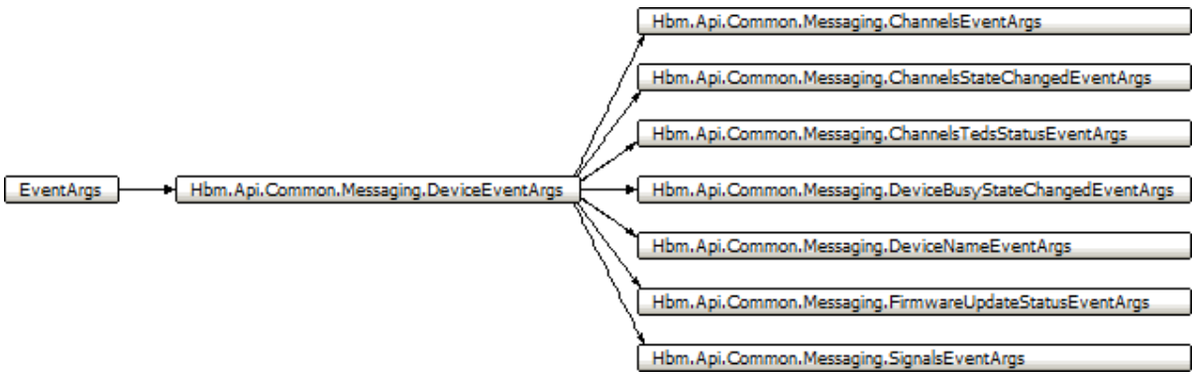
```
[PublicAPI]
public DeviceBusyStateChange DeviceBusyStateChange;
```

Visual Basic

```
<PublicAPI>
Public Property DeviceBusyStateChange() As DeviceBusyStateChange
```

DeviceEventArgs Class

MessageBroker.DeviceConnectedなどのDeviceStatusイベントのデータを提供します。



C++

```
class DeviceEventArgs : public EventArgs;
```

C#

```
public class DeviceEventArgs : EventArgs;
```

Visual Basic

```
Public Class DeviceEventArgs
Inherits EventArgs
```

ファイル

DeviceEventArgs.cs

DeviceEventArgs Members

次の表は、DeviceEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DeviceEventArgs	DeviceEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	Reason	変更の原因となった理由。
	UniqueDeviceID	固有のデバイスID。

DeviceEventArgs.DeviceEventArgs Constructor

DeviceEventArgsクラスの新しいインスタンスを初期化します。

C++

```
DeviceEventArgs(
    AnsiString uniqueDeviceID,
    ChangeReason reason = ChangeReason.Unspecific
);
```

C#

```
public DeviceEventArgs(
    String uniqueDeviceID,
    ChangeReason reason
);
```

Visual Basic

```
Public Sub New(
    uniqueDeviceID As String,
    reason As ChangeReason = ChangeReason.Unspecific
)
```

引数

引数	説明
uniqueDeviceID	チャネルのデバイス固有の識別子。
Reason	変更の原因となった理由。

DeviceEventArgs Properties

DeviceBusyStateChangedEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Reason	変更の原因となった理由。
	UniqueDeviceID	固有のデバイスID。

DeviceEventArgs.Reason Property

変更の原因となった理由。

C++

```
__property ChangeReason Reason;
```

C#

```
public ChangeReason Reason;
```

Visual Basic

```
Public Property Reason() As ChangeReason
```

DeviceEventArgs.UniqueDeviceID Property

固有のデバイスID。

C++

```
__property AnsiString UniqueDeviceID;
```

C#

```
public String UniqueDeviceID;
```

Visual Basic

```
Public Property UniqueDeviceID() As String
```

DeviceNameEventArgs Class

DeviceNameChangedイベントにデータを提供します。



C++

```
class DeviceNameEventArgs : public DeviceEventArgs;
```

C#

```
public class DeviceNameEventArgs : DeviceEventArgs;
```

Visual Basic

```
Public Class DeviceNameEventArgs
Inherits DeviceEventArgs
```

ファイル

DeviceNameEventArgs.cs

DeviceNameEventArgs Members

次の表は、DeviceNameEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	DeviceNameEventArgs	DeviceNameEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	NewName	新しいデバイス名。
	OldName	前のデバイス名。
	Reason	変更の原因となった理由。
	UniqueDeviceID	固有のデバイスID。

DeviceNameEventArgs.DeviceNameEventArgs Constructor

DeviceNameEventArgsクラスの新しいインスタンスを初期化します。

C++

```
DeviceNameEventArgs(
    AnsiString uniqueDeviceID,
    string oldName,
    string newName,
    ChangeReason reason = ChangeReason.Unspecific
);
```

C#

```
public DeviceNameEventArgs(
    String uniqueDeviceID,
    string oldName,
    string newName,
    ChangeReason reason
);
```

Visual Basic

```
Public Sub New(
    uniqueDeviceID As String,
    oldName As string,
    newName As string,
    reason As ChangeReason = ChangeReason.Unspecific
)
```

引数

引数	説明
uniqueDeviceID	デバイス固有の識別子。
oldName	前のデバイス名。
newName	新しいデバイス名。
Reason	変更の原因となった理由。

DeviceNameEventArgs Properties

DeviceBusyStateChangedEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	NewName	新しいデバイス名。
	OldName	前のデバイス名。

DeviceNameEventArgs.NewName Property

新しいデバイス名。

C++

```
__property AnsiString NewName;
```

C#

```
public String NewName;
```

Visual Basic

```
Public Property NewName() As String
```

DeviceNameEventArgs.OldName Property

前のデバイス名。

C++

```
__property AnsiString OldName;
```

C#

```
public String OldName;
```

Visual Basic

```
Public Property OldName() As String
```

FiltersEventArgs Class

MessageBroker.FiltersChangedなどのフィルタイベントのデータを提供します。



C++

```
class FiltersEventArgs : public SignalsEventArgs;
```

C#

```
public class FiltersEventArgs : SignalsEventArgs;
```

Visual Basic

```
Public Class FiltersEventArgs
Inherits SignalsEventArgs
```

ファイル

FiltersEventArgs.cs

FiltersEventArgs Members

次の表は、FiltersEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	FiltersEventArgs	FiltersEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	OldFiltersByUniqueSignalIDs	固有の信号IDによってグループ化された古いフィルタを取得します。
	Reason	変更の原因となった理由。
	UniqueDeviceID	固有のデバイスID。
	UniqueSignalIDs	固有の信号ID(DeviceEventArgs.UniqueDeviceIDを持つデバイスに対してのみ固有)。

FiltersEventArgs.FiltersEventArgs Constructor

FiltersEventArgsクラスの新しいインスタンスを初期化します。

C++

```
DeviceNameEventArgs(
    string uniqueDeviceID,
    List<SignalFilterChange> oldFiltersByUniqueSignalIDs,
    List<string> uniqueSignalIDs,
    ChangeReason  reason,
);
```

C#

```
public FiltersEventArgs(
    string uniqueDeviceID,
    List<SignalFilterChange> oldFiltersByUniqueSignalIDs,
    List<string> uniqueSignalIDs,
    ChangeReason reason
);
```

Visual Basic

```
Public Sub New(
    uniqueDeviceID As string,
    oldFiltersByUniqueSignalIDs As List<SignalFilterChange>,
    uniqueSignalIDs As List<string>,
    reason As ChangeReason
)
```

引数

引数	説明
uniqueDeviceID	信号のデバイス固有の識別子。
oldFiltersByUniqueSignalIDs	固有の信号識別子によってフィルタされた変更前の古いフィルタ。
uniqueSignalIDs	固有の信号識別子。
Reason	変更の原因となった理由。

FiltersEventArgs Properties

FiltersEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	OldFiltersByUniqueSignalIDs	固有の信号IDによってグループ化された古いフィルタを取得します。

FiltersEventArgs.OldFiltersByUniqueSignalIDs Property

固有の信号IDによってグループ化された古いフィルタを取得します。

C++

```
__property List<SignalFilterChange> OldFiltersByUniqueSignalIDs;
```

C#

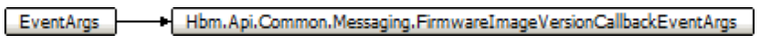
```
public List<SignalFilterChange> OldFiltersByUniqueSignalIDs;
```

Visual Basic

```
Public Property OldFiltersByUniqueSignalIDs() As List<SignalFilterChange>
```

FirmwareImageVersionCallbackEventArgs Class

イメージバージョンのコールバックがトリガされたときに発生するイベント。



C++

```
class FirmwareImageVersionCallbackEventArgs : public EventArgs;
```

C#

```
public class FirmwareImageVersionCallbackEventArgs : EventArgs;
```

Visual Basic

```
Public Class FirmwareImageVersionCallbackEventArgs
Inherits EventArgs
```

ファイル

FirmwareImageVersionCallbackEventArgs.cs

FirmwareImageVersionCallbackEventArgs Members

次の表は、FirmwareImageVersionCallbackEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	FirmwareImageVersionCallbackEventArgs	これは、FirmwareImageVersionCallbackEventArgsクラスのメンバーであるFirmwareImageVersionCallbackEventArgsです。

Public Properties

	名前	説明
	ImageVersion	更新プロセスの進行状況。

FirmwareImageVersionCallbackEventArgs.FirmwareImageVersionCallbackEventArgs Constructor

これは、FirmwareImageVersionCallbackEventArgsクラスのメンバーであるFirmwareImageVersionCallbackEventArgsです。

C++

```
FirmwareImageVersionCallbackEventArgs(
    string imageVersion
);
```

C#

```
public FirmwareImageVersionCallbackEventArgs(
    string imageVersion
);
```

Visual Basic

```
Public Sub New(
    imageVersion As string,
)
```

引数

引数	説明
uniqueDeviceID	信号のデバイス固有の識別子。
oldFiltersByUniqueSignalIDs	固有の信号識別子によってフィルタされた変更前の古いフィルタ。
uniqueSignalIDs	固有の信号識別子。
Reason	変更の原因となった理由。

FirmwareImageVersionCallbackEventArgs Properties

FirmwareImageVersionCallbackEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ImageVersion	更新プロセスの進行状況。

FirmwareImageVersionCallbackEventArgs.ImageVersion Property

更新プロセスの進行状況。

C++

```
__property string ImageVersion;
```

C#

```
public string ImageVersion;
```

Visual Basic

```
Public Property ImageVersion() As string
```

FirmwareUpdateStatusEventArgs Class

デバイスの更新ステータスを通知するイベントクラス。



C++

```
class FirmwareUpdateStatusEventArgs : public DeviceEventArgs;
```

C#

```
public class FirmwareUpdateStatusEventArgs : DeviceEventArgs;
```

Visual Basic

```
Public Class FirmwareUpdateStatusEventArgs
Inherits DeviceEventArgs
```

ファイル

FirmwareUpdateStatusEventArgs.cs

FirmwareUpdateStatusEventArgs Members

次の表は、FirmwareImageVersionCallbackEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	FirmwareUpdateStatusEventArgs	FirmwareUpdateStatusEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	DeviceName	更新したデバイス名。
	Error	更新中に発生したエラーを含む文字列。
	OverallProgress	更新プロセスの進行状況。
	Reason	変更の原因となった理由。
	Rebooting	デバイスが再起動中かどうかを示すブール値。
	TaskName	現在のタスク。
	TaskProgress	更新中に発生したエラーを含む文字列。
	UniqueDeviceID	固有のデバイスID。
	UpdateProcessFinished	更新プロセスが完了したかどうかを示すブール値。

FirmwareUpdateStatusEventArgs.FirmwareUpdateStatusEventArgs Constructor

FirmwareUpdateStatusEventArgsクラスの新しいインスタンスを初期化します。

C++

```
FirmwareUpdateStatusEventArgs(
    string overallProgress,
    string taskName,
    string uniqueDeviceID,
    string deviceName,
    string error,
    string taskProgress,
    bool updateProcessFinished,
    bool rebooting,
    ChangeReason reason = ChangeReason.Unspecific
);
```

C#

```
public FirmwareUpdateStatusEventArgs(
    string overallProgress,
    string taskName,
    string uniqueDeviceID,
    string deviceName,
    string error,
    string taskProgress,
    bool updateProcessFinished,
    bool rebooting,
    ChangeReason reason
);
```

Visual Basic

```
Public Sub New(
    overallProgress As string,
    taskName As string,
    uniqueDeviceID As string,
    deviceName As string,
    error As string,
    taskProgress As string,
    updateProcessFinished As bool,
    rebooting As bool,
    reason As ChangeReason = ChangeReason.Unspecific
)
```

引数

引数	説明
overallProgress	更新プロセスの進捗状況。
taskName	現在実行中のタスク。
uniqueDeviceID	デバイス固有の識別子。
deviceName	デバイス名。
error	エラーメッセージ。
taskProgress	タスクの進捗状況。
updateProcessFinished	更新プロセスが完了したかどうかを示すブール値。
rebooting	デバイスが再起動中かどうかを示すブール値。
reason	変更の原因となった理由。

FirmwareUpdateStatusEventArgs Properties

FirmwareUpdateStatusEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DeviceName	更新したデバイス名。
	Error	更新中に発生したエラーを含む文字列。
	OverallProgress	更新プロセスの進行状況。
	Rebooting	デバイスが再起動中かどうかを示すブール値。
	TaskName	現在のタスク。
	TaskProgress	更新中に発生したエラーを含む文字列。
	UpdateProcessFinished	更新プロセスが完了したかどうかを示すブール値。

FirmwareUpdateStatusEventArgs.DeviceName Property

更新したデバイス名。

C++

```
__property string DeviceName;
```

C#

```
public string DeviceName;
```

Visual Basic

```
Public Property DeviceName() As string
```

FirmwareUpdateStatusEventArgs.Error Property

更新中に発生したエラーを含む文字列。

C++

```
__property string Error;
```

C#

```
public string Error;
```

Visual Basic

```
Public Property Error() As string
```

FirmwareUpdateStatusEventArgs.OverallProgress Property

更新プロセスの進行状況。

C++

```
__property string OverallProgress;
```

C#

```
public string OverallProgress;
```

Visual Basic

```
Public Property OverallProgress() As string
```

FirmwareUpdateStatusEventArgs.Rebooting Property

デバイスが再起動中かどうかを示すブール値。

C++

```
__property bool Rebooting;
```

C#

```
public bool Rebooting;
```

Visual Basic

```
Public Property Rebooting() As bool
```

FirmwareUpdateStatusEventArgs.TaskName Property

現在のタスク。

C++

```
__property string TaskName;
```

C#

```
public string TaskName;
```

Visual Basic

```
Public Property TaskName() As string
```

FirmwareUpdateStatusEventArgs.TaskProgress Property

更新中に発生したエラーを含む文字列。

C++

```
__property string TaskProgress;
```

C#

```
public string TaskProgress;
```

Visual Basic

```
Public Property TaskProgress() As string
```

FirmwareUpdateStatusEventArgs.UpdateProcessFinished Property

更新プロセスが完了したかどうかを示すブール値。

C++

```
__property bool UpdateProcessFinished;
```

C#

```
public bool UpdateProcessFinished;
```

Visual Basic

```
Public Property UpdateProcessFinished() As bool
```

MessageBroker Class

完全なCommonApiのメッセージの中枢。



C++

```
[PublicAPI]
class MessageBroker;
```

C#

```
[PublicAPI]
public class MessageBroker;
```

Visual Basic

```
<PublicAPI>
Public Class MessageBroker
```

ファイル

MessageBroker.cs

備考

潜在的なリスナーは、1つ以上の公開イベントに登録できます。これらのリスナーは、サブスクライブされたイベントが発生するたびに非同期で通知が送信されます。

MessageBroker Members

次の表は、MessageBrokerによって公開されるメンバーの一覧です。

Public Delegates

名前	説明
AutoCalibrationModeChangedEventHandler	関連付けられたイベントのパラメータを定義します。
ChannelNameChangedEventHandler	関連付けられたイベントのパラメータを定義します。
ChannelsStateChangedEventHandler	関連付けられたイベントのパラメータを定義します。
ChannelTedsStatusEventHandler	関連付けられたイベントのパラメータを定義します。
DeviceBusyStateChangedEventHandler	関連付けられたイベントのパラメータを定義します。
DeviceNameEventHandler	関連付けられたイベントのパラメータを定義します。
DeviceStatusEventHandler	関連付けられたイベントのパラメータを定義します。
FiltersEventHandler	FiltersChangedなどのフィルタイベントを処理するメソッドを表します。
FirmwareImageVersionEventHandler	関連付けられたイベントのパラメータを定義します。
FirmwareUpdateStatusEventHandler	関連付けられたイベントのパラメータを定義します。
SampleRatesEventHandler	SampleRatesChangedなど、信号イベントの小数値の変更を処理するメソッドを表します。
SensorsEventHandler	関連付けられたイベントのパラメータを定義します。
SignalStatusEventHandler	関連付けられたイベントのパラメータを定義します。

Public Events

	名前	説明
	AutoCalibrationModeChanged	更新ライブラリによって行われたコールバックが起動されたときに発生します(更新プロセスのステータスを変更します)。
	ChannelNameChanged	チャネル名が変更されたときに発生します。
	ChannelsStateChanged	チャネルのステータスが変化したときに発生します。
	ChannelTedsStatusChanged	チャネルのTEDSステータスが変更されたときに発生します。
	DeviceBusyStateChanged	デバイスのビジーステータスが変化したときに発生します。
	DeviceConnected	デバイスが接続されたときに発生します。
	DeviceDisconnected	デバイスの接続が解除されたときに発生します。
	DeviceNameChanged	デバイス名が変更されたときに発生します。
	FiltersChanged	少なくとも1つのフィルタ設定が変更された場合に発生します。
	FirmwareImageVersionLoaded	アップデートライブラリによって行われたコールバックが起動されたとき(ファームウェアファイルのイメージ バージョンがロードされたとき)に発生します。
	FirmwareUpdateStatusChanged	更新ライブラリによって行われたコールバックが起動されたときに発生します(更新プロセスのステータスを変更します)。
	SampleRatesChanged	信号の少なくとも1つのサンプル レートが変更されたときに発生します(例: 依存信号のサンプルレートの変更をアナウンスするため)。
	SensorsChanged	少なくとも1つのセンサ設定が変更された場合に発生します。
	SignalNameChanged	信号名が変更されたときに発生します。

Protected Methods

	名前	説明
S	RaiseAutoCalibrationModeChangedAsync	AutoCalibrationModeChangedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseChannelNameChangedAsync	ChannelNameChangedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseChannelsStatesChangedAsync	ChannelsStateChangedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseChannelTedsStatusChangedAsync	ChannelTedsStatusChangedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseDeviceBusyStateChangedAsync	DeviceBusyStateChangedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseDeviceConnectedAsync	DeviceConnectedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseDeviceDisconnectedAsync	DeviceDisconnectedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseDeviceFirmwareImageVersionLoadedChangedAsync	FirmwareImageVersionLoadedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseDeviceFirmwareUpdateStatusChangedAsync	FirmwareUpdateStatusChangedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseDeviceNameChangedAsync	DeviceNameChangedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseFiltersChangedAsync	FiltersChangedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseSampleRatesChangedAsync	SampleRatesChangedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseSensorsChangedAsync	SensorsChangedイベントを非同期的に(Fire-and-Forget)発生させます。
S	RaiseSignalNameChangedAsync	SignalNameChangedイベントを非同期的に(Fire-and-Forget)発生させます。

MessageBroker Methods

MessageBrokerクラスのメソッドの一覧です。

Protected Methods

	名前	説明
	RaiseAutoCalibrationModeChangedAsync	AutoCalibrationModeChangedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseChannelNameChangedAsync	ChannelNameChangedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseChannelsStatesChangedAsync	ChannelsStateChangedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseChannelTedsStatusChangedAsync	ChannelTedsStatusChangedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseDeviceBusyStateChangedAsync	DeviceBusyStateChangedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseDeviceConnectedAsync	DeviceConnectedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseDeviceDisconnectedAsync	DeviceDisconnectedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseDeviceFirmwareImageVersionLoadedChangedAsync	FirmwareImageVersionLoadedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseDeviceFirmwareUpdateStatusChangedAsync	FirmwareUpdateStatusChangedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseDeviceNameChangedAsync	DeviceNameChangedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseFiltersChangedAsync	FiltersChangedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseSampleRatesChangedAsync	SampleRatesChangedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseSensorsChangedAsync	SensorsChangedイベントを非同期的に(Fire-and-Forget)発生させます。
	RaiseSignalNameChangedAsync	SignalNameChangedイベントを非同期的に(Fire-and-Forget)発生させます。

MessageBroker.RaiseAutoCalibrationModeChangedAsync Method

AutoCalibrationModeChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseAutoCalibrationModeChangedAsync(
    object sender,
    ChannelsEventArgs e
);
```

C#

```
public static void RaiseAutoCalibrationModeChangedAsync(
    object sender,
    ChannelsEventArgs e
);
```

Visual Basic

```
Public static Function RaiseAutoCalibrationModeChangedAsync(
    sender As object,
    e As ChannelsEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントAutoCalibrationModeChangedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseChannelNameChangedAsync Method

ChannelNameChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseChannelNameChangedAsync(
    object sender,
    ChannelsEventArgs e
);
```

C#

```
public static void RaiseChannelNameChangedAsync(
    object sender,
    ChannelsEventArgs e
);
```

Visual Basic

```
Public static Function RaiseChannelNameChangedAsync(
    sender As object,
    e As ChannelsEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

MessageBroker.RaiseChannelsStatesChangedAsync Method

ChannelsStateChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseChannelsStatesChangedAsync(
    object sender,
    ChannelsStateChangedEventArgs e
);
```

C#

```
public static void RaiseChannelsStatesChangedAsync(
    object sender,
    ChannelsStateChangedEventArgs e
);
```

Visual Basic

```
Public static Function RaiseChannelsStatesChangedAsync(
    sender As object,
    e As ChannelsStateChangedEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントChannelsStateChangedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseChannelTedsStatusChangedAsync Method

ChannelTedsStatusChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseChannelTedsStatusChangedAsync(
    object sender,
    ChannelsTedsStatusEventArgs e
);
```

C#

```
public static void RaiseChannelTedsStatusChangedAsync(
    object sender,
    ChannelsTedsStatusEventArgs e
);
```

Visual Basic

```
Public static Function RaiseChannelTedsStatusChangedAsync(
    sender As object,
    e As ChannelsTedsStatusEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントChannelTedsStatusChangedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseDeviceBusyStateChangedAsync Method

DeviceBusyStateChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseDeviceBusyStateChangedAsync(
    object sender,
    DeviceBusyStateChangedEventArgs e
);
```

C#

```
public static void RaiseDeviceBusyStateChangedAsync(
    object sender,
    DeviceBusyStateChangedEventArgs e
);
```

Visual Basic

```
Public static Function RaiseDeviceBusyStateChangedAsync(
    sender As object,
    e As DeviceBusyStateChangedEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントDeviceBusyStateChangedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseDeviceConnectedAsync Method

DeviceConnectedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseDeviceConnectedAsync(
    object sender,
    DeviceEventArgs e
);
```

C#

```
public static void RaiseDeviceConnectedAsync(
    object sender,
    DeviceEventArgs e
);
```

Visual Basic

```
Public static Function RaiseDeviceConnectedAsync(
    sender As object,
    e As DeviceEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントDeviceConnectedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseDeviceDisconnectedAsync Method

DeviceDisconnectedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseDeviceDisconnectedAsync(
    object sender,
    DeviceEventArgs e
);
```

C#

```
public static void RaiseDeviceDisconnectedAsync(
    object sender,
    DeviceEventArgs e
);
```

Visual Basic

```
Public static Function RaiseDeviceDisconnectedAsync(
    sender As object,
    e As DeviceEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントDeviceDisconnectedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseDeviceFirmwareImageVersionLoadedChangedAsync Method

FirmwareImageVersionLoadedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseDeviceFirmwareImageVersionLoadedChangedAsync(
    object sender,
    FirmwareImageVersionCallbackEventArgs e
);
```

C#

```
public static void RaiseDeviceFirmwareImageVersionLoadedChangedAsync(
    object sender,
    FirmwareImageVersionCallbackEventArgs e
);
```

Visual Basic

```
Public static Function RaiseDeviceFirmwareImageVersionLoadedChangedAsync(
    sender As object,
    e As FirmwareImageVersionCallbackEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントFirmwareImageVersionLoadedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseDeviceFirmwareUpdateStatusChangedAsync Method

FirmwareUpdateStatusChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseDeviceFirmwareUpdateStatusChangedAsync(
    object sender,
    FirmwareUpdateStatusEventArgs e
);
```

C#

```
public static void RaiseDeviceFirmwareUpdateStatusChangedAsync(
    object sender,
    FirmwareUpdateStatusEventArgs e
);
```

Visual Basic

```
Public static Function RaiseDeviceFirmwareUpdateStatusChangedAsync(
    sender As object,
    e As FirmwareUpdateStatusEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントFirmwareUpdateStatusChangedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseDeviceNameChangedAsync Method

DeviceNameChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseDeviceNameChangedAsync(
    object sender,
    DeviceNameEventArgs e
);
```

C#

```
public static void RaiseDeviceNameChangedAsync(
    object sender,
    DeviceNameEventArgs e
);
```

Visual Basic

```
Public static Function RaiseDeviceNameChangedAsync(
    sender As object,
    e As DeviceNameEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントDeviceNameChangedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseFiltersChangedAsync Method

FiltersChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseFiltersChangedAsync(
    object sender,
    FiltersEventArgs e
);
```

C#

```
public static void RaiseFiltersChangedAsync(
    object sender,
    FiltersEventArgs e
);
```

Visual Basic

```
Public static Function RaiseFiltersChangedAsync(
    sender As object,
    e As FiltersEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントFiltersChangedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseSampleRatesChangedAsync Method

SampleRatesChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseSampleRatesChangedAsync(
    object sender,
    SampleRatesEventArgs e
);
```

C#

```
public static void RaiseSampleRatesChangedAsync(
    object sender,
    SampleRatesEventArgs e
);
```

Visual Basic

```
Public static Function RaiseSampleRatesChangedAsync(
    sender As object,
    e As SampleRatesEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントSampleRatesChangedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseSensorsChangedAsync Method

SensorsChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseSensorsChangedAsync(
    object sender,
    SensorsEventArgs e
);
```

C#

```
public static void RaiseSensorsChangedAsync(
    object sender,
    SensorsEventArgs e
);
```

Visual Basic

```
Public static Function RaiseSensorsChangedAsync(
    sender As object,
    e As SensorsEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントSensorsChangedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker.RaiseSignalNameChangedAsync Method

SignalNameChangedイベントを非同期的に(Fire-and-Forget)発生させます。

C++

```
void RaiseSignalNameChangedAsync(
    object sender,
    SignalsEventArgs e
);
```

C#

```
public static void RaiseSignalNameChangedAsync(
    object sender,
    SignalsEventArgs e
);
```

Visual Basic

```
Public static Function RaiseSignalNameChangedAsync(
    sender As object,
    e As SignalsEventArgs
) As void
```

引数

引数	説明
sender	イベントのイニシエータ。
e	追加のイベント情報。

備考

登録されているすべてのリスナーにイベントSignalNameChangedを別のスレッドで通知します。これにより、プログラムの流れが滞ることはありません。

MessageBroker Delegates

MessageBrokerクラスのデリゲートの一覧です。

Public Delegates

名前	説明
AutoCalibrationModeChangedEventHandler	関連付けられたイベントのパラメータを定義します。
ChannelNameChangedEventHandler	関連付けられたイベントのパラメータを定義します。
ChannelsStateChangedEventHandler	関連付けられたイベントのパラメータを定義します。
ChannelTedsStatusEventHandler	関連付けられたイベントのパラメータを定義します。
DeviceBusyStateChangedEventHandler	関連付けられたイベントのパラメータを定義します。
DeviceNameEventHandler	関連付けられたイベントのパラメータを定義します。
DeviceStatusEventHandler	関連付けられたイベントのパラメータを定義します。
FiltersEventHandler	FiltersChangedなどのフィルタイベントを処理するメソッドを表します。
FirmwareImageVersionEventHandler	関連付けられたイベントのパラメータを定義します。
FirmwareUpdateStatusEventHandler	関連付けられたイベントのパラメータを定義します。
SampleRatesEventHandler	SampleRatesChangedなど、信号イベントの小数値の変更を処理するメソッドを表します。
SensorsEventHandler	関連付けられたイベントのパラメータを定義します。
SignalStatusEventHandler	関連付けられたイベントのパラメータを定義します。

MessageBroker.AutoCalibrationModeChangedEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void AutoCalibrationModeChangedEventHandler();
```

C#

```
public delegate void AutoCalibrationModeChangedEventHandler( object sender,
ChannelsEventArgs e );
```

Visual Basic

```
Public Delegate Sub AutoCalibrationModeChangedEventHandler( object sender,
ChannelsEventArgs e ) As void
```

MessageBroker.ChannelNameChangedEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void ChannelNameChangedEventHandler();
```

C#

```
public delegate void ChannelNameChangedEventHandler( object sender, ChannelsEventArgs e );
```

Visual Basic

```
Public Delegate Sub ChannelNameChangedEventHandler( object sender, ChannelsEventArgs e ) As
void
```

MessageBroker.ChannelsStateChangedEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void ChannelsStateChangedEventHandler();
```

C#

```
public delegate void ChannelsStateChangedEventHandler( object sender,
ChannelsStateChangedEventArgs e );
```

Visual Basic

```
Public Delegate Sub ChannelsStateChangedEventHandler( object sender,
ChannelsStateChangedEventArgs e ) As void
```

MessageBroker.ChannelTedsStatusEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void ChannelTedsStatusEventHandler();
```

C#

```
public delegate void ChannelTedsStatusEventHandler( object sender,
ChannelsTedsStatusEventArgs e );
```

Visual Basic

```
Public Delegate Sub ChannelTedsStatusEventHandler( object sender,
ChannelsTedsStatusEventArgs e ) As void
```

MessageBroker.DeviceBusyStateChangedEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void DeviceBusyStateChangedEventHandler();
```

C#

```
public delegate void DeviceBusyStateChangedEventHandler( object sender,
DeviceBusyStateChangedEventArgs e );
```

Visual Basic

```
Public Delegate Sub DeviceBusyStateChangedEventHandler( object sender,
DeviceBusyStateChangedEventArgs e ) As void
```

MessageBroker.DeviceNameEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void DeviceNameEventHandler();
```

C#

```
public delegate void DeviceNameEventHandler( object sender, DeviceNameEventArgs e );
```

Visual Basic

```
Public Delegate Sub DeviceNameEventHandler( object sender, DeviceNameEventArgs e ) As void
```

MessageBroker.DeviceStatusEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void DeviceStatusEventHandler();
```

C#

```
public delegate void DeviceStatusEventHandler( object sender, DeviceEventArgs e );
```

Visual Basic

```
Public Delegate Sub DeviceStatusEventHandler( object sender, DeviceEventArgs e ) As void
```

MessageBroker.FiltersEventHandler Delegate

FiltersChangedなどのフィルタイベントを処理するメソッドを表します。

C++

```
delegate void FiltersEventHandler();
```

C#

```
public delegate void FiltersEventHandler( object sender, FiltersEventArgs e );
```

Visual Basic

```
Public Delegate Sub FiltersEventHandler( object sender, FiltersEventArgs e ) As void
```

MessageBroker.FirmwareImageVersionEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void FirmwareImageVersionEventHandler();
```

C#

```
public delegate void FirmwareImageVersionEventHandler( object sender, FirmwareImageVersionCallbackEventArgs e );
```

Visual Basic

```
Public Delegate Sub FirmwareImageVersionEventHandler( object sender, FirmwareImageVersionCallbackEventArgs e ) As void
```

MessageBroker.FirmwareUpdateStatusEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void FirmwareUpdateStatusEventHandler();
```

C#

```
public delegate void FirmwareUpdateStatusEventHandler( object sender,
FirmwareUpdateStatusEventArgs e );
```

Visual Basic

```
Public Delegate Sub FirmwareUpdateStatusEventHandler( object sender,
FirmwareUpdateStatusEventArgs e ) As void
```

MessageBroker.SampleRatesEventHandler Delegate

SampleRatesChangedなど、信号イベントの小数値の変更を処理するメソッドを表します。

C++

```
delegate void SampleRatesEventHandler();
```

C#

```
public delegate void SampleRatesEventHandler( object sender, SampleRatesEventArgs e );
```

Visual Basic

```
Public Delegate Sub SampleRatesEventHandler( object sender, SampleRatesEventArgs e ) As
void
```

MessageBroker.SensorsEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void SensorsEventHandler();
```

C#

```
public delegate void SensorsEventHandler( object sender, SensorsEventArgs e );
```

Visual Basic

```
Public Delegate Sub SensorsEventHandler( object sender, SensorsEventArgs e ) As void
```

MessageBroker.SignalStatusEventHandler Delegate

関連付けられたイベントのパラメータを定義します。

C++

```
delegate void SignalStatusEventHandler();
```

C#

```
public delegate void SignalStatusEventHandler( object sender, SignalsEventArgs e );
```

Visual Basic

```
Public Delegate Sub SignalStatusEventHandler( object sender, SignalsEventArgs e ) As void
```

MessageBroker Events

MessageBrokerクラスのイベントの一覧です。

Public Events

	名前	説明
	AutoCalibrationModeChanged	更新ライブラリによって行われたコールバックが起動されたときに発生します(更新プロセスのステータスを変更します)。
	ChannelNameChanged	チャネル名が変更されたときに発生します。
	ChannelsStateChanged	チャネルのステータスが変化したときに発生します。
	ChannelTedsStatusChanged	チャネルのTEDSステータスが変更されたときに発生します。
	DeviceBusyStateChanged	デバイスのビジーステータスが変化したときに発生します。
	DeviceConnected	デバイスが接続されたときに発生します。
	DeviceDisconnected	デバイスの接続が解除されたときに発生します。
	DeviceNameChanged	デバイス名が変更されたときに発生します。
	FiltersChanged	少なくとも1つのフィルタ設定が変更された場合に発生します。
	FirmwareImageVersionLoaded	アップデートライブラリによって行われたコールバックが起動されたとき(ファームウェアファイルのイメージ バージョンがロードされたとき)に発生します。
	FirmwareUpdateStatusChanged	更新ライブラリによって行われたコールバックが起動されたときに発生します(更新プロセスのステータスを変更します)。
	SampleRatesChanged	信号の少なくとも1つのサンプル レートが変更されたときに発生します(例: 依存信号のサンプルレートの変更をアナウンスするため)。
	SensorsChanged	少なくとも1つのセンサ設定が変更された場合に発生します。
	SignalNameChanged	信号名が変更されたときに発生します。

MessageBroker.AutoCalibrationModeChanged Event

更新ライブラリによって行われたコールバックが起動されたときに発生します(更新プロセスのステータスを変更します)。

C++

```
__property AutoCalibrationModeChangedEventHandler AutoCalibrationModeChanged;
```

C#

```
public static event AutoCalibrationModeChangedEventHandler AutoCalibrationModeChanged;
```

Visual Basic

```
Public static Event AutoCalibrationModeChanged As AutoCalibrationModeChangedEventHandler
```

MessageBroker.ChannelNameChanged Event

チャネル名が変更されたときに発生します。

C++

```
__property ChannelNameChangedEventHandler ChannelNameChanged;
```

C#

```
public static event ChannelNameChangedEventHandler ChannelNameChanged;
```

Visual Basic

```
Public static Event ChannelNameChanged As ChannelNameChangedEventHandler
```

MessageBroker.ChannelsStateChanged Event

チャネルのステータスが変化したときに発生します。

C++

```
__property ChannelsStateChangedEventHandler ChannelsStateChanged;
```

C#

```
public static event ChannelsStateChangedEventHandler ChannelsStateChanged;
```

Visual Basic

```
Public static Event ChannelsStateChanged As ChannelsStateChangedEventHandler
```

MessageBroker.ChannelTedsStatusChanged Event

チャネルのTEDSステータスが変更されたときに発生します。

C++

```
__property ChannelTedsStatusEventHandler ChannelTedsStatusChanged;
```

C#

```
public static event ChannelTedsStatusEventHandler ChannelTedsStatusChanged;
```

Visual Basic

```
Public static Event ChannelTedsStatusChanged As ChannelTedsStatusEventHandler
```

MessageBroker.DeviceBusyStateChanged Event

デバイスのビジーステータスが変化したときに発生します。

C++

```
__property DeviceBusyStateChangedEventHandler DeviceBusyStateChanged;
```

C#

```
public static event DeviceBusyStateChangedEventHandler DeviceBusyStateChanged;
```

Visual Basic

```
Public static Event DeviceBusyStateChanged As DeviceBusyStateChangedEventHandler
```

MessageBroker.DeviceConnected Event

デバイスが接続されたときに発生します。

C++

```
__property DeviceStatusEventHandler DeviceConnected;
```

C#

```
public static event DeviceStatusEventHandler DeviceConnected;
```

Visual Basic

```
Public static Event DeviceConnected As DeviceStatusEventHandler
```

MessageBroker.DeviceDisconnected Event

デバイスの接続が解除されたときに発生します。

C++

```
__property DeviceStatusEventHandler DeviceDisconnected;
```

C#

```
public static event DeviceStatusEventHandler DeviceDisconnected;
```

Visual Basic

```
Public static Event DeviceDisconnected As DeviceStatusEventHandler
```

MessageBroker.DeviceNameChanged Event

デバイス名が変更されたときに発生します。

C++

```
__property DeviceNameEventHandler DeviceNameChanged;
```

C#

```
public static event DeviceNameEventHandler DeviceNameChanged;
```

Visual Basic

```
Public static Event DeviceNameChanged As DeviceNameEventHandler
```

MessageBroker.FiltersChanged Event

少なくとも1つのフィルタ設定が変更された場合に発生します。

C++

```
__property FiltersEventHandler FiltersChanged;
```

C#

```
public static event FiltersEventHandler FiltersChanged;
```

Visual Basic

```
Public static Event FiltersChanged As FiltersEventHandler
```

MessageBroker.FirmwareImageVersionLoaded Event

アップデートライブラリによって行われたコールバックが起動されたとき(ファームウェアファイルのイメージ バージョンがロードされたとき)に発生します。

C++

```
__property FirmwareImageVersionEventHandler FirmwareImageVersionLoaded;
```

C#

```
public static event FirmwareImageVersionEventHandler FirmwareImageVersionLoaded;
```

Visual Basic

```
Public static Event FirmwareImageVersionLoaded As FirmwareImageVersionEventHandler
```

MessageBroker.FirmwareUpdateStatusChanged Event

更新ライブラリによって行われたコールバックが起動されたときに発生します(更新プロセスのステータスを変更します)。

C++

```
__property FirmwareUpdateStatusEventHandler FirmwareUpdateStatusChanged;
```

C#

```
public static event FirmwareUpdateStatusEventHandler FirmwareUpdateStatusChanged;
```

Visual Basic

```
Public static Event FirmwareUpdateStatusChanged As FirmwareUpdateStatusEventHandler
```

MessageBroker.SampleRatesChanged Event

信号の少なくとも1つのサンプル レートが変更されたときに発生します(例: 依存信号のサンプルレートの変更をアナウンスするため)。

C++

```
__property SampleRatesEventHandler SampleRatesChanged;
```

C#

```
public static event SampleRatesEventHandler SampleRatesChanged;
```

Visual Basic

```
Public static Event SampleRatesChanged As SampleRatesEventHandler
```

MessageBroker.SensorsChanged Event

少なくとも1つのセンサ設定が変更された場合に発生します。

C++

```
__property SensorsEventHandler SensorsChanged;
```

C#

```
public static event SensorsEventHandler SensorsChanged;
```

Visual Basic

```
Public static Event SensorsChanged As SensorsEventHandler
```

MessageBroker.SignalNameChanged Event

信号名が変更されたときに発生します。

C++

```
__property SignalStatusEventHandler SignalNameChanged;
```

C#

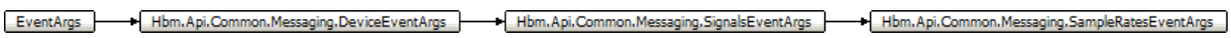
```
public static event SignalStatusEventHandler SignalNameChanged;
```

Visual Basic

```
Public static Event SignalNameChanged As SignalStatusEventHandler
```

SampleRatesEventArgs Class

MessageBroker.SampleRatesChangedなどのサンプルレート変更イベントのデータを提供します。



C++

```
class SampleRatesEventArgs : public SignalsEventArgs;
```

C#

```
public class SampleRatesEventArgs : SignalsEventArgs;
```

Visual Basic

```
Public Class SampleRatesEventArgs
Inherits SignalsEventArgs
```

ファイル

SampleRatesEventArgs.cs

SampleRatesEventArgs Members

次の表は、SampleRatesEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	SampleRatesEventArgs	SampleRatesEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	OldValueByUniqueSignalIDs	固有の信号IDによってグループ化された古い10進値を取得します。
	Reason	変更の原因となった理由。
	UniqueDeviceID	固有のデバイスID。
	UniqueSignalIDs	固有の信号ID(DeviceEventArgs.UniqueDeviceIDを持つデバイスに対してのみ固有)。

SampleRatesEventArgs.SampleRatesEventArgs Constructor

SampleRatesEventArgsクラスの新しいインスタンスを初期化します。

C++

```
SampleRatesEventArgs(
    string uniqueDeviceID,
    List<SignalSampleRateChange> dependentSignalsOldSampleRates,
    List<string> uniqueSignalIDs,
    ChangeReason reason
);
```

C#

```
public SampleRatesEventArgs(
    string uniqueDeviceID,
    List<SignalSampleRateChange> dependentSignalsOldSampleRates,
    List<string> uniqueSignalIDs,
    ChangeReason reason
);
```

Visual Basic

```
Public Sub New(
    uniqueDeviceID As string,
    dependentSignalsOldSampleRates As List<SignalSampleRateChange>,
    uniqueSignalIDs As List<string>,
    reason As ChangeReason
)
```

引数

引数	説明
uniqueDeviceID	シグナルのデバイスの固有の識別子。
dependentSignalsOldSampleRates	依存信号は信号IDごとに古いサンプル レートを示します。
uniqueSignalIDs	固有の信号ID。
reason	Reasonにより変更された10進信号のプロパティ。

SampleRatesEventArgs Properties

SampleRatesEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	OldValueByUniqueSignalIDs	固有の信号IDによってグループ化された古い10進値を取得します。

SampleRatesEventArgs.OldValueByUniqueSignalIDs Property

固有の信号IDによってグループ化された古い10進値を取得します。

C++

```
__property List<SignalSampleRateChange> OldValueByUniqueSignalIDs;
```

C#

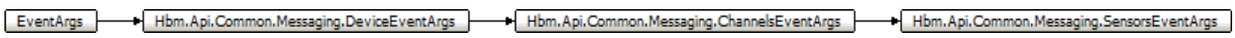
```
public List<SignalSampleRateChange> OldValueByUniqueSignalIDs;
```

Visual Basic

```
Public Property OldValueByUniqueSignalIDs() As List<SignalSampleRateChange>
```

SensorsEventArgs Class

MessageBroker.SensorsChangedなどのセンサイベントのデータを提供します。



C++

```
class SensorsEventArgs : public ChannelsEventArgs;
```

C#

```
public class SensorsEventArgs : ChannelsEventArgs;
```

Visual Basic

```
Public Class SensorsEventArgs
Inherits ChannelsEventArgs
```

ファイル

SensorsEventArgs.cs

SensorsEventArgs Members

次の表は、SampleRatesEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	SensorsEventArgs	SensorsEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	OldSensorsByUniqueChannelIDs	固有のチャネルIDによってグループ化された古いセンサを取得します。
	Reason	変更の原因となった理由。
	UniqueChannelIDs	固有のチャネルID(DeviceEventArgs.UniqueDeviceIDを持つデバイスに対してのみ固有)。
	UniqueDeviceID	固有のデバイスID。

SensorsEventArgs.SensorsEventArgs Constructor

SensorsEventArgsクラスの新しいインスタンスを初期化します。

C++

```
SensorsEventArgs(
    string uniqueDeviceId,
    List<ChannelSensorValueChange> oldSensorsByUniqueChannelIDs,
    List<string> uniqueChannelIDs,
    ChangeReason reason
);
```

C#

```
public SensorsEventArgs(
    string uniqueDeviceId,
    List<ChannelSensorValueChange> oldSensorsByUniqueChannelIDs,
    List<string> uniqueChannelIDs,
    ChangeReason reason
);
```

Visual Basic

```
Public Sub New(
    uniqueDeviceID As string,
    oldSensorsByUniqueChannelIDs As List<ChannelSensorValueChange>,
    uniqueChannelIDs As List<string>,
    reason As ChangeReason
)
```

引数

引数	説明
uniqueDeviceID	シグナルのデバイスの固有の識別子。
oldSensorsByUniqueChannelIDs	固有のチャネルIDによる古いセンサ(変更前)。
uniqueChannelIDs	固有のチャネルID。
reason	変更の原因となった理由。

SensorsEventArgs Properties

SensorsEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	OldSensorsByUniqueChannelIDs	固有のチャネルIDによってグループ化された古いセンサを取得します。

SensorsEventArgs.OldSensorsByUniqueChannelIDs Property

固有のチャネルIDによってグループ化された古いセンサを取得します。

C++

```
__property List<ChannelSensorValueChange> OldSensorsByUniqueChannelIDs;
```

C#

```
public List<ChannelSensorValueChange> OldSensorsByUniqueChannelIDs;
```

Visual Basic

```
Public Property OldSensorsByUniqueChannelIDs() As List<ChannelSensorValueChange>
```

SignalsEventArgs Class

MessageBroker.SignalNameChangedなどのシグナル ステータス イベントのデータを提供します。



C++

```
class SignalsEventArgs : public DeviceEventArgs;
```

C#

```
public class SignalsEventArgs : DeviceEventArgs;
```

Visual Basic

```
Public Class SignalsEventArgs
Inherits DeviceEventArgs
```

ファイル

SignalsEventArgs.cs

SignalsEventArgs Members

次の表は、SignalsEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	SignalsEventArgs	SignalsEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	Reason	変更の原因となった理由。
	UniqueDeviceID	固有のデバイスID。
	UniqueSignalIDs	固有の信号ID(DeviceEventArgs.UniqueDeviceIDを持つデバイスに対してのみ固有)。

SignalsEventArgs.SignalsEventArgs Constructor

SignalsEventArgsクラスの新しいインスタンスを初期化します。

C++

```
SignalsEventArgs(
    string uniqueDeviceID,
    List<string> uniqueSignalIDs,
    ChangeReason reason = ChangeReason.Unspecific
);
```

C#

```
public SignalsEventArgs(
    string uniqueDeviceID,
    List<string> uniqueSignalIDs,
    ChangeReason reason
);
```

Visual Basic

```
Public Sub New(
    uniqueDeviceID As string,
    uniqueSignalIDs As List<string>,
    reason As ChangeReason = ChangeReason.Unspecific
)
```

引数

引数	説明
uniqueDeviceID	シグナルのデバイスの固有の識別子。
uniqueSignalIDs	固有の信号ID。
reason	変更の原因となった理由。

SignalsEventArgs Properties

SignalsEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	UniqueSignalIDs	固有の信号ID(DeviceEventArgs.UniqueDeviceIDを持つデバイスに対してのみ固有)。

SignalsEventArgs.UniqueSignalIDs Property

固有の信号ID(DeviceEventArgs.UniqueDeviceIDを持つデバイスに対してのみ固有)。

C++

```
__property List<string> UniqueSignalIDs;
```

C#

```
public List<string> UniqueSignalIDs;
```

Visual Basic

```
Public Property UniqueSignalIDs() As List<string>
```

Hbm.Api.SensorDB Namespace

目次

Hbm.Api.SensorDB Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ **1399**
Hbm.Api.SensorDB Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1414
ScalingFactory Class ・・ 1415
ScalingFactory Members ・・ 1416
ScalingFactory Methods ・・ 1417
ScalingFactory.CreateScaling Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1418
SensorDBManager Class ・・・ 1419
SensorDBManager Members ・・・ 1420
SensorDBManager Constructor ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1424
SensorDBManager.SensorDBManager Constructor () ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1425
SensorDBManager.SensorDBManager Constructor (string) ・・・・・・・・・・・・・・・・・・・・・・・・ 1426
SensorDBManager.SensorDBManager Constructor (string, string, string) ・・・・・・・・・・・・・・・・ 1427
SensorDBManager Methods ・・・ 1428
SensorDBManager.BuildOrderedDeleteCategoryIdListInForeignKeyOrder Method ・・・・・・・・・・・・・・ 1430
SensorDBManager.CheckForInvalidChildCategoryDeletion Method ・・・・・・・・・・・・・・・・・・・・・ 1431
SensorDBManager.DeleteOrphanTranslations Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1432
SensorDBManager.GenerateXmlExtraTypes Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1433
SensorDBManager.GenerateXmlKnownScalingTypes Method ・・・・・・・・・・・・・・・・・・・・・・・・ 1434
SensorDBManager.GenerateXmlKnownSensorTypes Method ・・・・・・・・・・・・・・・・・・・・・・・・ 1435
SensorDBManager.GetSensorsForCategory Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1436
SensorDBManager.ISensorDB.CategoryExist Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1437
SensorDBManager.ISensorDB.ClearConnectionPool Method ・・・・・・・・・・・・・・・・・・・・・・・・ 1438
SensorDBManager.ISensorDB.DeleteCategory Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1439
SensorDBManager.ISensorDB.DeleteDescriptor Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 1440
SensorDBManager.ISensorDB.DeleteLanguage Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1441
SensorDBManager.ISensorDB.DeleteSensor Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1442
SensorDBManager.ISensorDB.DeleteSensors Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1443
SensorDBManager.ISensorDB.DeleteThermoCoupleType Method ・・・・・・・・・・・・・・・・・・・・・・ 1444
SensorDBManager.ISensorDB.GetActiveLanguage Method ・・・・・・・・・・・・・・・・・・・・・・・・・ 1445
SensorDBManager.ISensorDB.GetAttachment Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1446
SensorDBManager.ISensorDB.GetCategories Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1447
SensorDBManager.ISensorDB.GetCategory Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1448
SensorDBManager.ISensorDB.GetCategoryByInternalID Method ・・・・・・・・・・・・・・・・・・・・・・ 1449
SensorDBManager.ISensorDB.GetDescriptor Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1450
ISensorDB.GetDescriptors Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1451
SensorDBManager.ISensorDB.GetDescriptors Method () ・・・・・・・・・・・・・・・・・・・・・・・・・・ 1452
SensorDBManager.ISensorDB.GetDescriptors Method (Sensor) ・・・・・・・・・・・・・・・・・・・・・・ 1453
SensorDBManager.ISensorDB.GetFallbackLanguage Method ・・・・・・・・・・・・・・・・・・・・・・・・ 1454
SensorDBManager.ISensorDB.GetHierarchicalCategories Method ・・・・・・・・・・・・・・・・・・・・・ 1455
SensorDBManager.ISensorDB.GetLanguage Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1456
SensorDBManager.ISensorDB.GetLanguages Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1457
ISensorDB.GetSensor Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1458
SensorDBManager.ISensorDB.GetSensor Method (int) ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1459
SensorDBManager.ISensorDB.GetSensor Method (string) ・・・・・・・・・・・・・・・・・・・・・・・・・ 1460
SensorDBManager.ISensorDB.GetSensors Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1461
SensorDBManager.ISensorDB.GetSensorsBase Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 1462
SensorDBManager.ISensorDB.GetThermoCoupleTypes Method ・・・・・・・・・・・・・・・・・・・・・・・ 1463
ISensorDB.GetTranslations Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1464
SensorDBManager.ISensorDB.GetTranslations Method (Category) ・・・・・・・・・・・・・・・・・・・・・ 1465
SensorDBManager.ISensorDB.GetTranslations Method (Sensor) ・・・・・・・・・・・・・・・・・・・・・・ 1466
SensorDBManager.ISensorDB.GetVersionInfo Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1467
SensorDBManager.ISensorDB.HasChangedInDB Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 1468
SensorDBManager.ISensorDB.ImportCANDatabase Method ・・・・・・・・・・・・・・・・・・・・・・・・・ 1469
SensorDBManager.ISensorDB.InsertCategory Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1470
ISensorDB.InsertDescriptor Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1471
SensorDBManager.ISensorDB.InsertDescriptor Method (Descriptor) ・・・・・・・・・・・・・・・・・・・・・ 1472

SensorDBManager.ISensorDB.InsertDescriptor Method (Descriptor, byte[]) 1473
SensorDBManager.ISensorDB.InsertLanguage Method 1474
SensorDBManager.ISensorDB.InsertSensor Method 1475
SensorDBManager.ISensorDB.InsertThermoCoupleType Method 1476
SensorDBManager.ISensorDB.SaveVersion Method 1477
SensorDBManager.ISensorDB.SensorExist Method 1478
SensorDBManager.ISensorDB.SetActiveLanguage Method 1479
SensorDBManager.ISensorDB.SetFallbackLanguage Method 1480
SensorDBManager.ISensorDB.ShrinkDatabase Method 1481
SensorDBManager.ISensorDB.UpdateCategory Method 1482
ISensorDB.UpdateDescriptor Method 1483
SensorDBManager.ISensorDB.UpdateDescriptor Method (Descriptor) 1484
SensorDBManager.ISensorDB.UpdateDescriptor Method (Descriptor, byte[]) 1485
SensorDBManager.ISensorDB.UpdateLanguage Method 1486
SensorDBManager.ISensorDB.UpdateSensor Method 1487
SensorDBManager.ISensorDB.UpdateTranslation Method 1488
SensorDBManager.XmlDeserializeFromFile Method 1489
SensorDBManager.XmlDeserializeFromString Method 1490
SensorDBManager.XmlSerializeToFile Method 1491
SensorDBManager.XmlSerializeToString Method 1492
SensorDBManager Properties 1493
SensorDBManager.ISensorDB.ConnectionString Property 1494
SensorDBManager.ISensorDB.MaxAttachmentLength Property 1495
SensorDBManager.ISensorDB.ProviderInvariantName Property 1496
SensorDBManager.ThisInterface Property 1497
SensorFactory Class 1498
SensorFactory Members 1499
SensorFactory Methods 1500
SensorFactory.CreateSensor Method 1501
ISensorDB Interface 1502
ISensorDB Members 1503
ISensorDB Methods 1505
ISensorDB.CategoryExist Method 1507
ISensorDB.ClearConnectionPool Method 1508
ISensorDB.DeleteCategories Method 1509
ISensorDB.DeleteCategory Method 1510
ISensorDB.DeleteDescriptor Method 1511
ISensorDB.DeleteLanguage Method 1512
ISensorDB.DeleteSensor Method 1513
ISensorDB.DeleteSensors Method 1514
ISensorDB.DeleteThermoCoupleType Method 1515
ISensorDB.GetActiveLanguage Method 1516
ISensorDB.GetAttachment Method 1517
ISensorDB.GetCategories Method 1518
ISensorDB.GetCategory Method 1519
ISensorDB.GetCategoryByInternalID Method 1520
ISensorDB.GetDescriptor Method 1521
GetDescriptors Method 1522
ISensorDB.GetDescriptors Method () 1523
ISensorDB.GetDescriptors Method (Sensor) 1524
ISensorDB.GetFallbackLanguage Method 1525
ISensorDB.GetHierarchicalCategories Method 1526
ISensorDB.GetLanguage Method 1527
ISensorDB.GetLanguages Method 1528
GetSensor Method 1529
ISensorDB.GetSensor Method (int) 1530
ISensorDB.GetSensor Method (string) 1531
GetSensors Method 1532
ISensorDB.GetSensors Method () 1533
ISensorDB.GetSensors Method (Category) 1534
ISensorDB.GetSensorsBase Method 1535
ISensorDB.GetThermoCoupleTypes Method 1536
GetTranslations Method 1537

ISensorDB.GetTranslations Method (Category) ・・・

PointScalingBase Class ・・・

ZeroSpanScaling Class ・・・

CounterSensor.Polarity Property ・・・

FrequencySensor.GlitchFilter Property ・・・

ResistanceSensor.NominalResistance Property ・・・

HierarchicalCategory Constructor ・・・

Translation.Translation Constructor (int, string, string) ・・・

CategoryNotAssignedException.CategoryNotAssignedException Constructor (string, System.Exception) ・・・・・・・・・ 2053
FallbackLanguageDeleteException Class ・・・・・・・・・

CommonAPIのバージョン3.0より、センサデータベースへのアクセスがサポートされました。センサとスケーリングタイプの定義に加えて、この名前空間にはSensorDBManagerオブジェクトが含まれています。このオブジェクトは、センサの取得、挿入、更新、削除を可能にするISensorDBインタフェースを実装し、センサデータベース内のセンサの階層カテゴリ(グループ)をさまざまな言語で管理することを可能にします。

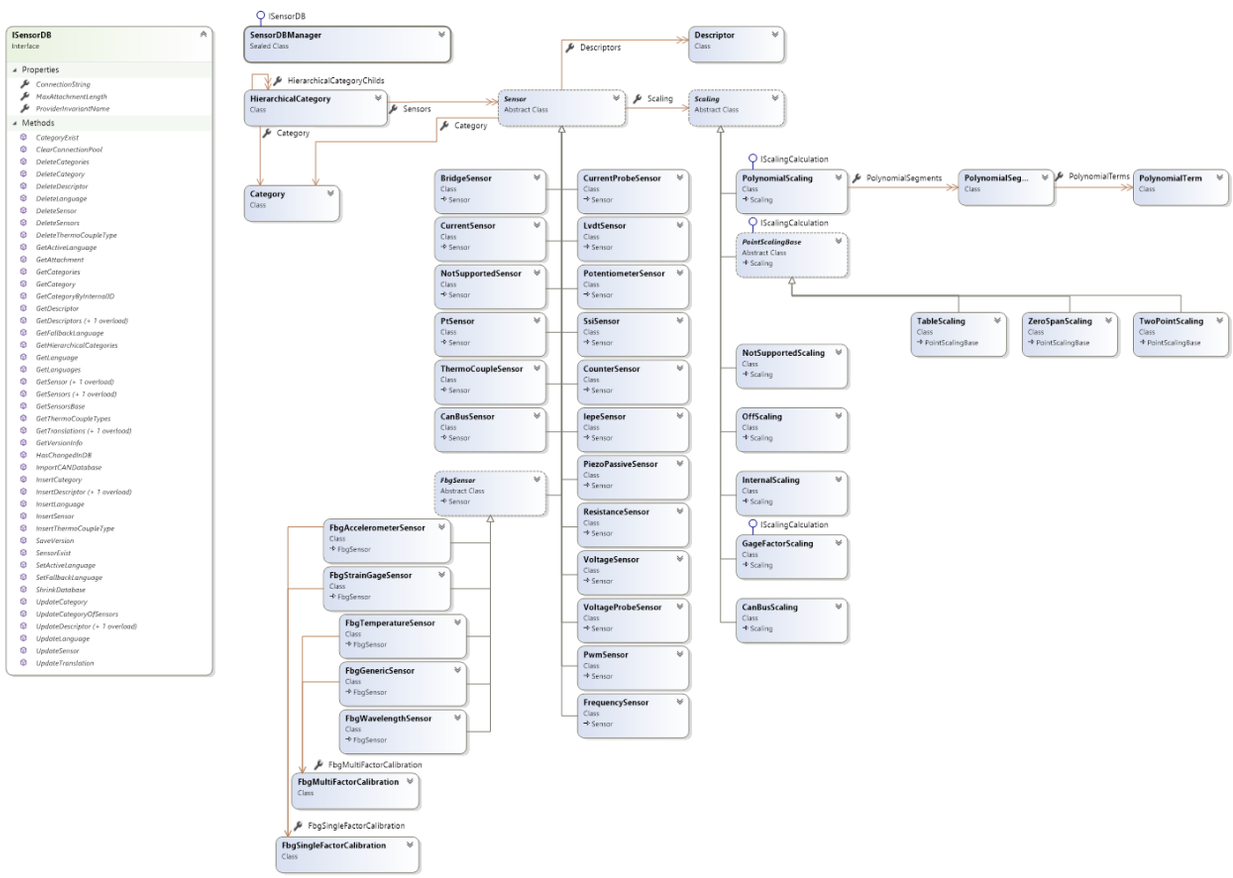
APIには、ほとんどのHBMセンサとユーザデータベースの空のテンプレートを含むHBMセンサデータベースが含まれています。Hbm.Api.DemoProjectは、これらのデータベースにアクセスする方法を示しています。

注意:

HBMセンサデータベースは変更しないでください。センサを変更したり、独自のカテゴリを構築したりする場合は、提供されているユーザデータベースのテンプレートをコピーして、そのコピーを使用してください。Common APIを今後インストールすると、新しいセンサや変更されたセンサが含まれる現在のHBMセンサデータベースが上書きされる可能性があります。

FBGセンサはセンサデータベースではサポートされていません。センサデータベースにロードしたり、センサデータベースに保存したりすることはできません。それでも、他のすべてのセンサタイプと同様に、それらをインスタンス化して使用することができます。

次の図は、関連する型とその最も重要なプロパティの要約を示しています:



次のコード スニペットは、その方法を示しています:

- HBMセンサデータベースに接続
- カテゴリとセンサの完全な階層を取得
- データベースから特定のセンサを取得

例

```
// Open HBM sensor database that is located in the same directory as the executable
string hbmSensorDatabaseFullFilename = System.IO.Path.Combine(System.IO.Directory.
GetCurrentDirectory(),
     "HBMSensorDatabase.vdb5");

// Create a sensorDBManager to get access to the sensor database (readonly)
ISensorDB sensorDBManagerForHbmSensorDatabase = new SensorDBManager("en", "Data Source=" +
hbmSensorDatabaseFullFilename +
     ";Open Mode=NonExclusiveReadOnly;", "System.Data.VistaDB5");
// Output some important settings
Debug.Print("Database: HBMSensorDatabase.vdb5");
Debug.Print("DatabaseProvioder: " + sensorDBManagerForHbmSensorDatabase.
ProviderInvariantName);
Debug.Print("Connectionstring:  " + sensorDBManagerForHbmSensorDatabase.ConnectionString);
Debug.Print("Database version:  " + sensorDBManagerForHbmSensorDatabase.GetVersionInfo().
Description + " V." +
   sensorDBManagerForHbmSensorDatabase.GetVersionInfo().VersionNumber);

// Get the complete hierarchy of the sensor database including all categories/subcategories
and their sensors
// just add a breakpoint here and explore hierarchicalCategoryNodes:
List<HierarchicalCategory> hierarchicalCategoryNodes = sensorDBManagerForHbmSensorDatabase.
GetHierarchicalCategories();
// whithin each hierarchicalCategoryNode you will find a list of sensor objects that belongs
to this category

Sensor sensor;
// there are various ways to get a sensor...
// e.g. by getting a list of all sensors of the sensor database (do that only once...) and
search
// for a specific sensor on your own...(here: get a list of all sensors and search for the
first voltage sensor)
List<Sensor> listOfAllSensors= sensorDBManagerForHbmSensorDatabase.GetSensors();
sensor = listOfAllSensors.Where(p => p.SensorType== SensorDB.Enums.SensorType.Voltage).
First();
// or e.g. let the sensorDBManager search for a sensor with a certain ID
sensor = sensorDBManagerForHbmSensorDatabase.GetSensor(112);
// or e.g. let the sensorDBManager search for a sensor with a certain unique name
sensor = sensorDBManagerForHbmSensorDatabase.GetSensor("HBM_T22_50Nm");
```

次のコード スニペットは、その方法を示しています:
•ユーザセンサデータベースに接続
•カテゴリとセンサの完全な階層を取得
•センサを特定のカテゴリに挿入
•既存のセンサを更新
•センサを削除

例

```
// Open user sensor database that is located in the same directory as the executable
string userSensorDatabaseFullFilename = System.IO.Path.Combine(System.IO.Directory.
GetCurrentDirectory(),
     "UserSensorDatabase.vdb5");

// Create a sensorDBManager to get access to the sensor database (read/write access)
ISensorDB sensorDBManagerForUserSensorDatabase = new SensorDBManager("en", "Data Source=" +
userSensorDatabaseFullFilename +
     ";Open Mode=ExclusiveReadWrite;", "System.Data.VistaDB5");

Debug.Print("Database:     " + userSensorDatabaseFullFilename);
Debug.Print("DatabaseProvioder: " + sensorDBManagerForUserSensorDatabase.
ProviderInvariantName);
Debug.Print("Connectionstring:  " + sensorDBManagerForUserSensorDatabase.ConnectionString);
Debug.Print("Database version:  " + sensorDBManagerForUserSensorDatabase.GetVersionInfo().
Description + " V." +
   sensorDBManagerForUserSensorDatabase.GetVersionInfo().VersionNumber);

// Get the complete hierarchy of the sensor database including all categories/subcategories
and their sensors
// just add a breakpoint here and explore nodes:
List<HierarchicalCategory> nodes = sensorDBManagerForUserSensorDatabase.
GetHierarchicalCategories();

// Insert a sensor into a certain category of the database

// Therefore we have to do 2 things here:
// 1. Create a sensor object (we use the SensorFactory here to get a certain sensortype
with default scaling...)
VoltageSensor myVoltageSensor= SensorFactory.CreateSensor(SensorDB.Enums.SensorType.
Voltage) as VoltageSensor;
myVoltageSensor.Description = "my voltage sensor"; // set the name of the sensor and
further properties if you like

// 2. To insert a sensor into a sensor database, we have to choose an existing category
(group) of the sensor database into
// which the sensor should be inserted.
// Here we insert the sensor into the "Imported" category of the database (this category
exists in each user sensor database).
Category categoryIntoWhichToInsertTheSensor = sensorDBManagerForUserSensorDatabase.
GetCategories().
     Where(c => c.Description == "Imported").First();
// Set the Category property of the sensor we want to insert
myVoltageSensor.Category = categoryIntoWhichToInsertTheSensor;
// Insert the sensor into the user database
sensorDBManagerForUserSensorDatabase.InsertSensor(myVoltageSensor);
// ATTENTION! After successfully inserting the sensor, some properties of the sensor
changed!!!
// Take a look at following sensor properties:  SensorID, UniqueName, IsUserDefined,
LastModifiedDate...

// To update the sensor we just change its description here
myVoltageSensor.Description = "my updated voltage sensor";
// and update the (now) already existing sensor in the sensor database... (Description AND
LastModifiedDate will be changed!)
sensorDBManagerForUserSensorDatabase.UpdateSensor(myVoltageSensor);

// To delete the sensor we just call
sensorDBManagerForUserSensorDatabase.DeleteSensor(myVoltageSensor.SensorID);
```

名前空間

名前	説明
Hbm.Api.SensorDB	これは名前空間Hbm.Api.SensorDBです。
Hbm.Api.SensorDB.Entities	これは名前空間Hbm.Api.SensorDB.Entitiesです。
Hbm.Api.SensorDB.Entities.FbgCalibration	これは名前空間Hbm.Api.SensorDB.Entities.FbgCalibrationです。
Hbm.Api.SensorDB.Enums	これは名前空間Hbm.Api.SensorDB.Enumsです。
Hbm.Api.SensorDB.Exceptions	これは名前空間Hbm.Api.SensorDB.Exceptionsです。

Hbm.Api.SensorDB Namespace

これは名前空間Hbm.Api.SensorDBです。

クラス

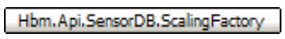
	名前	説明
S	ScalingFactory	スケーリングインスタンスを作成するためのファクトリ。
	SensorDBManager	センサデータベースAPIの完全な機能にアクセスするためのファサード。インタフェースISensorDBを明示的に実装します。
S	SensorFactory	センサインスタンスを作成するためのファクトリ。

インタフェース

	名前	説明
	ISensorDB	センサデータベースAPIのインタフェース。

ScalingFactory Class

スケーリングインスタンスを作成するためのファクトリ。



C++

```
class ScalingFactory;
```

C#

```
public static class ScalingFactory;
```

Visual Basic

```
Public static Class ScalingFactory
```

ファイル

ScalingFactory.cs

ScalingFactory Members

次の表は、ScalingFactoryによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CreateScaling	指定したスケーリング タイプに従って具体的なスケーリングインスタンスを作成し、スケーリング タイプを設定します。

ScalingFactory Methods

ScalingFactoryクラスのメソッドの一覧です。

Public Methods

	名前	説明
	CreateScaling	指定したスケーリング タイプに従って具体的なスケーリングインスタンスを作成し、スケーリング タイプを設定します。

ScalingFactory.CreateScaling Method

指定したスケーリング タイプに従って具体的なスケーリングインスタンスを作成し、スケーリング タイプを設定します。

C++

```
Scaling CreateScaling(
    ScalingType scalingType
);
```

C#

```
public static Scaling CreateScaling(
    ScalingType scalingType
);
```

Visual Basic

```
Public static Function CreateScaling(
    scalingType As ScalingType
) As Scaling
```

引数

引数	説明
scalingType	具体的なスケーリングを識別するスケーリングタイプ。

戻り値

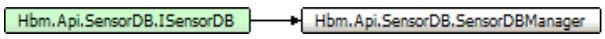
具体的なスケーリングインスタンス。

例外

例外	説明
System.NotImplementedException	実装されていないスケーリングタイプに対してスローされます。

SensorDBManager Class

センサデータベースAPIの完全な機能にアクセスするためのファサード。インタフェースISensorDBを明示的に実装します。



C++

```
class SensorDBManager : public ISensorDB;
```

C#

```
public sealed class SensorDBManager : ISensorDB;
```

Visual Basic

```
Public sealed Class SensorDBManager
Inherits ISensorDB
```

ファイル

SensorDBManager.cs

備考

インタフェースISensorDBは、このクラスに明示的に実装する必要があります。

例

```
ISensorDB sensorDBManager = new SensorDBManager();</p>
Sensor sensor = sensorDBManager.GetSensor("My unique sensor name");</p>
```

SensorDBManager Members

次の表は、SensorDBManagerによって公開されるメンバーの一覧です。

Private Methods

	名前	説明
	CheckForInvalidChildCategoryDeletion	階層カテゴリを反復して、指定した削除リストが削除に有効かどうかを判断します。deleteCategoryIdsToCheckで指定したカテゴリにセンサがあるかどうかをチェックし、サブカテゴリもリストに含まれているかどうかをチェックします(主キー違反がないことを確認します)。
	GenerateXmlExtraTypes	XMLシリアル化を目的として、すべての追加タイプ(Sensorクラスと Scalingクラス)のリストを動的に生成します。追加の型がないと、継承を使用する場合(センサやスケーリングなど)、派生クラスのXMLシリアル化が失敗します。
	GenerateXmlKnownScalingTypes	ScalingTypeインスタンスのすべてのタイプのリストを動的に生成します。
	GenerateXmlKnownSensorTypes	SensorTypeインスタンスのすべてのタイプのリストを動的に生成します。
	GetSensorsForCategory	指定したカテゴリの子であるすべてのセンサを取得します。
	ISensorDB.CategoryExist	指定したカテゴリIDによるカテゴリがデータベースに存在するかどうかをチェックします。
	ISensorDB.ClearConnectionPool	基礎となる接続に関連付けられている接続プールを空にします。
	ISensorDB.DeleteCategory	指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。
	ISensorDB.DeleteDescriptor	指定した記述子IDを持つ記述子(添付ファイルを含む)をデータベースから削除します。
	ISensorDB.DeleteLanguage	指定したISO言語名(2文字)の言語をデータベースから削除します。
	ISensorDB.DeleteSensor	指定したセンサIDを持つセンサをデータベースから削除します。
	ISensorDB.DeleteSensors	指定したセンサIDを持つセンサをデータベースから削除します。
	ISensorDB.DeleteThermoCoupleType	指定した熱電対タイプをデータベースから削除します。
	ISensorDB.GetActiveLanguage	アクティブな言語の言語インスタンスをデータベースから読み取ります。
	ISensorDB.GetAttachment	指定した記述子IDに従って添付ファイルをデータベースから読み取ります。
	ISensorDB.GetCategories	データベースからすべてのカテゴリを読み取ります。
	ISensorDB.GetCategory	指定したカテゴリIDを持つカテゴリをデータベースから読み取ります。
	ISensorDB.GetCategoryByInternalID	指定した内部IDを持つカテゴリをデータベースから読み取ります。
	ISensorDB.GetDescriptor	指定した記述子IDを持つ記述子をデータベースから読み取ります。
	ISensorDB.GetDescriptors	これは、ISensorDB.GetDescriptorsメソッドの多重定義の概要です。
	ISensorDB.GetFallbackLanguage	データベースからフォールバック言語を読み取ります。
	ISensorDB.GetHierarchicalCategories	データベースからすべての階層カテゴリを読み取ります。階層カテゴリは、センサデータベースコンテンツのツリー構造を作成するために使用します。たとえば、後でこれらの階層要素をビジュアルツリーコントロールにマッピングする場合などに使用します。
	ISensorDB.GetLanguage	指定された ISO 言語名 (2 文字) の言語をデータベースから読み取ります。
	ISensorDB.GetLanguages	データベースからすべての言語を読み取ります。
	ISensorDB.GetSensor	これは、ISensorDB.GetSensorメソッドの多重定義の概要です。
	ISensorDB.GetSensors	データベースからすべてのセンサを読み取ります。

	名前	説明
	ISensorDB.GetSensorsBase	データベースからすべてのセンサのリストとその ID、名前、説明を返します。すべてのセンサデータ(特定のセンサデータなど)を含むすべてのセンサのリストを取得する場合は、代わりにHbm.Api.SensorDB.ISensorDB.GetSensors()を使用します。
	ISensorDB.GetThermoCoupleTypes	データベースからすべての熱電対タイプを読み取ります。
	ISensorDB.GetTranslations	これは、ISensorDB.GetTranslationsメソッドの多重定義の概要です。
	ISensorDB.GetVersionInfo	データベースからセンサーデータベースのバージョン情報を読み取ります。
	ISensorDB.HasChangedInDB	指定したセンサインスタンスが、データベース内の同じSensorIDを持つセンサと異なるかどうかをチェックします。`LastModifiedDate` プロパティが異なる場合、センサには変更されています。
	ISensorDB.ImportCANDatabase	指定したファイル名からCANデータベースをインポートします。
	ISensorDB.InsertCategory	指定したカテゴリをデータベースに挿入します。
	ISensorDB.InsertDescriptor	これは、ISensorDB.InsertDescriptorメソッドの多重定義の概要です。
	ISensorDB.InsertLanguage	指定した言語をデータベースに挿入します。フォールバック言語のすべての翻訳がコピーされ、この新しい言語に挿入されます。
	ISensorDB.InsertSensor	指定したセンサをデータベースに挿入します。
	ISensorDB.InsertThermoCoupleType	指定した熱電対タイプをデータベースに挿入します。既存する熱電対タイプは更新せず、削除してから新しいタイプを挿入する必要があります。
	ISensorDB.SaveVersion	センサデータベースのバージョン情報をデータベースに保存します。
	ISensorDB.SensorExist	指定したセンサIDのセンサがデータベース内に存在するかどうかを確認します。
	ISensorDB.SetActiveLanguage	現在のアクティブな言語を設定します(データベースには保存されません)。
	ISensorDB.SetFallbackLanguage	指定した言語をフォールバック言語として設定し、データベース内の他の言語からフラグを削除します。
	ISensorDB.ShrinkDatabase	データベースページを再編成して空き領域を回復します。
	ISensorDB.UpdateCategory	データベース内の指定したカテゴリを更新します。
	ISensorDB.UpdateDescriptor	これは、ISensorDB.UpdateDescriptorメソッドの多重定義の概要です。
	ISensorDB.UpdateLanguage	指定した言語の説明を更新します。
	ISensorDB.UpdateSensor	データベース内の既存のセンサを更新します。
	ISensorDB.UpdateTranslation	データベース内の指定した翻訳を更新します。

Private Properties

	名前	説明
	ISensorDB.ConnectionString	データベースに接続するための接続文字列。デフォルト値: `"Data Source=HBMSensorDatabase.vdb5"`
R	ISensorDB.MaxAttachmentLength	添付ファイルの最大長を取得または設定します。
	ISensorDB.ProviderInvariantName	データベースプロバイダのプロバイダインバリアント名。デフォルトはVistaDBインバリアントのプロバイダ名です。
R	ThisInterface	現在のインスタンス(this)を明示的に実装されたインタフェースISensorDBにキャストすることをカプセル化します。このプロパティを使用すると、インタフェースへの追加のキャストは必要ありません。

Public Methods

	名前	説明
	BuildOrderedDeleteCategoryIdListInForeignKeyOrder	このメソッドは、正しい削除順序を提供するために、階層レベルでソートされた辞書を構築します。これにより、削除時に'主キー違反がない'ことが保証されます。有効なカテゴリIDリストを確認するには、その前にCheckForInvalidChildCategoryDeletionを呼び出してください!
	CategoryExist	指定したカテゴリIDによるカテゴリがデータベースに存在するかどうかをチェックします。
	ClearConnectionPool	基礎となる接続に関連付けられている接続プールを空にします。
	DeleteCategories	指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。
	DeleteCategory	指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。
	DeleteDescriptor	指定した記述子IDを持つ記述子(添付ファイルを含む)をデータベースから削除します。
	DeleteLanguage	指定したISO言語名(2文字)の言語をデータベースから削除します。
	DeleteOrphanTranslations	テーブル"Translation"からすべての孤立エントリを削除します。データベースレコード(カテゴリまたはセンサ)がデータベースから手動で削除した場合、孤立エントリが発生する可能性があります。VistaDBの制限により、カテゴリとその翻訳、およびセンサとその翻訳の間の自動データ整合性を維持するためのトリガは使用されません。
	DeleteSensor	指定したセンサIDを持つセンサをデータベースから削除します。
	DeleteSensors	指定したセンサIDを持つセンサをデータベースから削除します。
	DeleteThermoCoupleType	指定した熱電対タイプをデータベースから削除します。
	GetActiveLanguage	アクティブな言語(SetActiveLanguageによって設定)の言語インスタンスをデータベースから読み取ります。
	GetAttachment	指定した記述子IDに従って添付ファイルをデータベースから読み取ります。
	GetCategories	データベースからすべてのカテゴリを読み取ります。
	GetCategory	指定したカテゴリIDを持つカテゴリをデータベースから読み取ります。
	GetCategoryByInternalID	指定した内部IDを持つカテゴリをデータベースから読み取ります。
	GetDescriptor	指定した記述子IDを持つ記述子をデータベースから読み取ります。
	GetDescriptors	これは、GetDescriptorsメソッドの多重定義の概要です。
	GetFallbackLanguage	データベースからフォールバック言語を読み取ります。
	GetHierarchicalCategories	データベースからすべての階層カテゴリを読み取ります。階層カテゴリは、センサデータベースコンテンツのツリー構造を作成するために使用します。
	GetLanguage	指定したISO言語名(2文字)の言語をデータベースから読み取ります。
	GetLanguages	データベースからすべての言語を読み取ります。
	GetSensor	これは、GetSensorメソッドの多重定義の概要です。
	GetSensors	これは、GetSensorsメソッドの多重定義の概要です。
	GetSensorsBase	データベースからすべてのセンサのリストとその ID、名前、説明を返します。すべてのセンサデータ(特定のセンサデータなど)を含むすべてのセンサのリストを取得する場合は、代わりにHbm.Api.SensorDB.ISensorDB.GetSensors()を使用します。
	GetThermoCoupleTypes	データベースからすべての熱電対タイプを読み取ります。
	GetTranslations	これは、GetTranslationsメソッドの多重定義の概要です。
	GetVersionInfo	データベースからセンサーデータベースのバージョン情報を読み取ります。
	HasChangedInDB	指定したセンサインスタンスが、データベース内の同じSensorIDを持つセンサと異なるかどうかをチェックします。`LastModifiedDate`プロパティが異なる場合、センサには変更されています。

	名前	説明
	ImportCANDatabase	指定したファイル名からCANデータベースをインポートします。
	InsertCategory	指定したカテゴリをデータベースに挿入します。
	InsertDescriptor	これは、InsertDescriptorメソッドの多重定義の概要です。
	InsertLanguage	指定した言語をデータベースに挿入します。フォールバック言語のすべての翻訳がコピーされ、この新しい言語に挿入されます。
	InsertSensor	指定したセンサをデータベースに挿入します。
	InsertThermoCoupleType	指定した熱電対タイプをデータベースに挿入します。既存する熱電対タイプは更新せず、削除してから新しいタイプを挿入する必要があります。
	SaveVersion	センサデータベースのバージョン情報をデータベースに保存します。
	SensorDBManager	これは、SensorDBManagerコンストラクタの多重定義の概要です。
	SensorExist	指定したセンサIDのセンサがデータベース内に存在するかどうかを確認します。
	SetActiveLanguage	現在のアクティブな言語を設定します(データベースには保存されません)。
	SetFallbackLanguage	指定した言語をフォールバック言語として設定し、データベース内の他の言語からフラグを削除します。
	ShrinkDatabase	データベースページを再編成して空き領域を回復します。
	UpdateCategory	データベース内の指定したカテゴリを更新します。
	UpdateCategoryOfSensors	指定したセンサIDの親カテゴリを更新します。
	UpdateDescriptor	これは、UpdateDescriptorメソッドの多重定義の概要です。
	UpdateLanguage	指定した言語の説明を更新します。
	UpdateSensor	データベース内の既存のセンサを更新します。
	UpdateTranslation	データベース内の指定した翻訳を更新します。
	XmlDeserializeFromFile	XMLファイルからインスタンスをデシリアライズします。
	XmlDeserializeFromString	XML文字列からインスタンスをデシリアライズします。
	XmlSerializeToFile	インスタンスをXMLファイルにシリアライズします。
	XmlSerializeToString	インスタンスを文字列にシリアライズします。

Public Properties

	名前	説明
	ConnectionString	データベースに接続するための接続文字列。
	MaxAttachmentLength	添付ファイルの最大長を取得または設定します。
	ProviderInvariantName	データベースプロバイダのプロバイダインバリアント名。

SensorDBManager Constructor

これは、SensorDBManagerメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	SensorDBManager. SensorDBManager ()	新しいインスタンスを作成し、デフォルト値(データベース接続とユーザ言語)を設定します。
	SensorDBManager. SensorDBManager (string)	新しいインスタンスを作成し、指定したISO言語名を現在の言語として設定します。データベース接続にはデフォルト値が使用されます。
	SensorDBManager. SensorDBManager (string, string, string)	新しいインスタンスを作成し、指定したISO言語名、接続文字列、およびデータベースプロバイダを設定します。

SensorDBManager.SensorDBManager Constructor ()

新しいインスタンスを作成し、デフォルト値(データベース接続とユーザ言語)を設定します。

C++

```
SensorDBManager();
```

C#

```
public SensorDBManager();
```

Visual Basic

```
Public Sub New()
```

SensorDBManager.SensorDBManager Constructor (string)

新しいインスタンスを作成し、指定したISO言語名を現在の言語として設定します。データベース接続にはデフォルト値が使用されます。

C++

```
SensorDBManager(
    string twoLetterISOLanguageName
);
```

C#

```
public SensorDBManager(
    string twoLetterISOLanguageName
);
```

Visual Basic

```
Public Sub New(
    twoLetterISOLanguageName As string
)
```

引数

引数	説明
twoLetterISOLanguageName	使用言語のISO言語名(2文字)。

SensorDBManager.SensorDBManager Constructor (string, string, string)

新しいインスタンスを作成し、指定したISO言語名、接続文字列、およびデータベースプロバイダを設定します。

C++

```
SensorDBManager(
    string twoLetterISOLanguageName,
    string connectionString,
    string providerInvariantName
);
```

C#

```
public SensorDBManager(
    string twoLetterISOLanguageName,
    string connectionString,
    string providerInvariantName
);
```

Visual Basic

```
Public Sub New(
    twoLetterISOLanguageName As string,
    connectionString As string,
    providerInvariantName  As string
)
```

引数

引数	説明
twoLetterISOLanguageName	使用言語のISO言語名(2文字)。
connectionString	データベースに接続するための接続文字列。
providerInvariantName	データベースプロバイダのプロバイダインバリアント名(例: "System.Data.VistaDB5"または"System.Data.SqlClient")。

SensorDBManager Methods

SensorDBManagerクラスのメソッドの一覧です。

Private Methods

	名前	説明
	CheckForInvalidChildCategoryDeletion	階層カテゴリを反復して、指定した削除リストが削除に有効かどうかを判断します。deleteCategoryIdsToCheckで指定したカテゴリにセンサがあるかどうかをチェックし、サブカテゴリもリストに含まれているかどうかをチェックします(主キー違反がないことを確認します)。
	GenerateXmlExtraTypes	XMLシリアル化を目的として、すべての追加タイプ(Sensorクラスと Scalingクラス)のリストを動的に生成します。追加の型がないと、継承を使用する場合(センサやスケーリングなど)、派生クラスのXMLシリアル化が失敗します。
	GenerateXmlKnownScalingTypes	ScalingTypeインスタンスのすべてのタイプのリストを動的に生成します。
	GenerateXmlKnownSensorTypes	SensorTypeインスタンスのすべてのタイプのリストを動的に生成します。
	GetSensorsForCategory	指定したカテゴリの子であるすべてのセンサを取得します。
	ISensorDB.CategoryExist	指定したカテゴリIDによるカテゴリがデータベースに存在するかどうかをチェックします。
	ISensorDB.ClearConnectionPool	基礎となる接続に関連付けられている接続プールを空にします。
	ISensorDB.DeleteCategory	指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。
	ISensorDB.DeleteDescriptor	指定した記述子IDを持つ記述子(添付ファイルを含む)をデータベースから削除します。
	ISensorDB.DeleteLanguage	指定したISO言語名(2文字)の言語をデータベースから削除します。
	ISensorDB.DeleteSensor	指定したセンサIDを持つセンサをデータベースから削除します。
	ISensorDB.DeleteSensors	指定したセンサIDを持つセンサをデータベースから削除します。
	ISensorDB.DeleteThermoCoupleType	指定した熱電対タイプをデータベースから削除します。
	ISensorDB.GetActiveLanguage	アクティブな言語の言語インスタンスをデータベースから読み取ります。
	ISensorDB.GetAttachment	指定した記述子IDに従って添付ファイルをデータベースから読み取ります。
	ISensorDB.GetCategories	データベースからすべてのカテゴリを読み取ります。
	ISensorDB.GetCategory	指定したカテゴリIDを持つカテゴリをデータベースから読み取ります。
	ISensorDB.GetCategoryByInternalID	指定した内部IDを持つカテゴリをデータベースから読み取ります。
	ISensorDB.GetDescriptor	指定した記述子IDを持つ記述子をデータベースから読み取ります。
	ISensorDB.GetDescriptors	これは、ISensorDB.GetDescriptorsメソッドの多重定義の概要です。
	ISensorDB.GetFallbackLanguage	データベースからフォールバック言語を読み取ります。
	ISensorDB.GetHierarchicalCategories	データベースからすべての階層カテゴリを読み取ります。階層カテゴリは、センサデータベースコンテンツのツリー構造を作成するために使用します。たとえば、後でこれらの階層要素をビジュアルツリーコントロールにマッピングする場合などに使用します。
	ISensorDB.GetLanguage	指定された ISO 言語名 (2 文字) の言語をデータベースから読み取ります。
	ISensorDB.GetLanguages	データベースからすべての言語を読み取ります。
	ISensorDB.GetSensor	これは、ISensorDB.GetSensorメソッドの多重定義の概要です。
	ISensorDB.GetSensors	データベースからすべてのセンサを読み取ります。

	名前	説明
	ISensorDB.GetSensorsBase	データベースからすべてのセンサのリストとその ID、名前、説明を返します。すべてのセンサデータ(特定のセンサデータなど)を含むすべてのセンサのリストを取得する場合は、代わりにHbm.Api.SensorDB.ISensorDB.GetSensors()を使用します。
	ISensorDB.GetThermoCoupleTypes	データベースからすべての熱電対タイプを読み取ります。
	ISensorDB.GetTranslations	これは、ISensorDB.GetTranslationsメソッドの多重定義の概要です。
	ISensorDB.GetVersionInfo	データベースからセンサーデータベースのバージョン情報を読み取ります。
	ISensorDB.HasChangedInDB	指定したセンサインスタンスが、データベース内の同じSensorIDを持つセンサと異なるかどうかをチェックします。`LastModifiedDate`プロパティが異なる場合、センサには変更されています。
	ISensorDB.ImportCANDatabase	指定したファイル名からCANデータベースをインポートします。
	ISensorDB.InsertCategory	指定したカテゴリをデータベースに挿入します。
	ISensorDB.InsertDescriptor	これは、ISensorDB.InsertDescriptorメソッドの多重定義の概要です。
	ISensorDB.InsertLanguage	指定した言語をデータベースに挿入します。フォールバック言語のすべての翻訳がコピーされ、この新しい言語に挿入されます。
	ISensorDB.InsertSensor	指定したセンサをデータベースに挿入します。
	ISensorDB.InsertThermoCoupleType	指定した熱電対タイプをデータベースに挿入します。既存する熱電対タイプは更新せず、削除してから新しいタイプを挿入する必要があります。
	ISensorDB.SaveVersion	センサデータベースのバージョン情報をデータベースに保存します。
	ISensorDB.SensorExist	指定したセンサIDのセンサがデータベース内に存在するかどうかを確認します。
	ISensorDB.SetActiveLanguage	現在のアクティブな言語を設定します(データベースには保存されません)。
	ISensorDB.SetFallbackLanguage	指定した言語をフォールバック言語として設定し、データベース内の他の言語からフラグを削除します。
	ISensorDB.ShrinkDatabase	データベースページを再編成して空き領域を回復します。
	ISensorDB.UpdateCategory	データベース内の指定したカテゴリを更新します。
	ISensorDB.UpdateDescriptor	これは、ISensorDB.UpdateDescriptorメソッドの多重定義の概要です。
	ISensorDB.UpdateLanguage	指定した言語の説明を更新します。
	ISensorDB.UpdateSensor	データベース内の既存のセンサを更新します。
	ISensorDB.UpdateTranslation	データベース内の指定した翻訳を更新します。

Public Methods

	名前	説明
	BuildOrderedDeleteCategoryIdListInForeignKeyOrder	このメソッドは、正しい削除順序を提供するために、階層レベルでソートされた辞書を構築します。これにより、削除時に'主キー違反がない'ことが保証されます。有効なカテゴリIDリストを確認するには、その前にCheckForInvalidChildCategoryDeletionを呼び出してください!
	DeleteOrphanTranslations	テーブル"Translation"からすべての孤立エントリを削除します。データベースレコード(カテゴリまたはセンサ)がデータベースから手動で削除した場合、孤立エントリが発生する可能性があります。VistaDBの制限により、カテゴリとその翻訳、およびセンサとその翻訳の間の自動データ整合性を維持するためのトリガは使用されません。
	XmlDeserializeFromFile	XMLファイルからインスタンスをデシリアライズします。
	XmlDeserializeFromString	XML文字列からインスタンスをデシリアライズします。
	XmlSerializeToFile	インスタンスをXMLファイルにシリアライズします。
	XmlSerializeToString	インスタンスを文字列にシリアライズします。

SensorDBManager.BuildOrderedDeleteCategoryIdListInForeignKeyOrder Method

このメソッドは、正しい削除順序を提供するために、階層レベルでソートされた辞書を構築します。これにより、削除時に'主キー違反がない'ことが保証されます。有効なカテゴリIDリストを確認するには、その前にCheckForInvalidChildCategoryDeletionを呼び出してください!

C++

```
SortedDictionary<int, List<int>> BuildOrderedDeleteCategoryIdListInForeignKeyOrder(
    HierarchicalCategory rootCategory,
    List<int> categoryIdsToPushInDeleteList
);
```

C#

```
public SortedDictionary<int, List<int>> BuildOrderedDeleteCategoryIdListInForeignKeyOrder(
    HierarchicalCategory rootCategory,
    List<int> categoryIdsToPushInDeleteList
);
```

Visual Basic

```
Public Function BuildOrderedDeleteCategoryIdListInForeignKeyOrder(
    rootCategory As HierarchicalCategory,
    categoryIdsToPushInDeleteList As List<int>,
)As SortedDictionary<int, List<int>>
```

引数

引数	説明
rootCategory	ツリーのルートカテゴリ。
categoryIdsToPushInDeleteList	削除するカテゴリIDを含むリスト。完全かつ一貫性のあるリストである必要があります!

戻り値

レベル別に削除可能なカテゴリIDの順序付きリスト。

備考

ツリー反復(下から上へ、再帰なし): a b c d e f g h i => efbgchida

SensorDBManager.CheckForInvalidChildCategoryDeletion Method

階層カテゴリを反復して、指定した削除リストが削除に有効かどうかを判断します。deleteCategoryIdsToCheckで指定したカテゴリにセンサがあるかどうかをチェックし、サブカテゴリもリストに含まれているかどうかをチェックします(主キー違反がないことを確認します)。

C++

```
void CheckForInvalidChildCategoryDeletion(
    List<int> deleteCategoryIdsToCheck,
    HierarchicalCategory categoryToCheck,
    Category ^ invalidDeleteOfCategory
);
```

C#

```
private void CheckForInvalidChildCategoryDeletion(
    List<int> deleteCategoryIdsToCheck,
    HierarchicalCategory categoryToCheck,
    ref Category invalidDeleteOfCategory
);
```

Visual Basic

```
Private Function CheckForInvalidChildCategoryDeletion(
    deleteCategoryIdsToCheck As List<int>,
    categoryToCheck As HierarchicalCategory,
    ByRef invalidDeleteOfCategory As Category
)As void
```

引数

引数	説明
deleteCategoryIdsToCheck	チェック対象のカテゴリIDが含まれる削除リスト。
categoryToCheck	チェックする現在の階層カテゴリ。
invalidDeleteOfCategory	このオブジェクトは、a) カテゴリにセンサがある場合、b) 少なくとも1つの子カテゴリが削除対象としてマークされていない場合、リスト内の無効なカテゴリIDによって設定されます。初期状態の場合はnullを渡します。

SensorDBManager.DeleteOrphanTranslations Method

テーブル"Translation"からすべての孤立エントリを削除します。データベースレコード(カテゴリまたはセンサ)がデータベースから手動で削除した場合、孤立エントリが発生する可能性があります。VistaDBの制限により、カテゴリとその翻訳、およびセンサとその翻訳の間の自動データ整合性を維持するためのトリガは使用されません。

C++

```
[EditorBrowsable(EditorBrowsableState.Never)]
public void DeleteOrphanTranslations();
```

C#

```
[EditorBrowsable(EditorBrowsableState.Never)]
public void DeleteOrphanTranslations();
```

Visual Basic

```
<EditorBrowsable(EditorBrowsableState.Never)>
Public Function DeleteOrphanTranslations()As void
```

備考

この管理方法は、ISensorDBインタフェースの一部ではありません。

SensorDBManager.GenerateXmlExtraTypes Method

XMLシリアル化を目的として、すべての追加タイプ(Sensorクラスと Scalingクラス)のリストを動的に生成します。追加の型がないと、継承を使用する場合(センサやスケーリングなど)、派生クラスのXMLシリアル化が失敗します。

C++

```
Type[] GenerateXmlExtraTypes();
```

C#

```
private Type[] GenerateXmlExtraTypes();
```

Visual Basic

```
Private Function GenerateXmlExtraTypes()As Type()
```

戻り値

すべての追加タイプの型付き配列。

SensorDBManager.GenerateXmlKnownScalingTypes Method

ScalingTypeインスタンスのすべてのタイプのリストを動的に生成します。

C++

```
List<Type> GenerateXmlKnownScalingTypes();
```

C#

```
private List<Type> GenerateXmlKnownScalingTypes();
```

Visual Basic

```
Private Function GenerateXmlKnownScalingTypes()As List<Type>
```

戻り値

すべての ScalingTypeインスタンスの型のリスト。

現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB Namespace -> SensorDBManager Class -> SensorDBManager Methods

SensorDBManager.GenerateXmlKnownSensorTypes Method

SensorTypeインスタンスのすべてのタイプのリストを動的に生成します。

C++

```
List<Type> GenerateXmlKnownSensorTypes();
```

C#

```
private List<Type> GenerateXmlKnownSensorTypes();
```

Visual Basic

```
Private Function GenerateXmlKnownSensorTypes()As List<Type>
```

戻り値

すべての SensorTypeインスタンス の型のリスト。

SensorDBManager.GetSensorsForCategory Method

指定したカテゴリの子であるすべてのセンサを取得します。

C++

```
List<Sensor> GetSensorsForCategory(
    Category category,
    List<Sensor> allSensors
);
```

C#

```
private List<Sensor> GetSensorsForCategory(
    Category category,
    List<Sensor> allSensors
);
```

Visual Basic

```
Private Function GetSensorsForCategory(
    category As Category,
    allSensors As List<Sensor>
)As List<Sensor>
```

引数

引数	説明
category	検索条件として使用するHbm.Api.SensorDB.Entities.Categoryのインスタンス。
allSensors	すべてのセンサのリスト。

戻り値

Hbm.Api.SensorDB.Entities.Sensorインスタンスのリスト。

SensorDBManager.ISensorDB.CategoryExist Method

指定したカテゴリIDによるカテゴリがデータベースに存在するかどうかをチェックします。

C++

```
bool ISensorDB.CategoryExist(
    int categoryId
);
```

C#

```
private bool ISensorDB.CategoryExist(
    int categoryId
);
```

Visual Basic

```
Private Function ISensorDB.CategoryExist(
    categoryId As Integer
)As bool
```

引数

引数	説明
categoryId	チェックするカテゴリのID。

戻り値

カテゴリが存在する場合はtrue、存在しない場合はfalse。

SensorDBManager.ISensorDB.ClearConnectionPool Method

基礎となる接続に関連付けられている接続プールを空にします。

C++

```
void ISensorDB.ClearConnectionPool();
```

C#

```
private void ISensorDB.ClearConnectionPool();
```

Visual Basic

```
Private Function ISensorDB.ClearConnectionPool()As void
```

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.DeleteCategory Method

指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。

C++

```
void ISensorDB.DeleteCategory(
    int categoryID
);
```

C#

```
private void ISensorDB.DeleteCategory(
    int categoryID
);
```

Visual Basic

```
Private Function ISensorDB.DeleteCategory(
    categoryID As Integer
)As void
```

引数

引数	説明
categoryID	削除するカテゴリのID。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException	カテゴリに子カテゴリまたは子センサがある場合にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.DeleteDescriptor Method

指定した記述子IDを持つ記述子(添付ファイルを含む)をデータベースから削除します。

C++

```
void ISensorDB.DeleteDescriptor(
    int descriptorID
);
```

C#

```
private void ISensorDB.DeleteDescriptor(
    int descriptorID
);
```

Visual Basic

```
Private Function ISensorDB.DeleteDescriptor(
    descriptorID As Integer
)As void
```

引数

引数	説明
descriptorID	削除する記述子のID。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.DeleteLanguage Method

指定したISO言語名(2文字)の言語をデータベースから削除します。

C++

```
void ISensorDB.DeleteLanguage(
    string twoLetterISOLanguageName
);
```

C#

```
private void ISensorDB.DeleteLanguage(
    string twoLetterISOLanguageName
);
```

Visual Basic

```
Private Function ISensorDB.DeleteLanguage(
    twoLetterISOLanguageName As string
)As void
```

引数

引数	説明
twoLetterISOLanguageName	削除する言語のISO言語名(2文字)。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.FallbackLanguageDeleteException	フォールバック言語を削除しようとした場合にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.DeleteSensor Method

指定したセンサIDを持つセンサをデータベースから削除します。

C++

```
void ISensorDB.DeleteSensor(
    int sensorID
);
```

C#

```
private void ISensorDB.DeleteSensor(
    int sensorID
);
```

Visual Basic

```
Private Function ISensorDB.DeleteSensor(
    sensorID As Integer
)As void
```

引数

引数	説明
sensorID	削除するセンサのID。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.DeleteSensors Method

指定したセンサIDを持つセンサをデータベースから削除します。

C++

```
void ISensorDB.DeleteSensors(
    List<int> sensorIds
);
```

C#

```
private void ISensorDB.DeleteSensors(
    List<int> sensorIds
);
```

Visual Basic

```
Private Function ISensorDB.DeleteSensors(
    sensorIds As List<int>
)As void
```

引数

引数	説明
sensorIds	削除するセンサのID。

例外

例外	説明
System.ArgumentNullException	sensorIdsがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.DeleteThermoCoupleType Method

指定した熱電対タイプをデータベースから削除します。

C++

```
void ISensorDB.DeleteThermoCoupleType(
    string thermoCoupleType
);
```

C#

```
private void ISensorDB.DeleteThermoCoupleType(
    string thermoCoupleType
);
```

Visual Basic

```
Private Function ISensorDB.DeleteThermoCoupleType(
    thermoCoupleType As string
)As void
```

引数

引数	説明
thermoCoupleType	削除する熱電対タイプ。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetActiveLanguage Method

アクティブな言語の言語インスタンスをデータベースから読み取ります。

C++

```
Language ISensorDB.GetActiveLanguage();
```

C#

```
private Language ISensorDB.GetActiveLanguage();
```

Visual Basic

```
Private Function ISensorDB.GetActiveLanguage()As Language
```

戻り値

Hbm.Api.SensorDB.Entities.Languageのインスタンス。見つからない場合はnull。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetAttachment Method

指定した記述子IDに従って添付ファイルをデータベースから読み取ります。

C++

```
byte[] ISensorDB.GetAttachment(
    int descriptorID
);
```

C#

```
private byte[] ISensorDB.GetAttachment(
    int descriptorID
);
```

Visual Basic

```
Private Function ISensorDB.GetAttachment(
    descriptorID As Integer
)As byte()
```

引数

引数	説明
descriptorID	記述子のID。

戻り値

添付ファイルは`byte[]`として返されます。見つからない場合は`null`になります。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetCategories Method

データベースからすべてのカテゴリを読み取ります。

C++

```
List<Category> ISensorDB.GetCategories();
```

C#

```
private List<Category> ISensorDB.GetCategories();
```

Visual Basic

```
Private Function ISensorDB.GetCategories()As List<Category>
```

戻り値

Hbm.Api.SensorDB.Entities.Categoryインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetCategory Method

指定したカテゴリIDを持つカテゴリをデータベースから読み取ります。

C++

```
Category ISensorDB.GetCategory(
    int categoryID
);
```

C#

```
private Category ISensorDB.GetCategory(
    int categoryID
);
```

Visual Basic

```
Private Function ISensorDB.GetCategory(
categoryID As Integer
)As Category
```

引数

引数	説明
categoryID	カテゴリのID。

戻り値

Hbm.Api.SensorDB.Entities.Categoryインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetCategoryByInternalID Method

指定した内部IDを持つカテゴリをデータベースから読み取ります。

C++

```
Category ISensorDB.GetCategoryByInternalID(
    int internalID
);
```

C#

```
private Category ISensorDB.GetCategoryByInternalID(
    int internalID
);
```

Visual Basic

```
Private Function ISensorDB.GetCategoryByInternalID(
internalID As Integer
)As Category
```

引数

引数	説明
internalID	カテゴリの内部ID。

戻り値

Hbm.Api.SensorDB.Entities.Categoryインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetDescriptor Method

指定した記述子IDを持つ記述子をデータベースから読み取ります。

C++

```
Descriptor ISensorDB.GetDescriptor(
    int descriptorID
);
```

C#

```
private Descriptor ISensorDB.GetDescriptor(
    int descriptorID
);
```

Visual Basic

```
Private Function ISensorDB.GetDescriptor(
descriptorID As Integer
)As Descriptor
```

引数

引数	説明
descriptorID	記述子のID。

戻り値

Hbm.Api.SensorDB.Entities.Descriptorインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetDescriptors Method

これは、ISensorDB.GetDescriptorsメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	SensorDBManager.ISensorDB.GetDescriptors ()	データベースからすべての記述子を読み取ります。
	SensorDBManager.ISensorDB.GetDescriptors (Sensor)	指定したセンサの記述子をデータベースから読み取ります。

SensorDBManager.ISensorDB.GetDescriptors Method ()

データベースからすべての記述子を読み取ります。

C++

```
List<Descriptor> ISensorDB.GetDescriptors();
```

C#

```
private List<Descriptor> ISensorDB.GetDescriptors();
```

Visual Basic

```
Private Function ISensorDB.GetDescriptors()As List<Descriptor>
```

戻り値

Hbm.Api.SensorDB.Entities.Descriptorインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetDescriptors Method (Sensor)

指定したセンサの記述子をデータベースから読み取ります。

C++

```
List<Descriptor> ISensorDB.GetDescriptors(
    Sensor sensor
);
```

C#

```
private List<Descriptor> ISensorDB.GetDescriptors(
    Sensor sensor
);
```

Visual Basic

```
Private Function ISensorDB.GetDescriptors(
    sensor As Sensor
)As List<Descriptor>
```

引数

引数	説明
sensor	記述子の検索対象となるHbm.Api.SensorDB.Entities.Sensorのインスタンス。

戻り値

Hbm.Api.SensorDB.Entities.Descriptorインスタンスのリスト。

例外

例外	説明
System.ArgumentNullException	sensorがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetFallbackLanguage Method

データベースからフォールバック言語を読み取ります。

C++

```
Language ISensorDB.GetFallbackLanguage();
```

C#

```
private Language ISensorDB.GetFallbackLanguage();
```

Visual Basic

```
Private Function ISensorDB.GetFallbackLanguage()As Language
```

戻り値

Hbm.Api.SensorDB.Entities.Languageインスタンス。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetHierarchicalCategories Method

データベースからすべての階層カテゴリを読み取ります。階層カテゴリは、センサデータベースコンテンツのツリー構造を作成するために使用します。たとえば、後でこれらの階層要素をビジュアルツリーコントロールにマッピングする場合などに使用します。

C++

```
List<HierarchicalCategory> ISensorDB.GetHierarchicalCategories();
```

C#

```
private List<HierarchicalCategory> ISensorDB.GetHierarchicalCategories();
```

Visual Basic

```
Private Function ISensorDB.GetHierarchicalCategories()As List<HierarchicalCategory>
```

戻り値

Hbm.Api.SensorDB.Entities.HierarchicalCategoryインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetLanguage Method

指定したISO言語名(2文字)の言語をデータベースから読み取ります。

C++

```
Language ISensorDB.GetLanguage(
    string twoLetterISOLanguageName
);
```

C#

```
private Language ISensorDB.GetLanguage(
    string twoLetterISOLanguageName
);
```

Visual Basic

```
Private Function ISensorDB.GetLanguage(
    twoLetterISOLanguageName As string
)As Language
```

引数

引数	説明
twoLetterISOLanguageName	ISO言語名(2文字)。

戻り値

Hbm.Api.SensorDB.Entities.Languageインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetLanguages Method

データベースからすべての言語を読み取ります。

C++

```
List<Language> ISensorDB.GetLanguages();
```

C#

```
private List<Language> ISensorDB.GetLanguages();
```

Visual Basic

```
Private Function ISensorDB.GetLanguages()As List<Language>
```

戻り値

Hbm.Api.SensorDB.Entities.Languageインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetSensor Method

これは、ISensorDB.GetSensorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	SensorDBManager.ISensorDB.GetSensor (int)	指定したセンサIDを持つセンサをデータベースから読み取ります。
	SensorDBManager.ISensorDB.GetSensor (string)	指定した固有の名前を持つセンサをデータベースから読み取ります。

SensorDBManager.ISensorDB.GetSensor Method (int)

指定したセンサIDを持つセンサをデータベースから読み取ります。

C++

```
Sensor ISensorDB.GetSensor(
    int sensorID
);
```

C#

```
private Sensor ISensorDB.GetSensor(
    int sensorID
);
```

Visual Basic

```
Private Function ISensorDB.GetSensor(
    sensorID As Integer
)As Sensor
```

引数

引数	説明
sensorID	センサのID。

戻り値

Hbm.Api.SensorDB.Entities.Sensorインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetSensor Method (string)

指定した固有の名前を持つセンサをデータベースから読み取ります。

C++

```
Sensor ISensorDB.GetSensor(
    string uniqueName
);
```

C#

```
private Sensor ISensorDB.GetSensor(
    string uniqueName
);
```

Visual Basic

```
Private Function ISensorDB.GetSensor(
    uniqueName As string
)As Sensor
```

引数

引数	説明
uniqueName	センサ固有の名前。

戻り値

Hbm.Api.SensorDB.Entities.Sensorインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetSensors Method

データベースからすべてのセンサを読み取ります。

C++

```
List<Sensor> ISensorDB.GetSensors();
```

C#

```
private List<Sensor> ISensorDB.GetSensors();
```

Visual Basic

```
Private Function ISensorDB.GetSensors()As List<Sensor>
```

戻り値

Hbm.Api.SensorDB.Entities.Sensorインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetSensorsBase Method

データベースからすべてのセンサのリストとその ID、名前、説明を返します。すべてのセンサデータ(特定のセンサデータなど)を含むすべてのセンサのリストを取得する場合は、代わりにHbm.Api.SensorDB.ISensorDB.GetSensors()を使用します。

C++

```
List<Sensor> ISensorDB.GetSensorsBase();
```

C#

```
private List<Sensor> ISensorDB.GetSensorsBase();
```

Visual Basic

```
Private Function ISensorDB.GetSensorsBase()As List<Sensor>
```

戻り値

Hbm.Api.SensorDB.Entities.Sensorインスタンスのリスト。

SensorDBManager.ISensorDB.GetThermoCoupleTypes Method

データベースからすべての熱電対タイプを読み取ります。

C++

```
List<string> ISensorDB.GetThermoCoupleTypes();
```

C#

```
private List<string> ISensorDB.GetThermoCoupleTypes();
```

Visual Basic

```
Private Function ISensorDB.GetThermoCoupleTypes()As List<string>
```

戻り値

すべての熱電対タイプの文字列のリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetTranslations Method

これは、ISensorDB.GetTranslationsメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	SensorDBManager.ISensorDB.GetTranslations (Category)	指定したカテゴリの翻訳をデータベースから読み取ります。
	SensorDBManager.ISensorDB.GetTranslations (Sensor)	指定したセンサの翻訳をデータベースから読み取ります。

SensorDBManager.ISensorDB.GetTranslations Method (Category)

指定したカテゴリの翻訳をデータベースから読み取ります。

C++

```
List<Translation> ISensorDB.GetTranslations(
    Category category
);
```

C#

```
private List<Translation> ISensorDB.GetTranslations(
    Category category
);
```

Visual Basic

```
Private Function ISensorDB.GetTranslations(
    category As Category
)As List<Translation>
```

引数

引数	説明
category	翻訳の検索対象となるHbm.Api.SensorDB.Entities.Categoryのインスタンス。

戻り値

カテゴリのHbm.Api.SensorDB.Entities.Translationインスタンスのリスト。

例外

例外	説明
System.ArgumentNullException	categoryがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetTranslations Method (Sensor)

指定したセンサの翻訳をデータベースから読み取ります。

C++

```
List<Translation> ISensorDB.GetTranslations(
    Sensor sensor
);
```

C#

```
private List<Translation> ISensorDB.GetTranslations(
    Sensor sensor
);
```

Visual Basic

```
Private Function ISensorDB.GetTranslations(
    sensor As Sensor
)As List<Translation>
```

引数

引数	説明
sensor	翻訳の検索対象となるHbm.Api.SensorDB.Entities.Sensorのインスタンス。

戻り値

センサのHbm.Api.SensorDB.Entities.Translationインスタンスのリスト。

例外

例外	説明
System.ArgumentNullException	sensorがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.GetVersionInfo Method

データベースからセンサーデータベースのバージョン情報を読み取ります。

C++

```
VersionInfo ISensorDB.GetVersionInfo();
```

C#

```
private VersionInfo ISensorDB.GetVersionInfo();
```

Visual Basic

```
Private Function ISensorDB.GetVersionInfo()As VersionInfo
```

戻り値

バージョン情報。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.HasChangedInDB Method

指定したセンサインスタンスが、データベース内の同じSensorIDを持つセンサと異なるかどうかをチェックします。
`LastModifiedDate`プロパティが異なる場合、センサには変更されています。

C++

```
bool ISensorDB.HasChangedInDB(
    Sensor sensor
);
```

C#

```
private bool ISensorDB.HasChangedInDB(
    Sensor sensor
);
```

Visual Basic

```
Private Function ISensorDB.HasChangedInDB(
    sensor As Sensor
)As bool
```

引数

引数	説明
sensor	データベースと比較するHbm.Api.SensorDB.Entities.Sensorのインスタンス。

戻り値

センサに変更があった場合は`true`を返し、そうでない場合は`false`を返します。センサインスタンスが指定されていない場合は`false`を返します。センサインスタンスがデータベースに見つからなかった場合は`true`を返します(新しいセンサ)。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.ImportCANDatabase Method

指定したファイル名からCANデータベースをインポートします。

C++

```
void ISensorDB.ImportCANDatabase(
    string filename,
    Category rootCategory,
    bool useCompoundNames
);
```

C#

```
private void ISensorDB.ImportCANDatabase(
    string filename,
    Category rootCategory,
    bool useCompoundNames
);
```

Visual Basic

```
Private Function ISensorDB.ImportCANDatabase(
    filename As string,
    rootCategory As Category,
    useCompoundNames As bool
)As void
```

引数

引数	説明
filename	CANデータベースのパスとファイル名。
rootCategory	インポートしたデータのルートカテゴリ。
useCompoundNames	センサの説明に複合名を生成するかどうかのフラグ。

SensorDBManager.ISensorDB.InsertCategory Method

指定したカテゴリをデータベースに挿入します。

C++

```
void ISensorDB.InsertCategory(
    Category category
);
```

C#

```
private void ISensorDB.InsertCategory(
    Category category
);
```

Visual Basic

```
Private Function ISensorDB.InsertCategory(
    category As Category
)As void
```

引数

引数	説明
category	挿入するHbm.Api.SensorDB.Entities.Categoryのインスタンス。

例外

例外	説明
System.ArgumentNullException	categoryがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.InsertDescriptor Method

これは、ISensorDB.InsertDescriptorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	SensorDBManager.ISensorDB.InsertDescriptor (Descriptor)	指定した記述子を指定された添付ファイル(存在する場合)とともにデータベースに挿入します。
	SensorDBManager.ISensorDB.InsertDescriptor (Descriptor, byte[])	指定した記述子を指定された添付ファイル(存在する場合)とともにデータベースに挿入します。

SensorDBManager.ISensorDB.InsertDescriptor Method (Descriptor)

指定した記述子を指定された添付ファイル(存在する場合)とともにデータベースに挿入します。

C++

```
void ISensorDB.InsertDescriptor(
    Descriptor descriptor
);
```

C#

```
private void ISensorDB.InsertDescriptor(
    Descriptor descriptor
);
```

Visual Basic

```
Private Function ISensorDB.InsertDescriptor(
    descriptor As Descriptor
)As void
```

引数

引数	説明
descriptor	挿入するHbm.Api.SensorDB.Entities.Descriptorのインスタンス。

例外

例外	説明
System.ArgumentNullException	記述子がnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.InsertDescriptor Method (Descriptor, byte[])

指定した記述子を指定された添付ファイル(存在する場合)とともにデータベースに挿入します。

C++

```
void ISensorDB.InsertDescriptor(
    Descriptor descriptor,
    byte[] attachment
);
```

C#

```
private void ISensorDB.InsertDescriptor(
    Descriptor descriptor,
    byte[] attachment
);
```

Visual Basic

```
Private Function ISensorDB.InsertDescriptor(
    descriptor As Descriptor,
    attachment As byte[]
)As void
```

引数

引数	説明
descriptor	挿入するHbm.Api.SensorDB.Entities.Descriptorのインスタンス。
attachment	記述子とともに挿入するbyte[]としての添付ファイル。

例外

例外	説明
System.ArgumentNullException	記述子がnullです。 - または - 添付ファイルがnullです。
Hbm.Api.SensorDB.Exceptions.AttachmentToLargeException	添付ファイルが大きすぎる場合(> 1GB)にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.InsertLanguage Method

指定した言語をデータベースに挿入します。フォールバック言語のすべての翻訳がコピーされ、この新しい言語に挿入されます。

C++

```
void ISensorDB.InsertLanguage(
    Language language
);
```

C#

```
private void ISensorDB.InsertLanguage(
    Language language
);
```

Visual Basic

```
Private Function ISensorDB.InsertLanguage(
    language As Language
)As void
```

引数

引数	説明
descriptor	挿入するHbm.Api.SensorDB.Entities.Languageのインスタンス。

例外

例外	説明
System.ArgumentNullException	languageが`null`です。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.InsertSensor Method

指定したセンサをデータベースに挿入します。

C++

```
void ISensorDB.InsertSensor(
    Sensor sensor
);
```

C#

```
private void ISensorDB.InsertSensor(
    Sensor sensor
);
```

Visual Basic

```
Private Function ISensorDB.InsertSensor(
    sensor As Sensor
)As void
```

引数

引数	説明
sensor	挿入するHbm.Api.SensorDB.Entities.Sensorのインスタンス。

例外

例外	説明
System.ArgumentNullException	sensorがnullです。
Hbm.Api.SensorDB.Exceptions.CategoryNotAssignedException	センサにカテゴリインスタンスが割り当てられていない場合にスローされます。
Hbm.Api.SensorDB.Exceptions.ScalingNotAssignedException	センサにスケーリングインスタンスが割り当てられていない場合にスローされます。
Hbm.Api.SensorDB.Exceptions.ScalingNotSupportedException	割り当てられたスケーリングの型がサポートされていないHbm.Api.SensorDB.Entities.Scalings.NotSupportedScalingの場合にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorNotSupportedException	センサの型がHbm.Api.SensorDB.Entities.Sensors.NotSupportedSensorである場合にスローされますが、これはサポートされていません。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.InsertThermoCoupleType Method

指定した熱電対タイプをデータベースに挿入します。既存する熱電対タイプは更新せず、削除してから新しいタイプを挿入する必要があります。

C++

```
void ISensorDB.InsertThermoCoupleType(
    string thermoCoupleType
);
```

C#

```
private void ISensorDB.InsertThermoCoupleType(
    string thermoCoupleType
);
```

Visual Basic

```
Private Function ISensorDB.InsertThermoCoupleType(
    thermoCoupleType As string
)As void
```

引数

引数	説明
thermoCoupleType	挿入する新しい熱電対タイプ。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.SaveVersion Method

センサデータベースのバージョン情報をデータベースに保存します。

C++

```
void ISensorDB.SaveVersion(
    VersionInfo versionInfo
);
```

C#

```
private void ISensorDB.SaveVersion(
    VersionInfo versionInfo
);
```

Visual Basic

```
Private Function ISensorDB.SaveVersion(
    versionInfo As VersionInfo
)As void
```

引数

引数	説明
versionInfo	保存するバージョン情報。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.SensorExist Method

指定したセンサIDのセンサがデータベース内に存在するかどうかを確認します。

C++

```
bool ISensorDB.SensorExist(
    int sensorId
);
```

C#

```
private bool ISensorDB.SensorExist(
    int sensorId
);
```

Visual Basic

```
Private Function ISensorDB.SensorExist(
    sensorId As Integer
)As bool
```

引数

引数	説明
sensorId	チェックするセンサのID。

戻り値

センサが存在する場合はtrue、存在しない場合はfalse。

SensorDBManager.ISensorDB.SetActiveLanguage Method

現在のアクティブな言語を設定します(データベースには保存されません)。

C++

```
bool ISensorDB.SetActiveLanguage(
    string twoLetterISOLanguageName
);
```

C#

```
private bool ISensorDB.SetActiveLanguage(
    string twoLetterISOLanguageName
);
```

Visual Basic

```
Private Function ISensorDB.SetActiveLanguage(
    twoLetterISOLanguageName As string
)As bool
```

引数

引数	説明
twoLetterISOLanguageName	現在の言語のISO言語名(2文字)。

戻り値

現在の言語が存在し、設定されている場合は`true`、そうでない場合は`false`。現在の言語が存在しない(したがって`false`を取得)場合は、フォールバック言語がアクティブ言語として設定されます。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.SetFallbackLanguage Method

指定した言語をフォールバック言語として設定し、データベース内の他の言語からフラグを削除します。

C++

```
void ISensorDB.SetFallbackLanguage(
    Language fallbackLanguage
);
```

C#

```
private void ISensorDB.SetFallbackLanguage(
    Language fallbackLanguage
);
```

Visual Basic

```
Private Function ISensorDB.SetFallbackLanguage(
    fallbackLanguage As Language
)As void
```

引数

引数	説明
fallbackLanguage	フォールバック言語として設定されるHbm.Api.SensorDB.Entities.Languageのインスタンス。

例外

例外	説明
System.ArgumentNullException	fallbackLanguageが`null`です。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.ShrinkDatabase Method

データベースページを再編成して空き領域を回復します。

C++

```
void ISensorDB.ShrinkDatabase();
```

C#

```
private void ISensorDB.ShrinkDatabase();
```

Visual Basic

```
Private Function ISensorDB.ShrinkDatabase()As void
```

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.UpdateCategory Method

データベース内の指定したカテゴリを更新します。

C++

```
void ISensorDB.UpdateCategory(
    Category category
);
```

C#

```
private void ISensorDB.UpdateCategory(
    Category category
);
```

Visual Basic

```
Private Function ISensorDB.UpdateCategory(
    category As Category
)As void
```

引数

引数	説明
category	更新するHbm.Api.SensorDB.Entities.Categoryのインスタンス。

例外

例外	説明
System.ArgumentNullException	categoryがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.UpdateDescriptor Method

これは、ISensorDB.UpdateDescriptorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	SensorDBManager.ISensorDB.UpdateDescriptor (Descriptor)	添付ファイルを含めずに指定した記述子をデータベースに更新します。
	SensorDBManager.ISensorDB.UpdateDescriptor (Descriptor, byte[])	指定した添付ファイル(利用可能な場合)を使用して、指定した記述子をデータベースに更新します。

SensorDBManager.ISensorDB.UpdateDescriptor Method (Descriptor)

添付ファイルを含めずに指定した記述子をデータベースに更新します。

C++

```
void ISensorDB.UpdateDescriptor(
    Descriptor descriptor
);
```

C#

```
private void ISensorDB.UpdateDescriptor(
    Descriptor descriptor
);
```

Visual Basic

```
Private Function ISensorDB.UpdateDescriptor(
    descriptor As Descriptor
)As void
```

引数

引数	説明
descriptor	更新するHbm.Api.SensorDB.Entities.Descriptorのインスタンス。

例外

例外	説明
System.ArgumentNullException	descriptorがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.UpdateDescriptor Method (Descriptor, byte[])

指定した添付ファイル(利用可能な場合)を使用して、指定した記述子をデータベースに更新します。

C++

```
void ISensorDB.UpdateDescriptor(
    Descriptor descriptor,
    byte[] attachment
);
```

C#

```
private void ISensorDB.UpdateDescriptor(
    Descriptor descriptor,
    byte[] attachment
);
```

Visual Basic

```
Private Function ISensorDB.UpdateDescriptor(
    descriptor As Descriptor,
    attachment As byte[]
)As void
```

引数

引数	説明
descriptor	更新するHbm.Api.SensorDB.Entities.Descriptorのインスタンス。
attachment	記述子で更新されるbyte[]としての添付ファイル。

例外

例外	説明
System.ArgumentNullException	記述子がnullです。 - または - 添付ファイルがnullです。
Hbm.Api.SensorDB.Exceptions.AttachmentToLargeException	添付ファイルが大きすぎる場合(> 1GB)にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.UpdateLanguage Method

指定した言語の説明を更新します。

C++

```
void ISensorDB.UpdateLanguage(
    Language language
);
```

C#

```
private void ISensorDB.UpdateLanguage(
    Language language
);
```

Visual Basic

```
Private Function ISensorDB.UpdateLanguage(
    language As Language
)As void
```

引数

引数	説明
language	更新するHbm.Api.SensorDB.Entities.Languageのインスタンス。

例外

例外	説明
System.ArgumentNullException	languageがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.UpdateSensor Method

データベース内の既存のセンサを更新します。

C++

```
void ISensorDB.UpdateSensor(
    Sensor sensor
);
```

C#

```
private void ISensorDB.UpdateSensor(
    Sensor sensor
);
```

Visual Basic

```
Private Function ISensorDB.UpdateSensor(
    sensor As Sensor
)As void
```

引数

引数	説明
sensor	更新するHbm.Api.SensorDB.Entities.Sensorのインスタンス。

例外

例外	説明
System.ArgumentNullException	sensorがnullです。
Hbm.Api.SensorDB.Exceptions.CategoryNotAssignedException	センサにカテゴリインスタンスが割り当てられていない場合にスローされます。
Hbm.Api.SensorDB.Exceptions.ScalingNotAssignedException	センサにスケーリングインスタンスが割り当てられていない場合にスローされます。
Hbm.Api.SensorDB.Exceptions.ScalingNotSupportedException	割り当てられたスケーリングの型がサポートされていないHbm.Api.SensorDB.Entities.Scalings.NotSupportedScalingの場合にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorNotSupportedException	センサの型がHbm.Api.SensorDB.Entities.Sensors.NotSupportedSensorである場合にスローされますが、これはサポートされていません。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.ISensorDB.UpdateTranslation Method

データベース内の指定した翻訳を更新します。

C++

```
void ISensorDB.UpdateTranslation(
    Translation translation
);
```

C#

```
private void ISensorDB.UpdateTranslation(
    Translation translation
);
```

Visual Basic

```
Private Function ISensorDB.UpdateTranslation(
    translation As Translation
)As void
```

引数

引数	説明
translation	更新するHbm.Api.SensorDB.Entities.Translationのインスタンス。

備考

このメソッドは翻訳を更新するだけです。翻訳の挿入は、新しいセンサまたは新しいカテゴリを挿入することによって間接的に行われます。したがって、新しい翻訳を明示的に挿入するメソッドは必要ありません。

例外

例外	説明
System.ArgumentNullException	translationが`null`です。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

SensorDBManager.XmlDeserializeFromFile Method

XMLファイルからインスタンスをデシリアライズします。

C++

```
[EditorBrowsable(EditorBrowsableState.Never)]
T XmlDeserializeFromFile<T>(
    string filename
);
```

C#

```
[EditorBrowsable(EditorBrowsableState.Never)]
public T XmlDeserializeFromFile<T>(
    string filename
);
```

Visual Basic

```
<EditorBrowsable(EditorBrowsableState.Never)>
Public Function XmlDeserializeFromFile(Of T)(
    filename As string
)As T
```

引数

引数	説明
filename	XMLファイルのパスとファイル名。
T	ルートインスタンスの型。

戻り値

デシリアライズされたオブジェクトグラフ。

備考

この管理方法は、ISensorDBインタフェースの一部ではありません。

SensorDBManager.XmlDeserializeFromString Method

XML文字列からインスタンスをデシリアライズします。

C++

```
[EditorBrowsable(EditorBrowsableState.Never)]
T XmlDeserializeFromString<T>(
    string xmlContent
);
```

C#

```
[EditorBrowsable(EditorBrowsableState.Never)]
public T XmlDeserializeFromString<T>(
    string xmlContent
);
```

Visual Basic

```
<EditorBrowsable(EditorBrowsableState.Never)>
Public Function XmlDeserializeFromString(Of T)(
    xmlContent As string
)As T
```

引数

引数	説明
xmlContent	XMLコンテンツを含む文字列。
T	ルートインスタンスの型。

戻り値

デシリアライズされたオブジェクトグラフ。

備考

この管理方法は、ISensorDBインタフェースの一部ではありません。

SensorDBManager.XmlSerializeToFile Method

インスタンスをXMLファイルにシリアライズします。

C++

```
[EditorBrowsable(EditorBrowsableState.Never)]
void XmlSerializeToFile<T>(
    string filename,
    T instance
);
```

C#

```
[EditorBrowsable(EditorBrowsableState.Never)]
public void XmlSerializeToFile<T>(
    string filename,
    T instance
);
```

Visual Basic

```
<EditorBrowsable(EditorBrowsableState.Never)>
Public Function XmlSerializeToFile(Of T)(
    filename As string,
    instance As T
)As void
```

引数

引数	説明
filename	XMLファイルのパスとファイル名。
instance	ファイルに保存するインスタンス。
T	ルートインスタンスの型。

備考

この管理方法は、ISensorDBインタフェースの一部ではありません。

SensorDBManager.XmlSerializeToString Method

インスタンスを文字列にシリアライズします。

C++

```
[EditorBrowsable(EditorBrowsableState.Never)]
string XmlSerializeToString<T>(
    T instance
);
```

C#

```
[EditorBrowsable(EditorBrowsableState.Never)]
public string XmlSerializeToString<T>(
    T instance
);
```

Visual Basic

```
<EditorBrowsable(EditorBrowsableState.Never)>
Public Function XmlSerializeToString(Of T)(
    instance As T
)As string
```

引数

引数	説明
instance	文字列にシリアライズされるインスタンス。
T	ルートインスタンスの型。

備考

この管理方法は、ISensorDBインタフェースの一部ではありません。

SensorDBManager Properties

SensorDBManagerクラスのプロパティの一覧です。

Private Properties

	名前	説明
	ISensorDB.ConnectionString	データベースに接続するための接続文字列。デフォルト値: "Data Source=HBMSensorDatabase.vdb5"
R	ISensorDB.MaxAttachmentLength	添付ファイルの最大長を取得または設定します。
	ISensorDB.ProviderInvariantName	データベースプロバイダのプロバイダインバリアント名。デフォルトはVistaDBインバリアントのプロバイダ名です。
R	ThisInterface	現在のインスタンス(this)を明示的に実装されたインタフェースISensorDBにキャストすることをカプセル化します。このプロパティを使用すると、インタフェースへの追加のキャストは必要ありません。

SensorDBManager.ISensorDB.ConnectionString Property

データベースに接続するための接続文字列。デフォルト値: "Data Source=HBMSensorDatabase.vdb5"

C++

```
__property string ISensorDB.ConnectionString;
```

C#

```
private string ISensorDB.ConnectionString;
```

Visual Basic

```
Private Property ISensorDB.ConnectionString() As string
```

SensorDBManager.ISensorDB.MaxAttachmentLength Property

添付ファイルの最大長を取得または設定します。

C++

```
__property int ISensorDB.MaxAttachmentLength;
```

C#

```
private int ISensorDB.MaxAttachmentLength;
```

Visual Basic

```
Private Property ISensorDB.MaxAttachmentLength() As Integer
```

説明

添付ファイルの最大長。

SensorDBManager.ISensorDB.ProviderInvariantName Property

データベースプロバイダのプロバイダインバリアント名。デフォルトはVistaDBインバリアントのプロバイダ名です。

C++

```
__property string ISensorDB.ProviderInvariantName;
```

C#

```
private string ISensorDB.ProviderInvariantName;
```

Visual Basic

```
Private Property ISensorDB.ProviderInvariantName() As string
```

SensorDBManager.ThisInterface Property

現在のインスタンス(this)を明示的に実装されたインタフェースISensorDBにキャストすることをカプセル化します。このプロパティを使用すると、インタフェースへの追加のキャストは必要ありません。

C++

```
__property ISensorDB ThisInterface;
```

C#

```
private ISensorDB ThisInterface;
```

Visual Basic

```
Private Property ThisInterface() As ISensorDB
```

SensorFactory Class

センサインスタンスを作成するためのファクトリ。



C++

```
class SensorFactory;
```

C#

```
public static class SensorFactory;
```

Visual Basic

```
Public static Class SensorFactory
```

ファイル

SensorFactory.cs

SensorFactory Members

次の表は、SensorFactoryによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CreateSensor	指定したセンサタイプに従って具体的なセンサインスタンスを作成し、Hbm.Api.SensorDB.Entities.Sensor.SensorTypeとHbm.Api.SensorDB.Entities.Sensor.UniqueNameを設定します。

SensorFactory Methods

SensorFactoryクラスのメソッドの一覧です。

Public Methods

	名前	説明
	CreateSensor	指定したセンサタイプに従って具体的なセンサインスタンスを作成し、Hbm.Api.SensorDB.Entities.Sensor.SensorTypeとHbm.Api.SensorDB.Entities.Sensor.UniqueNameを設定します。

SensorFactory.CreateSensor Method

指定したセンサタイプに従って具体的なセンサインスタンスを作成し、Hbm.Api.SensorDB.Entities.Sensor.SensorTypeとHbm.Api.SensorDB.Entities.Sensor.UniqueNameを設定します。

C++

```
Sensor CreateSensor(
    SensorType sensorType
);
```

C#

```
public static Sensor CreateSensor(
    SensorType sensorType
);
```

Visual Basic

```
Public static Function CreateSensor(
    sensorType As SensorType
)As Sensor
```

引数

引数	説明
sensorType	具体的なセンサを識別するセンサタイプ。

戻り値

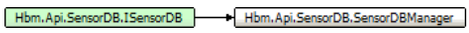
具体的なセンサインスタンス。

例外

例外	説明
System.NotImplementedException	実装されていないスケーリングタイプに対してスローされます。

ISensorDB Interface

センサデータベースAPIのインタフェース。



C++

```
__interface ISensorDB;
```

C#

```
public interface ISensorDB;
```

Visual Basic

```
Public Interface ISensorDB
```

ファイル

ISensorDB.cs

ISensorDB Members

次の表は、ISensorDBによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CategoryExist	指定したカテゴリIDによるカテゴリがデータベースに存在するかどうかをチェックします。
	ClearConnectionPool	基礎となる接続に関連付けられている接続プールを空にします。
	DeleteCategories	指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。
	DeleteCategory	指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。
	DeleteDescriptor	指定した記述子IDを持つ記述子(添付ファイルを含む)をデータベースから削除します。
	DeleteLanguage	指定したISO言語名(2文字)の言語をデータベースから削除します。
	DeleteSensor	指定したセンサIDを持つセンサをデータベースから削除します。
	DeleteSensors	指定したセンサIDを持つセンサをデータベースから削除します。
	DeleteThermoCoupleType	指定した熱電対タイプをデータベースから削除します。
	GetActiveLanguage	アクティブな言語(SetActiveLanguageによって設定)の言語インスタンスをデータベースから読み取ります。
	GetAttachment	指定した記述子IDに従って添付ファイルをデータベースから読み取ります。
	GetCategories	データベースからすべてのカテゴリを読み取ります。
	GetCategory	指定したカテゴリIDを持つカテゴリをデータベースから読み取ります。
	GetCategoryByInternalID	指定した内部IDを持つカテゴリをデータベースから読み取ります。
	GetDescriptor	指定した記述子IDを持つ記述子をデータベースから読み取ります。
	GetDescriptors	これは、GetDescriptorsメソッドの多重定義の概要です。
	GetFallbackLanguage	データベースからフォールバック言語を読み取ります。
	GetHierarchicalCategories	データベースからすべての階層カテゴリを読み取ります。階層カテゴリは、センサデータベースコンテンツのツリー構造を作成するために使用します。
	GetLanguage	指定したISO言語名(2文字)の言語をデータベースから読み取ります。
	GetLanguages	データベースからすべての言語を読み取ります。
	GetSensor	これは、GetSensorメソッドの多重定義の概要です。
	GetSensors	これは、GetSensorsメソッドの多重定義の概要です。
	GetSensorsBase	データベースからすべてのセンサのリストとその ID、名前、説明を返します。すべてのセンサデータ(特定のセンサデータなど)を含むすべてのセンサのリストを取得する場合は、代わりにHbm.Api.SensorDB.ISensorDB.GetSensors()を使用します。
	GetThermoCoupleTypes	データベースからすべての熱電対タイプを読み取ります。
	GetTranslations	これは、GetTranslationsメソッドの多重定義の概要です。
	GetVersionInfo	データベースからセンサーデータベースのバージョン情報を読み取ります。
	HasChangedInDB	指定したセンサインスタンスが、データベース内の同じSensorIDを持つセンサと異なるかどうかをチェックします。`LastModifiedDate`プロパティが異なる場合、センサには変更されています。
	ImportCANDatabase	指定したファイル名からCANデータベースをインポートします。
	InsertCategory	指定したカテゴリをデータベースに挿入します。
	InsertDescriptor	これは、InsertDescriptorメソッドの多重定義の概要です。
	InsertLanguage	指定した言語をデータベースに挿入します。フォールバック言語のすべての翻訳がコピーされ、この新しい言語に挿入されます。

	名前	説明
	InsertSensor	指定したセンサをデータベースに挿入します。
	InsertThermoCoupleType	指定した熱電対タイプをデータベースに挿入します。既存する熱電対タイプは更新せず、削除してから新しいタイプを挿入する必要があります。
	SaveVersion	センサデータベースのバージョン情報をデータベースに保存します。
	SensorExist	指定したセンサIDのセンサがデータベース内に存在するかどうかを確認します。
	SetActiveLanguage	現在のアクティブな言語を設定します(データベースには保存されません)。
	SetFallbackLanguage	指定した言語をフォールバック言語として設定し、データベース内の他の言語からフラグを削除します。
	ShrinkDatabase	データベースページを再編成して空き領域を回復します。
	UpdateCategory	データベース内の指定したカテゴリを更新します。
	UpdateCategoryOfSensors	指定したセンサIDの親カテゴリを更新します。
	UpdateDescriptor	これは、UpdateDescriptorメソッドの多重定義の概要です。
	UpdateLanguage	指定した言語の説明を更新します。
	UpdateSensor	データベース内の既存のセンサを更新します。
	UpdateTranslation	データベース内の指定した翻訳を更新します。

Public Properties

	名前	説明
	ConnectionString	データベースに接続するための接続文字列。
	MaxAttachmentLength	添付ファイルの最大長を取得または設定します。
	ProviderInvariantName	データベースプロバイダのプロバイダインバリアント名。

ISensorDB Methods

ISensorDBクラスのメソッドの一覧です。

Public Methods

	名前	説明
	CategoryExist	指定したカテゴリIDによるカテゴリがデータベースに存在するかどうかをチェックします。
	ClearConnectionPool	基礎となる接続に関連付けられている接続プールを空にします。
	DeleteCategories	指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。
	DeleteCategory	指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。
	DeleteDescriptor	指定した記述子IDを持つ記述子(添付ファイルを含む)をデータベースから削除します。
	DeleteLanguage	指定したISO言語名(2文字)の言語をデータベースから削除します。
	DeleteSensor	指定したセンサIDを持つセンサをデータベースから削除します。
	DeleteSensors	指定したセンサIDを持つセンサをデータベースから削除します。
	DeleteThermoCoupleType	指定した熱電対タイプをデータベースから削除します。
	GetActiveLanguage	アクティブな言語(SetActiveLanguageによって設定)の言語インスタンスをデータベースから読み取ります。
	GetAttachment	指定した記述子IDに従って添付ファイルをデータベースから読み取ります。
	GetCategories	データベースからすべてのカテゴリを読み取ります。
	GetCategory	指定したカテゴリIDを持つカテゴリをデータベースから読み取ります。
	GetCategoryByInternalID	指定した内部IDを持つカテゴリをデータベースから読み取ります。
	GetDescriptor	指定した記述子IDを持つ記述子をデータベースから読み取ります。
	GetDescriptors	これは、GetDescriptorsメソッドの多重定義の概要です。
	GetFallbackLanguage	データベースからフォールバック言語を読み取ります。
	GetHierarchicalCategories	データベースからすべての階層カテゴリを読み取ります。階層カテゴリは、センサデータベースコンテンツのツリー構造を作成するために使用します。
	GetLanguage	指定したISO言語名(2文字)の言語をデータベースから読み取ります。
	GetLanguages	データベースからすべての言語を読み取ります。
	GetSensor	これは、GetSensorメソッドの多重定義の概要です。
	GetSensors	これは、GetSensorsメソッドの多重定義の概要です。
	GetSensorsBase	データベースからすべてのセンサのリストとその ID、名前、説明を返します。すべてのセンサデータ(特定のセンサデータなど)を含むすべてのセンサのリストを取得する場合は、代わりにHbm.Api.SensorDB.ISensorDB.GetSensors()を使用します。
	GetThermoCoupleTypes	データベースからすべての熱電対タイプを読み取ります。
	GetTranslations	これは、GetTranslationsメソッドの多重定義の概要です。
	GetVersionInfo	データベースからセンサーデータベースのバージョン情報を読み取ります。
	HasChangedInDB	指定したセンサインスタンスが、データベース内の同じSensorIDを持つセンサと異なるかどうかをチェックします。`LastModifiedDate`プロパティが異なる場合、センサには変更されています。
	ImportCANDatabase	指定したファイル名からCANデータベースをインポートします。
	InsertCategory	指定したカテゴリをデータベースに挿入します。
	InsertDescriptor	これは、InsertDescriptorメソッドの多重定義の概要です。
	InsertLanguage	指定した言語をデータベースに挿入します。フォールバック言語のすべての翻訳がコピーされ、この新しい言語に挿入されます。

	名前	説明
	InsertSensor	指定したセンサをデータベースに挿入します。
	InsertThermoCoupleType	指定した熱電対タイプをデータベースに挿入します。既存する熱電対タイプは更新せず、削除してから新しいタイプを挿入する必要があります。
	SaveVersion	センサデータベースのバージョン情報をデータベースに保存します。
	SensorExist	指定したセンサIDのセンサがデータベース内に存在するかどうかを確認します。
	SetActiveLanguage	現在のアクティブな言語を設定します(データベースには保存されません)。
	SetFallbackLanguage	指定した言語をフォールバック言語として設定し、データベース内の他の言語からフラグを削除します。
	ShrinkDatabase	データベースページを再編成して空き領域を回復します。
	UpdateCategory	データベース内の指定したカテゴリを更新します。
	UpdateCategoryOfSensors	指定したセンサIDの親カテゴリを更新します。
	UpdateDescriptor	これは、UpdateDescriptorメソッドの多重定義の概要です。
	UpdateLanguage	指定した言語の説明を更新します。
	UpdateSensor	データベース内の既存のセンサを更新します。
	UpdateTranslation	データベース内の指定した翻訳を更新します。

ISensorDB.CategoryExist Method

指定したカテゴリIDによるカテゴリがデータベースに存在するかどうかをチェックします。

C++

```
bool CategoryExist(
    int categoryId
);
```

C#

```
bool CategoryExist(
    int categoryId
);
```

Visual Basic

```
Function CategoryExist(
    categoryId As Integer
)As bool
```

引数

引数	説明
categoryId	チェックするカテゴリのID。

戻り値

カテゴリが存在する場合はtrue、存在しない場合はfalse。

ISensorDB.ClearConnectionPool Method

基礎となる接続に関連付けられている接続プールを空にします。

C++

```
void ClearConnectionPool();
```

C#

```
void ClearConnectionPool();
```

Visual Basic

```
Function ClearConnectionPool()As void
```

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.DeleteCategories Method

指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。

C++

```
void DeleteCategories(
    List<int> categoryIds
);
```

C#

```
void DeleteCategories(
    List<int> categoryIds
);
```

Visual Basic

```
Function DeleteCategories(
    categoryIds As List<int>
)As void
```

引数

引数	説明
categoryIds	削除するカテゴリのID。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException	データベースエラーが発生した場合にスローされます。たとえば、リストにない別のカテゴリによって参照されているカテゴリを削除しようとした場合などです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	その他のデータベースエラーが発生した場合にスローされます。

ISensorDB.DeleteCategory Method

指定したカテゴリIDを持つカテゴリをデータベースから削除します。これは、子カテゴリと子センサのないカテゴリの場合にのみ成功します。それ以外の場合は、Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException例外が発生します。

C++

```
void DeleteCategory(
    int categoryID
);
```

C#

```
void DeleteCategory(
    int categoryID
);
```

Visual Basic

```
Function DeleteCategory(
    categoryID As Integer
)As void
```

引数

引数	説明
categoryID	削除するカテゴリのID。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.CategoryHasChildsException	カテゴリに子カテゴリまたは子センサがある場合にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.DeleteDescriptor Method

指定した記述子IDを持つ記述子(添付ファイルを含む)をデータベースから削除します。

C++

```
void DeleteDescriptor(
    int descriptorID
);
```

C#

```
void DeleteDescriptor(
    int descriptorID
);
```

Visual Basic

```
Function DeleteDescriptor(
    descriptorID As Integer
)As void
```

引数

引数	説明
descriptorID	削除する記述子のID。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.DeleteLanguage Method

指定したISO言語名(2文字)の言語をデータベースから削除します。

C++

```
void DeleteLanguage(
    string twoLetterISOLanguageName
);
```

C#

```
void DeleteLanguage(
    string twoLetterISOLanguageName
);
```

Visual Basic

```
Function DeleteLanguage(
    twoLetterISOLanguageName As string
)As void
```

引数

引数	説明
twoLetterISOLanguageName	削除する言語のISO言語名(2文字)。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.FallbackLanguageDeleteException	フォールバック言語を削除しようとした場合にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.DeleteSensor Method

指定したセンサIDを持つセンサをデータベースから削除します。

C++

```
void DeleteSensor(
    int sensorID
);
```

C#

```
void DeleteSensor(
    int sensorID
);
```

Visual Basic

```
Function DeleteSensor(
    sensorID As Integer
)As void
```

引数

引数	説明
sensorID	削除するセンサのID。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.DeleteSensors Method

指定したセンサIDを持つセンサをデータベースから削除します。

C++

```
void DeleteSensors(
    List<int> sensorIds
);
```

C#

```
void DeleteSensors(
    List<int> sensorIds
);
```

Visual Basic

```
Function DeleteSensors(
    sensorIds As List<int>
)As void
```

引数

引数	説明
sensorIds	削除するセンサのID。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.DeleteThermoCoupleType Method

指定した熱電対タイプをデータベースから削除します。

C++

```
void DeleteThermoCoupleType(
    string thermoCoupleType
);
```

C#

```
void DeleteThermoCoupleType(
    string thermoCoupleType
);
```

Visual Basic

```
Function DeleteThermoCoupleType(
    thermoCoupleType As string
)As void
```

引数

引数	説明
thermoCoupleType	削除する熱電対タイプ。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetActiveLanguage Method

アクティブな言語(SetActiveLanguageによって設定)の言語インスタンスをデータベースから読み取ります。

C++

```
Language GetActiveLanguage();
```

C#

```
Language GetActiveLanguage();
```

Visual Basic

```
Function GetActiveLanguage()As Language
```

戻り値

Hbm.Api.SensorDB.Entities.Languageのインスタンス。見つからない場合はnull。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetAttachment Method

指定した記述子IDに従って添付ファイルをデータベースから読み取ります。

C++

```
byte[] GetAttachment(
    int descriptorID
);
```

C#

```
byte[] GetAttachment(
    int descriptorID
);
```

Visual Basic

```
Function GetAttachment(
    descriptorID As Integer
)As byte()
```

引数

引数	説明
descriptorID	記述子のID。

戻り値

添付ファイルは`byte[]`として返されます。見つからない場合は`null`になります。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetCategories Method

データベースからすべてのカテゴリを読み取ります。

C++

```
List<Category> GetCategories();
```

C#

```
List<Category> GetCategories();
```

Visual Basic

```
Function GetCategories()As List<Category>
```

戻り値

Hbm.Api.SensorDB.Entities.Categoryインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetCategory Method

指定したカテゴリIDを持つカテゴリをデータベースから読み取ります。

C++

```
Category GetCategory(
    int categoryID
);
```

C#

```
Category GetCategory(
    int categoryID
);
```

Visual Basic

```
Function GetCategory(
categoryID As Integer
)As Category
```

引数

引数	説明
categoryID	カテゴリのID。

戻り値

Hbm.Api.SensorDB.Entities.Categoryインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetCategoryByInternalID Method

指定した内部IDを持つカテゴリをデータベースから読み取ります。

C++

```
Category GetCategoryByInternalID(
    int internalID
);
```

C#

```
Category GetCategoryByInternalID(
    int internalID
);
```

Visual Basic

```
Function GetCategoryByInternalID(
internalID As Integer
)As Category
```

引数

引数	説明
internalID	カテゴリの内部ID。

戻り値

Hbm.Api.SensorDB.Entities.Categoryインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetDescriptor Method

指定した記述子IDを持つ記述子をデータベースから読み取ります。

C++

```
Descriptor GetDescriptor(
    int descriptorID
);
```

C#

```
Descriptor GetDescriptor(
    int descriptorID
);
```

Visual Basic

```
Function GetDescriptor(
descriptorID As Integer
)As Descriptor
```

引数

引数	説明
descriptorID	記述子のID。

戻り値

Hbm.Api.SensorDB.Entities.Descriptorインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

GetDescriptors Method

これは、GetDescriptorsメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ISensorDB.GetDescriptors ()	データベースからすべての記述子を読み取ります。
	ISensorDB.GetDescriptors (Sensor)	指定したセンサの記述子をデータベースから読み取ります。

ISensorDB.GetDescriptors Method ()

データベースからすべての記述子を読み取ります。

C++

```
List<Descriptor> GetDescriptors();
```

C#

```
List<Descriptor> GetDescriptors();
```

Visual Basic

```
Function GetDescriptors()As List<Descriptor>
```

戻り値

Hbm.Api.SensorDB.Entities.Descriptorインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetDescriptors Method (Sensor)

指定したセンサの記述子をデータベースから読み取ります。

C++

```
List<Descriptor> GetDescriptors(
    Sensor sensor
);
```

C#

```
List<Descriptor> GetDescriptors(
    Sensor sensor
);
```

Visual Basic

```
Function GetDescriptors(
    sensor As Sensor
)As List<Descriptor>
```

引数

引数	説明
sensor	記述子の検索対象となるHbm.Api.SensorDB.Entities.Sensorのインスタンス。

戻り値

Hbm.Api.SensorDB.Entities.Descriptorインスタンスのリスト。

例外

例外	説明
System.ArgumentNullException	sensorがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetFallbackLanguage Method

データベースからフォールバック言語を読み取ります。

C++

```
Language GetFallbackLanguage();
```

C#

```
Language GetFallbackLanguage();
```

Visual Basic

```
Function GetFallbackLanguage()As Language
```

戻り値

Hbm.Api.SensorDB.Entities.Languageのインスタンス。見つからない場合はnull。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetHierarchicalCategories Method

データベースからすべての階層カテゴリを読み取ります。階層カテゴリは、センサデータベースコンテンツのツリー構造を作成するために使用します。

C++

```
List<HierarchicalCategory> GetHierarchicalCategories();
```

C#

```
List<HierarchicalCategory> GetHierarchicalCategories();
```

Visual Basic

```
Function GetHierarchicalCategories()As List<HierarchicalCategory>
```

戻り値

Hbm.Api.SensorDB.Entities.HierarchicalCategoryインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetLanguage Method

指定したISO言語名(2文字)の言語をデータベースから読み取ります。

C++

```
Language GetLanguage(
    string twoLetterISOLanguageName
);
```

C#

```
Language GetLanguage(
    string twoLetterISOLanguageName
);
```

Visual Basic

```
Function GetLanguage(
    twoLetterISOLanguageName As string
)As Language
```

引数

引数	説明
twoLetterISOLanguageName	ISO言語名(2文字)。

戻り値

Hbm.Api.SensorDB.Entities.Languageインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetLanguages Method

データベースからすべての言語を読み取ります。

C++

```
List<Language> GetLanguages();
```

C#

```
List<Language> GetLanguages();
```

Visual Basic

```
Function GetLanguages()As List<Language>
```

戻り値

Hbm.Api.SensorDB.Entities.Languageインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

GetSensor Method

これは、GetSensorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ISensorDB.GetSensor (int)	指定したセンサIDを持つセンサをデータベースから読み取ります。
	ISensorDB.GetSensor (string)	指定した固有の名前を持つセンサをデータベースから読み取ります。

ISensorDB.GetSensor Method (int)

指定したセンサIDを持つセンサをデータベースから読み取ります。

C++

```
Sensor GetSensor(
    int sensorID
);
```

C#

```
Sensor GetSensor(
    int sensorID
);
```

Visual Basic

```
Function GetSensor(
    sensorID As Integer
)As Sensor
```

引数

引数	説明
sensorID	センサのID。

戻り値

Hbm.Api.SensorDB.Entities.Sensorインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetSensor Method (string)

指定した固有の名前を持つセンサをデータベースから読み取ります。

C++

```
Sensor GetSensor(
    string uniqueName
);
```

C#

```
Sensor GetSensor(
    string uniqueName
);
```

Visual Basic

```
Function GetSensor(
    uniqueName As string
)As Sensor
```

引数

引数	説明
uniqueName	センサ固有の名前。

戻り値

Hbm.Api.SensorDB.Entities.Sensorインスタンス。見つからない場合は`null`。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

GetSensors Method

これは、GetSensorsメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ISensorDB.GetSensors ()	データベースからすべてのセンサを読み取ります。
	ISensorDB.GetSensors (Category)	データベースから指定したカテゴリのすべてのセンサを読み取ります。

ISensorDB.GetSensors Method ()

データベースからすべてのセンサを読み取ります。

C++

```
List<Sensor> GetSensors();
```

C#

```
List<Sensor> GetSensors();
```

Visual Basic

```
Function GetSensors()As List<Sensor>
```

戻り値

Hbm.Api.SensorDB.Entities.Sensorインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetSensors Method (Category)

データベースから指定したカテゴリのすべてのセンサを読み取ります。

C++

```
List<Sensor> GetSensors(
    Category category
);
```

C#

```
List<Sensor> GetSensors(
    Category category
);
```

Visual Basic

```
Function GetSensors(
    category As Category
)As List<Sensor>
```

引数

引数	説明
category	センサ読み取りの対象カテゴリ。

戻り値

Hbm.Api.SensorDB.Entities.Sensorインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetSensorsBase Method

データベースからすべてのセンサのリストとその ID、名前、説明を返します。すべてのセンサデータ(特定のセンサデータなど)を含むすべてのセンサのリストを取得する場合は、代わりにHbm.Api.SensorDB.ISensorDB.GetSensors()を使用します。

C++

```
List<Sensor> GetSensorsBase();
```

C#

```
List<Sensor> GetSensorsBase();
```

Visual Basic

```
Function GetSensorsBase()As List<Sensor>
```

戻り値

Hbm.Api.SensorDB.Entities.Sensorインスタンスのリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetThermoCoupleTypes Method

データベースからすべての熱電対タイプを読み取ります。

C++

```
List<string> GetThermoCoupleTypes();
```

C#

```
List<string> GetThermoCoupleTypes();
```

Visual Basic

```
Function GetThermoCoupleTypes()As List<string>
```

戻り値

すべての熱電対タイプの文字列のリスト。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

GetTranslations Method

これは、GetTranslationsメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ISensorDB.GetTranslations (Category)	指定したカテゴリの翻訳をデータベースから読み取ります。
	ISensorDB.GetTranslations (Sensor)	指定したセンサの翻訳をデータベースから読み取ります。

ISensorDB.GetTranslations Method (Category)

指定したカテゴリの翻訳をデータベースから読み取ります。

C++

```
List<Translation> GetTranslations(
    Category category
);
```

C#

```
List<Translation> GetTranslations(
    Category category
);
```

Visual Basic

```
Function GetTranslations(
    category As Category
)As List<Translation>
```

引数

引数	説明
category	翻訳の検索対象となるHbm.Api.SensorDB.Entities.Categoryのインスタンス。

戻り値

カテゴリのHbm.Api.SensorDB.Entities.Translationインスタンスのリスト。

例外

例外	説明
System.ArgumentNullException	categoryがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetTranslations Method (Sensor)

指定したセンサの翻訳をデータベースから読み取ります。

C++

```
List<Translation> GetTranslations(
    Sensor sensor
);
```

C#

```
List<Translation> GetTranslations(
    Sensor sensor
);
```

Visual Basic

```
Function GetTranslations(
    sensor As Sensor
)As List<Translation>
```

引数

引数	説明
sensor	翻訳の検索対象となるHbm.Api.SensorDB.Entities.Sensorのインスタンス。

戻り値

センサのHbm.Api.SensorDB.Entities.Translationインスタンスのリスト。

例外

例外	説明
System.ArgumentNullException	sensorがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.GetVersionInfo Method

データベースからセンサーデータベースのバージョン情報を読み取ります。

C++

```
VersionInfo GetVersionInfo();
```

C#

```
VersionInfo GetVersionInfo();
```

Visual Basic

```
Function GetVersionInfo()As VersionInfo
```

戻り値

バージョン情報。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.HasChangedInDB Method

指定したセンサインスタンスが、データベース内の同じSensorIDを持つセンサと異なるかどうかをチェックします。
`LastModifiedDate`プロパティが異なる場合、センサには変更されています。

C++

```
bool HasChangedInDB(
    Sensor sensor
);
```

C#

```
bool HasChangedInDB(
    Sensor sensor
);
```

Visual Basic

```
Function HasChangedInDB(
    sensor As Sensor
)As bool
```

戻り値

センサに変更があった場合は`true`を返し、そうでない場合は`false`を返します。センサインスタンスが指定されていない場合は`false`を返します。センサインスタンスがデータベースに見つからなかった場合は`true`を返します(新しいセンサ)。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.ImportCANDatabase Method

指定したファイル名からCANデータベースをインポートします。

C++

```
void ImportCANDatabase(
    string filename,
    Category rootCategory,
    bool useCompoundNames
);
```

C#

```
void ImportCANDatabase(
    string filename,
    Category rootCategory,
    bool useCompoundNames
);
```

Visual Basic

```
Function ImportCANDatabase(
    filename As string,
    rootCategory As Category,
    useCompoundNames As bool
)As void
```

引数

引数	説明
filename	CANデータベースのパスとファイル名。
rootCategory	インポートしたデータのルートカテゴリ。
useCompoundNames	センサの説明に複合名を生成するかどうかのフラグ。

ISensorDB.InsertCategory Method

指定したカテゴリをデータベースに挿入します。

C++

```
void InsertCategory(
    Category category
);
```

C#

```
void InsertCategory(
    Category category
);
```

Visual Basic

```
Function InsertCategory(
    category As Category
)As void
```

引数

引数	説明
category	挿入するHbm.Api.SensorDB.Entities.Categoryのインスタンス。

例外

例外	説明
System.ArgumentNullException	categoryがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

InsertDescriptor Method

これは、InsertDescriptorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ISensorDB.InsertDescriptor (Descriptor)	指定した記述子をデータベースに挿入します。
	ISensorDB.InsertDescriptor (Descriptor, byte[])	指定した記述子をデータベースに挿入します。

ISensorDB.InsertDescriptor Method (Descriptor)

指定した記述子をデータベースに挿入します。

C++

```
void InsertDescriptor(
    Descriptor descriptor
);
```

C#

```
void InsertDescriptor(
    Descriptor descriptor
);
```

Visual Basic

```
Function InsertDescriptor(
    descriptor As Descriptor
)As void
```

引数

引数	説明
descriptor	挿入するHbm.Api.SensorDB.Entities.Descriptorのインスタンス。

例外

例外	説明
System.ArgumentNullException	記述子がnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.InsertDescriptor Method (Descriptor, byte[])

指定した記述子をデータベースに挿入します。

C++

```
void InsertDescriptor(
    Descriptor descriptor,
    byte[] attachment
);
```

C#

```
void InsertDescriptor(
    Descriptor descriptor,
    byte[] attachment
);
```

Visual Basic

```
Function InsertDescriptor(
    descriptor As Descriptor,
    attachment As byte[]
)As void
```

引数

引数	説明
descriptor	挿入するHbm.Api.SensorDB.Entities.Descriptorのインスタンス。
attachment	記述子とともに挿入するbyte[]としての添付ファイル。

例外

例外	説明
System.ArgumentNullException	記述子がnullです。 - または - 添付ファイルがnullです。
Hbm.Api.SensorDB.Exceptions.AttachmentToLargeException	添付ファイルが大きすぎる場合(> 1GB)にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.InsertLanguage Method

指定した言語をデータベースに挿入します。フォールバック言語のすべての翻訳がコピーされ、この新しい言語に挿入されます。

C++

```
void InsertLanguage(
    Language language
);
```

C#

```
void InsertLanguage(
    Language language
);
```

Visual Basic

```
Function InsertLanguage(
    language As Language
)As void
```

引数

引数	説明
descriptor	挿入するHbm.Api.SensorDB.Entities.Languageのインスタンス。

例外

例外	説明
System.ArgumentNullException	languageが`null`です。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.InsertSensor Method

指定したセンサをデータベースに挿入します。

C++

```
void InsertSensor(
    Sensor sensor
);
```

C#

```
void InsertSensor(
    Sensor sensor
);
```

Visual Basic

```
Function InsertSensor(
    sensor As Sensor
)As void
```

引数

引数	説明
sensor	挿入するHbm.Api.SensorDB.Entities.Sensorのインスタンス。

例外

例外	説明
System.ArgumentNullException	sensorがnullです。
Hbm.Api.SensorDB.Exceptions.CategoryNotAssignedException	センサにカテゴリインスタンスが割り当てられていない場合にスローされます。
Hbm.Api.SensorDB.Exceptions.ScalingNotAssignedException	センサにスケーリングインスタンスが割り当てられていない場合にスローされます。
Hbm.Api.SensorDB.Exceptions.ScalingNotSupportedException	割り当てられたスケーリングの型がサポートされていないHbm.Api.SensorDB.Entities.Scalings.NotSupportedScalingの場合にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorNotSupportedException	センサの型がHbm.Api.SensorDB.Entities.Sensors.NotSupportedSensorである場合にスローされますが、これはサポートされていません。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.InsertThermoCoupleType Method

指定した熱電対タイプをデータベースに挿入します。既存する熱電対タイプは更新せず、削除してから新しいタイプを挿入する必要があります。

C++

```
void InsertThermoCoupleType(
    string thermoCoupleType
);
```

C#

```
void InsertThermoCoupleType(
    string thermoCoupleType
);
```

Visual Basic

```
Function InsertThermoCoupleType(
    thermoCoupleType As string
)As void
```

引数

引数	説明
thermoCoupleType	挿入する新しい熱電対タイプ。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.SaveVersion Method

センサデータベースのバージョン情報をデータベースに保存します。

C++

```
void SaveVersion(
    VersionInfo versionInfo
);
```

C#

```
void SaveVersion(
    VersionInfo versionInfo
);
```

Visual Basic

```
Function SaveVersion(
    versionInfo As VersionInfo
)As void
```

引数

引数	説明
versionInfo	保存するバージョン情報。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.SensorExist Method

指定したセンサIDのセンサがデータベース内に存在するかどうかを確認します。

C++

```
bool SensorExist(
    int sensorId
);
```

C#

```
bool SensorExist(
    int sensorId
);
```

Visual Basic

```
Function SensorExist(
    sensorId As Integer
)As bool
```

引数

引数	説明
sensorId	チェックするセンサのID。

戻り値

センサが存在する場合はtrue、存在しない場合はfalse。

ISensorDB.SetActiveLanguage Method

現在のアクティブな言語を設定します(データベースには保存されません)。

C++

```
bool SetActiveLanguage(
    string twoLetterISOLanguageName
);
```

C#

```
bool SetActiveLanguage(
    string twoLetterISOLanguageName
);
```

Visual Basic

```
Function SetActiveLanguage(
    twoLetterISOLanguageName As string
)As bool
```

引数

引数	説明
twoLetterISOLanguageName	現在の言語のISO言語名(2文字)。

戻り値

現在の言語が存在し、設定されている場合は`true`、そうでない場合は`false`。現在の言語が存在しない(したがって`false`を取得)場合は、フォールバック言語がアクティブ言語として設定されます。

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.SetFallbackLanguage Method

指定した言語をフォールバック言語として設定し、データベース内の他の言語からフラグを削除します。

C++

```
void SetFallbackLanguage(
    Language fallbackLanguage
);
```

C#

```
void SetFallbackLanguage(
    Language fallbackLanguage
);
```

Visual Basic

```
Function SetFallbackLanguage(
    fallbackLanguage As Language
)As void
```

引数

引数	説明
fallbackLanguage	フォールバック言語として設定されるHbm.Api.SensorDB.Entities.Languageのインスタンス。

例外

例外	説明
System.ArgumentNullException	fallbackLanguageが`null`です。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.ShrinkDatabase Method

データベースページを再編成して空き領域を回復します。

C++

```
void ShrinkDatabase();
```

C#

```
void ShrinkDatabase();
```

Visual Basic

```
Function ShrinkDatabase()As void
```

例外

例外	説明
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.UpdateCategory Method

データベース内の指定したカテゴリを更新します。

C++

```
void UpdateCategory(
    Category category
);
```

C#

```
void UpdateCategory(
    Category category
);
```

Visual Basic

```
Function UpdateCategory(
    category As Category
)As void
```

引数

引数	説明
category	更新するHbm.Api.SensorDB.Entities.Categoryのインスタンス。

例外

例外	説明
System.ArgumentNullException	categoryが`null`です。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.UpdateCategoryOfSensors Method

指定したセンサIDの親カテゴリを更新します。

C++

```
void UpdateCategoryOfSensors(
List<Sensor> sensors
);
```

C#

```
void UpdateCategoryOfSensors(
List<Sensor> sensors
);
```

Visual Basic

```
Function UpdateCategoryOfSensors(
    sensors As List<Sensor>
)As void
```

引数

引数	説明
sensors	カテゴリ更新を適用するセンサのリスト。

注意

データベースにはカテゴリプロパティのみが適用されます。センサ全体を更新する場合は、UpdateSensorを参照してください。

例外

例外	説明
System.ArgumentNullException	sensorが`null`です。
Hbm.Api.SensorDB.Exceptions.CategoryNotAssignedException	センサにカテゴリインスタンスが割り当てられていない場合にスローされます。
Hbm.Api.SensorDB.Exceptions.ScalingNotAssignedException	センサにスケーリングインスタンスが割り当てられていない場合にスローされます。
Hbm.Api.SensorDB.Exceptions.ScalingNotSupportedException	割り当てられたスケーリングの型がサポートされていないHbm.Api.SensorDB.Entities.Scalings.NotSupportedScalingの場合にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorNotSupportedException	センサの型がHbm.Api.SensorDB.Entities.Sensors.NotSupportedSensorである場合にスローされますが、これはサポートされていません。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

UpdateDescriptor Method

これは、UpdateDescriptorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ISensorDB.UpdateDescriptor (Descriptor)	データベース内の指定した記述子を更新します。
	ISensorDB.UpdateDescriptor (Descriptor, byte[])	データベース内の指定した記述子を更新します。

ISensorDB.UpdateDescriptor Method (Descriptor)

データベース内の指定した記述子を更新します。

C++

```
void UpdateDescriptor(
    Descriptor descriptor
);
```

C#

```
void UpdateDescriptor(
    Descriptor descriptor
);
```

Visual Basic

```
Function UpdateDescriptor(
    descriptor As Descriptor
)As void
```

引数

引数	説明
descriptor	更新するHbm.Api.SensorDB.Entities.Descriptorのインスタンス。

例外

例外	説明
System.ArgumentNullException	descriptorが`null`です。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.UpdateDescriptor Method (Descriptor, byte[])

データベース内の指定した記述子を更新します。

C++

```
void UpdateDescriptor(
    Descriptor descriptor,
    byte[] attachment
);
```

C#

```
void UpdateDescriptor(
    Descriptor descriptor,
    byte[] attachment
);
```

Visual Basic

```
Function UpdateDescriptor(
    descriptor As Descriptor,
    attachment As byte[]
)As void
```

引数

引数	説明
descriptor	更新するHbm.Api.SensorDB.Entities.Descriptorのインスタンス。
attachment	記述子で更新されるbyte[]としての添付ファイル。

例外

例外	説明
System.ArgumentNullException	記述子がnullです。 - または - 添付ファイルがnullです。
Hbm.Api.SensorDB.Exceptions.AttachmentToLargeException	添付ファイルが大きすぎる場合(> 1GB)にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.UpdateLanguage Method

指定した言語の説明を更新します。

C++

```
void UpdateLanguage(
    Language language
);
```

C#

```
void UpdateLanguage(
    Language language
);
```

Visual Basic

```
Function UpdateLanguage(
    language As Language
)As void
```

引数

引数	説明
language	更新するHbm.Api.SensorDB.Entities.Languageのインスタンス。

例外

例外	説明
System.ArgumentNullException	languageがnullです。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.UpdateSensor Method

データベース内の既存のセンサを更新します。

C++

```
void UpdateSensor(
    Sensor sensor
);
```

C#

```
void UpdateSensor(
    Sensor sensor
);
```

Visual Basic

```
Function UpdateSensor(
    sensor As Sensor
)As void
```

引数

引数	説明
sensor	更新するHbm.Api.SensorDB.Entities.Sensorのインスタンス。

例外

例外	説明
System.ArgumentNullException	sensorがnullです。
Hbm.Api.SensorDB.Exceptions.CategoryNotAssignedException	センサにカテゴリインスタンスが割り当てられていない場合にスローされます。
Hbm.Api.SensorDB.Exceptions.ScalingNotAssignedException	センサにスケーリングインスタンスが割り当てられていない場合にスローされます。
Hbm.Api.SensorDB.Exceptions.ScalingNotSupportedException	割り当てられたスケーリングの型がサポートされていないHbm.Api.SensorDB.Entities.Scalings.NotSupportedScalingの場合にスローされます。
Hbm.Api.SensorDB.Exceptions.SensorNotSupportedException	センサの型がHbm.Api.SensorDB.Entities.Sensors.NotSupportedSensorである場合にスローされますが、これはサポートされていません。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB.UpdateTranslation Method

データベース内の指定した翻訳を更新します。

C++

```
void UpdateTranslation(
    Translation translation
);
```

C#

```
void UpdateTranslation(
    Translation translation
);
```

Visual Basic

```
Function UpdateTranslation(
    translation As Translation
)As void
```

引数

引数	説明
translation	更新するHbm.Api.SensorDB.Entities.Translationのインスタンス。

備考

このメソッドは翻訳を更新するだけです。翻訳の挿入は、新しいセンサまたは新しいカテゴリを挿入することによって間接的に行われます。したがって、新しい翻訳を明示的に挿入するメソッドは必要ありません。

例外

例外	説明
System.ArgumentNullException	translationが`null`です。
Hbm.Api.SensorDB.Exceptions.SensorDBDataException	データベースエラーが発生した場合にスローされます。

ISensorDB Properties

ISensorDBクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ConnectionString	データベースに接続するための接続文字列。
	MaxAttachmentLength	添付ファイルの最大長を取得または設定します。
	ProviderInvariantName	データベースプロバイダのプロバイダインバリアント名。

ISensorDB.ConnectionString Property

データベースに接続するための接続文字列。

C++

```
__property string ConnectionString;
```

C#

```
string ConnectionString;
```

Visual Basic

```
Property ConnectionString() As string
```

ISensorDB.MaxAttachmentLength Property

添付ファイルの最大長を取得または設定します。

C++

```
__property int MaxAttachmentLength;
```

C#

```
int MaxAttachmentLength;
```

Visual Basic

```
Property MaxAttachmentLength() As Integer
```

説明

添付ファイルの最大長。

ISensorDB.ProviderInvariantName Property

データベースプロバイダのプロバイダインバリアント名。

C++

```
__property string ProviderInvariantName;
```

C#

```
string ProviderInvariantName;
```

Visual Basic

```
Property ProviderInvariantName() As string
```

Hbm.Api.SensorDB.Entities Namespace

これは名前空間Hbm.Api.SensorDB.Entitiesです。

クラス

	名前	説明
	Category	センサのカテゴリを表します。
	Descriptor	バイナリドキュメントなど、センサの記述子を表します。
	HierarchicalCategory	カプセル化されたカテゴリの階層ビューを表します。カプセル化されたカテゴリを、階層的な子カテゴリのリストと子センサのリストに拡張します。
	Language	翻訳に使用する言語をカプセル化します。
A	Scaling	センサ値のスケーリングを表します。
A	Sensor	温度、ブリッジ、電圧などの物理センサを表します。
	Translation	翻訳済みエンティティをカプセル化します。
	VersionInfo	センサデータベースのバージョン情報をカプセル化します。

名前空間

名前	説明
Hbm.Api.SensorDB.Entities.Scalings	これは名前空間Hbm.Api.SensorDB.Entities.Scalingsです。
Hbm.Api.SensorDB.Entities.Sensors	これは名前空間Hbm.Api.SensorDB.Entities.Sensorsです。

Hbm.Api.SensorDB.Entities.Scalings Namespace

これは名前空間Hbm.Api.SensorDB.Entities.Scalingsです。

クラス

	名前	説明
	CanBusScaling	CAN busのスケーリングを表します。
	GageFactorScaling	ゲージ係数のスケーリングを表します。
	InternalScaling	センサの内部スケーリングを表します。
	NotSupportedScaling	サポートされていないスケーリングを表します。
	OffScaling	無効なスケーリング(オフ)を表します。
A	PointScalingBase	ポイントスケーリングの基本クラス。
	PolynomialScaling	多項式スケーリングを表します。
	PolynomialSegment	1つの多項式セグメントを表します。
	PolynomialTerm	1つの多項式項を表します。
	ScalingPoint	単一のスケーリング ポイントを表します。
	TableScaling	テーブルスケーリングを表します。
	TwoPointScaling	2点スケーリングを表します。
	ZeroSpanScaling	ゼロスパンスケーリングを表します。

インタフェース

	名前	説明
	IScalingCalculation	電気値と工学値間の計算を使用するスケーリングのインタフェース。

CanBusScaling Class

CAN busのスケーリングを表します。



C++

```
[Serializable]
class CanBusScaling : public Scaling;
```

C#

```
[Serializable]
public class CanBusScaling: Scaling;
```

Visual Basic

```
<Serializable>
Public Class CanBusScaling
Inherits Scaling
```

ファイル

CanBusScaling.cs

CanBusScaling Members

次の表は、CanBusScalingによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。

Public Methods

	名前	説明
	CanBusScaling	これは、CanBusScalingコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	Factor	係数を取得または設定します。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
	Offset	オフセットを取得または設定します。
R	ScalingType	スケーリングタイプ(読み取り専用)。

CanBusScaling Constructor

これは、CanBusScalingコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	CanBusScaling.CanBusScaling ()	新しいインスタンスを作成します。
	CanBusScaling.CanBusScaling (decimal, decimal, string, decimal, decimal, bool)	CanBusScalingクラスの新しいインスタンスを初期化します。

CanBusScaling.CanBusScaling Constructor ()

新しいインスタンスを作成します。

C++

```
CanBusScaling();
```

C#

```
public CanBusScaling();
```

Visual Basic

```
Public Sub New()
```

CanBusScaling.CanBusScaling Constructor (decimal, decimal, string, decimal, decimal, bool)

CanBusScalingクラスの新しいインスタンスを初期化します。

C++

```
CanBusScaling(
    decimal offset,
    decimal factor,
    string engineeringUnit,
    decimal minEngineeringRange,
    decimal maxEngineeringRange,
    bool isEngineeringRangeRms
);
```

C#

```
public CanBusScaling(
    decimal offset,
    decimal factor,
    string engineeringUnit,
    decimal minEngineeringRange,
    decimal maxEngineeringRange,
    bool isEngineeringRangeRms
);
```

Visual Basic

```
Public Sub New(
    offset As decimal,
    factor As decimal,
    engineeringUnit As string,
    minEngineeringRange As decimal,
    maxEngineeringRange As decimal,
    isEngineeringRangeRms As bool
)
```

引数

引数	説明
offset	オフセット。
factor	係数。
engineeringUnit	工学単位。
minEngineeringRange	最小工学範囲。
maxEngineeringRange	最大工学範囲。
isEngineeringRangeRms	trueに設定している場合[エンジニアリング範囲はRMS]。

CanBusScaling Properties

CanBusScalingクラスのプロパティの一覧です。

Private Properties

	名前	説明
	Factor	係数を取得または設定します。
	Offset	オフセットを取得または設定します。
R	ScalingType	スケーリングタイプ(読み取り専用)。

CanBusScaling.Factor Property

係数を取得または設定します。

C++

```
__property decimal Factor;
```

C#

```
public decimal Factor;
```

Visual Basic

```
Public Property Factor() As decimal
```

説明

スケーリング係数。

CanBusScaling.Offset Property

オフセットを取得または設定します。

C++

```
__property decimal Offset;
```

C#

```
public decimal Offset;
```

Visual Basic

```
Public Property Offset() As decimal
```

説明

スケーリングオフセット。

CanBusScaling.ScalingType Property

スケーリングタイプ(読み取り専用)。

C++

```
__property ScalingType ScalingType;
```

C#

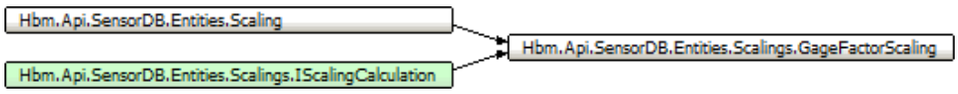
```
public override ScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As ScalingType
```

GageFactorScaling Class

ゲージ係数のスケーリングを表します。



C++

```
[Serializable]
class GageFactorScaling : public Scaling, public IScalingCalculation;
```

C#

```
[Serializable]
public class GageFactorScaling: Scaling, IScalingCalculation;
```

Visual Basic

```
<Serializable>
Public Class GageFactorScaling
Inherits Scaling
Inherits IScalingCalculation
```

ファイル

GageFactorScaling.cs

GageFactorScaling Members

次の表は、GageFactorScalingによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。

Public Methods

	名前	説明
	CalculateElectricalValue	現在のスケーリングのブリッジ係数とゲージ係数(k係数)を使用して、指定した工学値から電気値を計算します。
	CalculateEngineeringValue	現在のスケーリングのブリッジ係数とゲージ係数(k係数)を使用して、指定した電気値から工学値を計算します。

Public Properties

	名前	説明
	BridgeFactor	ブリッジ係数。
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	GageFactor	ゲージ係数(K係数)。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
R	MaxElectricalRange	計算した最大電気値(読み取り専用)。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
R	ScalingType	スケーリングタイプ(読み取り専用)。

GageFactorScaling Methods

次の表は、GageFactorScalingクラスによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CalculateElectricalValue	現在のスケーリングのブリッジ係数とゲージ係数(k係数)を使用して、指定した工学値から電気値を計算します。
	CalculateEngineeringValue	現在のスケーリングのブリッジ係数とゲージ係数(k係数)を使用して、指定した電気値から工学値を計算します。

GageFactorScaling.CalculateElectricalValue Method

現在のスケーリングのブリッジ係数とゲージ係数(k係数)を使用して、指定した工学値から電気値を計算します。

C++

```
decimal CalculateElectricalValue(
    decimal engineeringValue
);
```

C#

```
public decimal CalculateElectricalValue(
    decimal engineeringValue
);
```

Visual Basic

```
Public Function CalculateElectricalValue(
    engineeringValue As decimal
)As decimal
```

引数

引数	説明
engineeringValue	工学値。

戻り値

計算した電気値。

GageFactorScaling.CalculateEngineeringValue Method

現在のスケーリングのブリッジ係数とゲージ係数(k係数)を使用して、指定した電気値から工学値を計算します。

C++

```
decimal CalculateEngineeringValue(
    decimal electricalValue
);
```

C#

```
public decimal CalculateEngineeringValue(
    decimal electricalValue
);
```

Visual Basic

```
Public Function CalculateEngineeringValue(
    electricalValue As decimal
)As decimal
```

引数

引数	説明
electricalValue	電気値。

戻り値

計算した工学値。

GageFactorScaling Properties

GageFactorScalingクラスのプロパティの一覧です。

Public Properties

	名前	説明
	BridgeFactor	ブリッジ係数。
	GageFactor	ゲージ係数(K係数)。
R	MaxElectricalRange	計算した最大電気値(読み取り専用)。
R	ScalingType	スケーリングタイプ(読み取り専用)。

GageFactorScaling.BridgeFactor Property

ブリッジ係数。

C++

```
__property decimal BridgeFactor;
```

C#

```
public decimal BridgeFactor;
```

Visual Basic

```
Public Property BridgeFactor() As decimal
```

GageFactorScaling.GageFactor Property

ゲージ係数(K係数)。

C++

```
__property decimal GageFactor;
```

C#

```
public decimal GageFactor;
```

Visual Basic

```
Public Property GageFactor() As decimal
```

GageFactorScaling.MaxElectricalRange Property

計算した最大電気値(読み取り専用)。

C++

```
__property decimal MaxElectricalRange;
```

C#

```
public decimal MaxElectricalRange;
```

Visual Basic

```
Public Property MaxElectricalRange() As decimal
```

GageFactorScaling.ScalingType Property

スケーリングタイプ(読み取り専用)。

C++

```
__property ScalingType ScalingType;
```

C#

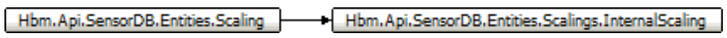
```
public override ScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As ScalingType
```

InternalScaling Class

センサの内部スケーリングを表します。



C++

```
[Serializable]
class InternalScaling : public Scaling;
```

C#

```
[Serializable]
public class InternalScaling: Scaling;
```

Visual Basic

```
<Serializable>
Public Class InternalScaling
Inherits Scaling
```

ファイル

InternalScaling.cs

InternalScaling Members

次の表は、InternalScalingによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。

Public Properties

	名前	説明
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
R	ScalingType	スケーリングタイプ(読み取り専用)。

InternalScaling Properties

InternalScalingクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ScalingType	スケーリングタイプ(読み取り専用)。

InternalScaling.ScalingType Property

スケーリングタイプ(読み取り専用)。

C++

```
__property ScalingType ScalingType;
```

C#

```
public override ScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As ScalingType
```

NotSupportedScaling Class

サポートされていないスケーリングを表します。



C++

```
[Serializable]
class NotSupportedScaling : public Scaling;
```

C#

```
[Serializable]
public class NotSupportedScaling: Scaling;
```

Visual Basic

```
<Serializable>
Public Class NotSupportedScaling
Inherits Scaling
```

ファイル

NotSupportedScaling.cs

NotSupportedScaling Members

次の表は、NotSupportedScalingによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。

Public Properties

	名前	説明
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
R	ScalingType	スケーリングタイプ(読み取り専用)。

NotSupportedScaling Properties

NotSupportedScalingクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ScalingType	スケーリングタイプ(読み取り専用)。

NotSupportedScaling.ScalingType Property

スケーリングタイプ(読み取り専用)。

C++

```
__property ScalingType ScalingType;
```

C#

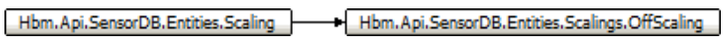
```
public override ScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As ScalingType
```

OffScaling Class

無効なスケーリング(オフ)を表します。



C++

```
[Serializable]
class OffScaling : public Scaling;
```

C#

```
[Serializable]
public class OffScaling : Scaling;
```

Visual Basic

```
<Serializable>
Public Class OffScaling
Inherits Scaling
```

ファイル

OffScaling.cs

OffScaling Members

次の表は、OffScalingによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。

Public Properties

	名前	説明
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
R	ScalingType	スケーリングタイプ(読み取り専用)。

OffScaling Properties

OffScalingクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ScalingType	スケーリングタイプ(読み取り専用)。

OffScaling.ScalingType Property

スケーリングタイプ(読み取り専用)。

C++

```
__property ScalingType ScalingType;
```

C#

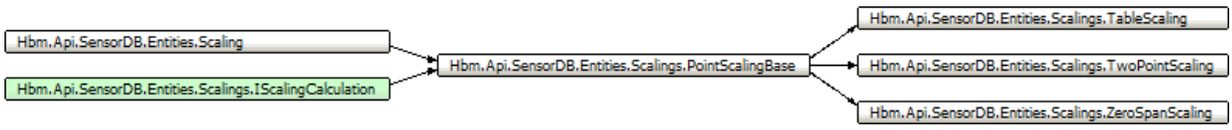
```
public override ScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As ScalingType
```

PointScalingBase Class

ポイントスケーリングの基本クラス。



C++

```
[Serializable]
class PointScalingBase : public Scaling, public IScalingCalculation;
```

C#

```
[Serializable]
public abstract class PointScalingBase: Scaling, IScalingCalculation;
```

Visual Basic

```
<Serializable>
Public abstract Class PointScalingBase
Inherits Scaling
Inherits IScalingCalculation
```

ファイル

PointScalingBase.cs

PointScalingBase Members

次の表は、PointScalingBaseによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	SetScalingPoints	内部スケーリングポイントリストを設定します。派生クラスの内部プロパティを初期化して、そのプロパティをスケーリングポイントにマッピングするには、このメソッドをオーバーライドします(例: 2点スケーリング)。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。
	ScalingPoints	スケーリングポイントのリスト(内部使用のみ、例: データアクセスレイヤ)。

Protected Methods

	名前	説明
	CalculateXFromLinearEquation	2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたy値に対するx値を計算します。
	CalculateYFromLinearEquation	2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたx値に対するy値を計算します。
	PrepareTwoPointList	正確に2つの点を含むリストが格納されるようにスケーリングポイントのリストを準備します。

Public Methods

	名前	説明
	CalculateElectricalValue	現在のスケーリングのスケーリングポイントを使用して、指定した工学値の電気値を計算します。
	CalculateEngineeringValue	現在のスケーリングのスケーリングポイントを使用して、指定した電気値の工学値を計算します。
	PointScalingBase	新しいインスタンスを作成し、スケーリングポイントのリストを初期化します。

Public Properties

	名前	説明
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
	MaxElectricalRange	計算した最大電気値(読み取り専用)。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
	ScalingType	スケーリングタイプ(読み取り専用)。

PointScalingBase.PointScalingBase Constructor

新しいインスタンスを作成し、スケーリングポイントのリストを初期化します。

C++

```
PointScalingBase();
```

C#

```
public PointScalingBase();
```

Visual Basic

```
Public Sub New
```

PointScalingBase Methods

次の表は、PointScalingBaseクラスによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	SetScalingPoints	内部スケーリングポイントリストを設定します。派生クラスの内部プロパティを初期化して、そのプロパティをスケーリングポイントにマッピングするには、このメソッドをオーバーライドします(例: 2点スケーリング)。

Protected Methods

	名前	説明
	CalculateXFromLinearEquation	2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたy値に対するx値を計算します。
	CalculateYFromLinearEquation	2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたx値に対するy値を計算します。
	PrepareTwoPointList	正確に2つの点を含むリストが格納されるようにスケーリングポイントのリストを準備します。

Public Methods

	名前	説明
	CalculateElectricalValue	現在のスケーリングのスケーリングポイントを使用して、指定した工学値の電気値を計算します。
	CalculateEngineeringValue	現在のスケーリングのスケーリングポイントを使用して、指定した電気値の工学値を計算します。

PointScalingBase.CalculateElectricalValue Method

現在のスケーリングのスケーリングポイントを使用して、指定した工学値の電気値を計算します。

C++

```
decimal CalculateElectricalValue(
    decimal engineeringValue
);
```

C#

```
public virtual decimal CalculateElectricalValue(
    decimal engineeringValue
);
```

Visual Basic

```
Public virtual Function CalculateElectricalValue(
    engineeringValue As decimal
)As decimal
```

引数

引数	説明
engineeringValue	工学値。

戻り値

計算した電気値。

備考

このメソッドは、ゼロ スパンスケーリングなどによって上書き可能である必要があるため重要です。

PointScalingBase.CalculateEngineeringValue Method

現在のスケーリングのスケーリングポイントを使用して、指定した電気値の工学値を計算します。

C++

```
decimal CalculateEngineeringValue(
    decimal electricalValue
);
```

C#

```
public virtual decimal CalculateEngineeringValue(
    decimal electricalValue
);
```

Visual Basic

```
Public virtual Function CalculateEngineeringValue(
    electricalValue As decimal
)As decimal
```

引数

引数	説明
electricalValue	電気値。

戻り値

計算した工学値。

備考

このメソッドは、ゼロ スパンスケーリングなどによって上書き可能である必要があるため重要です。

PointScalingBase.CalculateXFromLinearEquation Method

2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたy値に対するx値を計算します。

C++

```
decimal CalculateXFromLinearEquation(
    decimal y,
    decimal x1,
    decimal y1,
    decimal x2,
    decimal y2
);
```

C#

```
protected decimal CalculateXFromLinearEquation(
    decimal y,
    decimal x1,
    decimal y1,
    decimal x2,
    decimal y2
);
```

Visual Basic

```
Protected Function CalculateXFromLinearEquation(
    y As decimal,
    x1 As decimal,
    y1 As decimal,
    x2 As decimal,
    y2 As decimal
) As decimal
```

引数

引数	説明
y	対応するx値を計算するy値。
x1	最初のポイントのx値。
y1	最初のポイントのy値。
x2	2番目のポイントのx値。
y2	2番目のポイントのy値。

戻り値

線上の指定されたy値に対するx値、またはy2 = y1の場合はINVALID_VALUE(これは許可されていません)。

現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace -> Hbm.Api.SensorDB.Entities.Scalings Namespace -> PointScalingBase Class -> PointScalingBase.PointScalingBase Constructor

PointScalingBase.CalculateYFromLinearEquation Method

2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたx値に対するy値を計算します。

C++

```
decimal CalculateYFromLinearEquation(
    decimal x,
    decimal x1,
    decimal y1,
    decimal x2,
    decimal y2
);
```

C#

```
protected decimal CalculateYFromLinearEquation(
    decimal x,
    decimal x1,
    decimal y1,
    decimal x2,
    decimal y2
);
```

Visual Basic

```
Protected Function CalculateYFromLinearEquation(
    x As decimal,
    x1 As decimal,
    y1 As decimal,
    x2 As decimal,
    y2 As decimal
) As decimal
```

引数

引数	説明
x	対応するy値を計算するx値。
x1	最初のポイントのx値。
y1	最初のポイントのy値。
x2	2番目のポイントのx値。
y2	2番目のポイントのy値。

戻り値

線上の指定されたx値に対するy値、またはx2 = x1の場合はINVALID_VALUE(これは許可されていません)。

PointScalingBase.PrepareTwoPointList Method

正確に2つの点を含むリストが格納されるようにスケーリングポイントのリストを準備します。

C++

```
void PrepareTwoPointList();
```

C#

```
protected void PrepareTwoPointList();
```

Visual Basic

```
Protected Function PrepareTwoPointList()As void
```

PointScalingBase.SetScalingPoints Method

内部スケーリングポイントリストを設定します。派生クラスの内部プロパティを初期化して、そのプロパティをスケーリングポイントにマッピングするには、このメソッドをオーバーライドします(例: 2点スケーリング)。

C++

```
void SetScalingPoints(
    List<ScalingPoint> scalingPoints
);
```

C#

```
internal virtual void SetScalingPoints(
    List<ScalingPoint> scalingPoints
);
```

Visual Basic

```
Friend virtual Function SetScalingPoints(
    scalingPoints As List<ScalingPoint>
)As void
```

引数

引数	説明
scalingPoints	ScalingPointのリスト。

PointScalingBase Properties

PointScalingBaseクラスのプロパティの一覧です。

Internal Properties

	名前	説明
	ScalingPoints	スケーリングポイントのリスト(内部使用のみ、例: データアクセスレイヤ)。

Public Properties

	名前	説明
	MaxElectricalRange	計算した最大電気値(読み取り専用)。
	MaxEngineeringRange	最大の物理値。

PointScalingBase.MaxElectricalRange Property

計算した最大電気値(読み取り専用)。

C++

```
__property decimal MaxElectricalRange;
```

C#

```
public virtual decimal MaxElectricalRange;
```

Visual Basic

```
Public virtual Property MaxElectricalRange() As decimal
```

備考

このメソッドは、ゼロスパンスケーリングなどによって上書き可能である必要があるため重要です。

PointScalingBase.ScalingPoints Property

スケーリングポイントのリスト(内部使用のみ、例: データアクセスレイヤ)。

C++

```
__property List<ScalingPoint> ScalingPoints;
```

C#

```
internal List<ScalingPoint> ScalingPoints;
```

Visual Basic

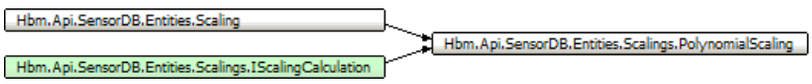
```
Friend Property ScalingPoints()As List<ScalingPoint>
```

備考

セッタは読み取り専用です。仮想メソッドSetScalingPointsを使用して、リストを割り当て、派生クラスの必要なプロパティを初期化します。

PolynomialScaling Class

多項式スケーリングを表します。



C++

```
[Serializable]
class PolynomialScaling : public Scaling, public IScalingCalculation;
```

C#

```
[Serializable]
public class PolynomialScaling: Scaling, IScalingCalculation;
```

Visual Basic

```
<Serializable>
Public Class PolynomialScaling
Inherits Scaling
Inherits IScalingCalculation
```

ファイル

PolynomialScaling.cs

PolynomialScaling Members

次の表は、PolynomialScalingによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。

Public Methods

	名前	説明
	CalculateEngineeringValue	現在のスケーリングの多項式を使用して、指定した電気値の工学値を計算します。
	PolynomialScaling	新しいインスタンスを作成し、多項式セグメントのリストを初期化します。

Public Properties

	名前	説明
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
	MaxElectricalRange	多項式スケーリングの最大電気値範囲。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
	PolynomialSegments	この多項式スケーリングの多項式セグメントのリスト。
	ScalingType	スケーリングタイプ(読み取り専用)。

PolynomialScaling.PolynomialScaling Constructor

新しいインスタンスを作成し、多項式セグメントのリストを初期化します。

C++

```
PolynomialScaling();
```

C#

```
public PolynomialScaling();
```

Visual Basic

```
Public Sub New()
```

PolynomialScaling Methods

次の表は、PolynomialScalingクラスによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CalculateEngineeringValue	現在のスケーリングの多項式を使用して、指定した電気値の工学値を計算します。

PolynomialScaling.CalculateEngineeringValue Method

現在のスケーリングの多項式を使用して、指定した電気値の工学値を計算します。

C++

```
decimal CalculateEngineeringValue(
    decimal electricalValue
);
```

C#

```
public decimal CalculateEngineeringValue(
    decimal electricalValue
);
```

Visual Basic

```
Public Function CalculateEngineeringValue(
    electricalValue As decimal
)As decimal
```

引数

引数	説明
electricalValue	電気値。

戻り値

計算した工学値。

PolynomialScaling Properties

PolynomialScalingクラスのプロパティの一覧です。

Public Properties

	名前	説明
	MaxElectricalRange	多項式スケーリングの最大電気値範囲。
	PolynomialSegments	この多項式スケーリングの多項式セグメントのリスト。
R	ScalingType	スケーリングタイプ(読み取り専用)。

PolynomialScaling.MaxElectricalRange Property

多項式スケーリングの最大電気値範囲。

C++

```
__property decimal MaxElectricalRange;
```

C#

```
public decimal MaxElectricalRange;
```

Visual Basic

```
Public Property MaxElectricalRange() As decimal
```

PolynomialScaling.PolynomialSegments Property

この多項式スケーリングの多項式セグメントのリスト。

C++

```
__property List<PolynomialSegment> PolynomialSegments;
```

C#

```
public List<PolynomialSegment> PolynomialSegments;
```

Visual Basic

```
Public Property PolynomialSegments()As List<PolynomialSegment>
```

PolynomialScaling.ScalingType Property

スケーリングタイプ(読み取り専用)。

C++

```
__property ScalingType ScalingType;
```

C#

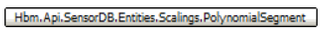
```
public override ScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As ScalingType
```

PolynomialSegment Class

1つの多項式セグメントを表します。



C++

```
[Serializable]
class PolynomialSegment;
```

C#

```
[Serializable]
public class PolynomialSegment;
```

Visual Basic

```
<Serializable>
Public Class PolynomialSegment
```

ファイル

PolynomialSegment.cs

PolynomialSegment Members

次の表は、PolynomialSegmentによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	PolynomialSegment	これは、PolynomialSegmentコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	PolynomialTerms	多項式項のリスト。
	StartX	セグメントの開始X値。

PolynomialSegment Constructor

これは、PolynomialSegmentコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	PolynomialSegment. PolynomialSegment ()	新しいインスタンスを作成し、多項式項のリストを初期化します。
	PolynomialSegment. PolynomialSegment (double)	新しいインスタンスを作成し、値を初期化します。

PolynomialSegment.PolynomialSegment Constructor ()

新しいインスタンスを作成し、多項式項のリストを初期化します。

C++

```
PolynomialSegment();
```

C#

```
public PolynomialSegment();
```

Visual Basic

```
Public Sub New()
```

PolynomialSegment.PolynomialSegment Constructor (double)

新しいインスタンスを作成し、値を初期化します。

C++

```
PolynomialSegment(
    double startX
);
```

C#

```
public PolynomialSegment(
    double startX
);
```

Visual Basic

```
Public Sub New(
    startX As double
)
```

引数

引数	説明
startX	セグメントの開始X値。

PolynomialSegment Properties

PolynomialSegmentクラスのプロパティの一覧です。

Public Properties

	名前	説明
	PolynomialTerms	多項式項のリスト。
	StartX	セグメントの開始X値。

PolynomialSegment.PolynomialTerms Property

多項式項のリスト。

C++

```
__property List<PolynomialTerm> PolynomialTerms;
```

C#

```
public List<PolynomialTerm> PolynomialTerms;
```

Visual Basic

```
Public Property PolynomialTerms()As List<PolynomialTerm>
```

PolynomialSegment.StartX Property

セグメントの開始X値。

C++

```
__property double StartX;
```

C#

```
public double StartX;
```

Visual Basic

```
Public Property StartX()As double
```

PolynomialTerm Class

1つの多項式項を表します。



C++

```
[Serializable]
class PolynomialTerm;
```

C#

```
[Serializable]
public class PolynomialTerm;
```

Visual Basic

```
<Serializable>
Public Class PolynomialTerm
```

ファイル

PolynomialTerm.cs

PolynomialTerm Members

次の表は、PolynomialTermによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	PolynomialTerm	これは、PolynomialTermコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	Coefficient	多項式項の係数。
	Exponent	多項式項の指数。

PolynomialTerm Constructor

これは、PolynomialTermコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	PolynomialTerm.PolynomialTerm ()	新しいインスタンスを作成します。
	PolynomialTerm.PolynomialTerm (decimal, int)	新しいインスタンスを作成し、値を初期化します。

PolynomialTerm.PolynomialTerm Constructor ()

新しいインスタンスを作成します。

C++

```
PolynomialTerm();
```

C#

```
public PolynomialTerm();
```

Visual Basic

```
Public Sub New()
```

PolynomialTerm.PolynomialTerm Constructor (decimal, int)

新しいインスタンスを作成し、値を初期化します。

C++

```
PolynomialTerm(
    decimal coefficient,
    int exponent
);
```

C#

```
public PolynomialTerm(
    decimal coefficient,
    int exponent
);
```

Visual Basic

```
Public Sub New(
    coefficient As decimal,
    exponent As Integer
)
```

引数

引数	説明
coefficient	係数の値。
exponent	指数の値。

PolynomialTerm Properties

PolynomialTermクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Coefficient	多項式項の係数。
	Exponent	多項式項の指数。

PolynomialTerm.Coefficient Property

多項式項の係数。

C++

```
__property decimal Coefficient;
```

C#

```
public decimal Coefficient;
```

Visual Basic

```
Public Property Coefficient()As decimal
```

PolynomialTerm.Exponent Property

多項式項の指数。

C++

```
__property int Exponent;
```

C#

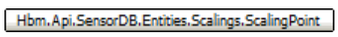
```
public int Exponent;
```

Visual Basic

```
Public Property Exponent()As Integer
```

ScalingPoint Class

単一のスケーリング ポイントを表します。



C++

```
[Serializable]
class ScalingPoint;
```

C#

```
[Serializable]
public class ScalingPoint;
```

Visual Basic

```
<Serializable>
Public Class ScalingPoint
```

ファイル

ScalingPoint.cs

ScalingPoint Members

次の表は、ScalingPointによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ScalingPoint	これは、ScalingPointコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	X	ポイントのX値。
	Y	ポイントのY値。

ScalingPoint Constructor

これは、ScalingPointコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	ScalingPoint.ScalingPoint ()	新しいインスタンスを作成します。
	ScalingPoint.ScalingPoint (decimal, decimal)	新しいインスタンスを作成し、値を初期化します。

ScalingPoint.ScalingPoint Constructor ()

新しいインスタンスを作成します。

C++

```
ScalingPoint();
```

C#

```
public ScalingPoint();
```

Visual Basic

```
Public Sub New()
```

ScalingPoint.ScalingPoint Constructor (decimal, decimal)

新しいインスタンスを作成し、値を初期化します。

C++

```
ScalingPoint(
    decimal x,
    decimal y
);
```

C#

```
public ScalingPoint(
    decimal x,
    decimal y
);
```

Visual Basic

```
Public Sub New(
     x As decimal,
     y As decimal
)
```

引数

引数	説明
x	x値。
y	y値。

ScalingPoint Properties

ScalingPointクラスのプロパティの一覧です。

Public Properties

	名前	説明
	X	ポイントのX値。
	Y	ポイントのY値。

ScalingPoint.X Property

ポイントのX値。

C++

```
__property decimal X;
```

C#

```
public decimal X;
```

Visual Basic

```
Public Property X() As decimal
```

ScalingPoint.Y Property

ポイントのY値。

C++

```
__property decimal Y;
```

C#

```
public decimal Y;
```

Visual Basic

```
Public Property Y() As decimal
```

TableScaling Class

テーブルスケーリングを表します。



C++

```
[Serializable]
class TableScaling : public PointScalingBase;
```

C#

```
[Serializable]
public class TableScaling : PointScalingBase;
```

Visual Basic

```
<Serializable>
Public Class TableScaling
Inherits PointScalingBase
```

ファイル

TableScaling.cs

TableScaling Members

次の表は、TableScalingによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	SetScalingPoints	内部スケーリングポイントリストを設定します。派生クラスの内部プロパティを初期化して、そのプロパティをスケーリングポイントにマッピングするには、このメソッドをオーバーライドします(例: 2点スケーリング)。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。

Protected Methods

	名前	説明
	CalculateXFromLinearEquation	2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたy値に対するx値を計算します。
	CalculateYFromLinearEquation	2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたx値に対するy値を計算します。
	PrepareTwoPointList	正確に2つの点を含むリストが格納されるようにスケーリングポイントのリストを準備します。

Public Methods

	名前	説明
	CalculateElectricalValue	現在のスケーリングのスケーリングポイントを使用して、指定した工学値の電気値を計算します。
	CalculateEngineeringValue	現在のスケーリングデータを使用して、指定した電気値の工学値を計算します。
	CalculateEngineeringValue	現在のスケーリングのスケーリングポイントを使用して、指定した電気値の工学値を計算します。

Public Properties

	名前	説明
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
	MaxElectricalRange	計算した最大電気値(読み取り専用)。
	MaxElectricalRange	計算した最大電気値(読み取り専用)。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
	ScalingPoints	スケーリングテーブルを表すスケーリングポイントのリスト。
	ScalingType	スケーリングタイプ(読み取り専用)。

TableScaling Properties

TableScalingクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ScalingPoints	スケーリングテーブルを表すスケーリングポイントのリスト。
R	ScalingType	スケーリングタイプ(読み取り専用)。

TableScaling.ScalingPoints Property

スケーリングテーブルを表すスケーリングポイントのリスト。

C++

```
__property List<ScalingPoint> ScalingPoints;
```

C#

```
public new List<ScalingPoint> ScalingPoints;
```

Visual Basic

```
Public new Property ScalingPoints()As List<ScalingPoint>
```

TableScaling.ScalingType Property

スケーリングタイプ(読み取り専用)。

C++

```
__property ScalingType ScalingType;
```

C#

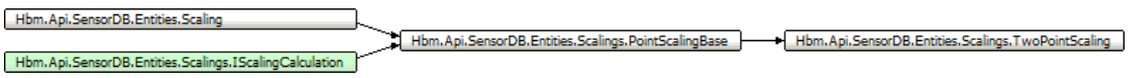
```
public override ScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As ScalingType
```

TwoPointScaling Class

2点スケーリングを表します。



C++

```
[Serializable]
class TwoPointScaling : public PointScalingBase;
```

C#

```
[Serializable]
public class TwoPointScaling : PointScalingBase;
```

Visual Basic

```
<Serializable>
Public Class TwoPointScaling
Inherits PointScalingBase
```

ファイル

TwoPointScaling.cs

TwoPointScaling Members

次の表は、TwoPointScalingによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	SetScalingPoints	内部スケーリングポイントリストを設定します。派生クラスの内部プロパティを初期化して、そのプロパティをスケーリングポイントにマッピングするには、このメソッドをオーバーライドします(例: 2点スケーリング)。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。
	ScalingPoints	内部スケーリングポイントリストを設定します。派生クラスの内部プロパティを初期化して、そのプロパティをスケーリングポイントにマッピングするには、このメソッドをオーバーライドします(例: 2点スケーリング)。

Protected Methods

	名前	説明
	CalculateXFromLinearEquation	2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたy値に対するx値を計算します。
	CalculateYFromLinearEquation	2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたx値に対するy値を計算します。
	PrepareTwoPointList	正確に2つの点を含むリストが格納されるようにスケーリングポイントのリストを準備します。

Public Methods

	名前	説明
	CalculateElectricalValue	現在のスケーリングのスケーリングポイントを使用して、指定した工学値の電気値を計算します。
	CalculateEngineeringValue	現在のスケーリングデータを使用して、指定した電気値の工学値を計算します。
	CalculateEngineeringValue	現在のスケーリングのスケーリングポイントを使用して、指定した電気値の工学値を計算します。
	TwoPointScaling	これは、TwoPointScalingコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	ElectricalP1	電気値P1。
	ElectricalP2	電気値P2。
	EngineeringP1	工学値P1。
	EngineeringP2	工学値P2。
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
	MaxElectricalRange	計算した最大電気値(読み取り専用)。
R V	MaxElectricalRange	計算した最大電気値(読み取り専用)。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
R	ScalingType	スケーリングタイプ(読み取り専用)。

TwoPointScaling Constructor

これは、TwoPointScalingコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	TwoPointScaling.TwoPointScaling ()	新しいインスタンスを作成します。
	TwoPointScaling.TwoPointScaling (decimal, decimal, string, decimal, decimal, bool)	新しいインスタンスを作成し、データを初期化します。
	TwoPointScaling.TwoPointScaling (decimal, decimal, string, decimal, decimal, decimal, decimal, bool)	新しいインスタンスを作成し、データを初期化します。

TwoPointScaling.TwoPointScaling Constructor ()

新しいインスタンスを作成します。

C++

```
TwoPointScaling();
```

C#

```
public TwoPointScaling();
```

Visual Basic

```
Public Sub New()
```

TwoPointScaling.TwoPointScaling Constructor (decimal, decimal, string, decimal, decimal, bool)

新しいインスタンスを作成し、データを初期化します。

C++

```
TwoPointScaling(
    decimal electricalP1,
    decimal electricalP2,
    string engineeringUnit,
    decimal engineeringP1,
    decimal engineeringP2,
    bool isEngineeringRangeRms
);
```

C#

```
public TwoPointScaling(
    decimal electricalP1,
    decimal electricalP2,
    string engineeringUnit,
    decimal engineeringP1,
    decimal engineeringP2,
    bool isEngineeringRangeRms
);
```

Visual Basic

```
Public Sub New(
    electricalP1 As decimal,
    electricalP2 As decimal,
    engineeringUnit As string,
    engineeringP1 As decimal,
    engineeringP2 As decimal,
    isEngineeringRangeRms As bool
)
```

TwoPointScaling.TwoPointScaling Constructor (decimal, decimal, string, decimal, decimal, decimal, decimal, bool)

新しいインスタンスを作成し、データを初期化します。

C++

```
TwoPointScaling(
    decimal electricalP1,
    decimal electricalP2,
    string engineeringUnit,
    decimal engineeringP1,
    decimal engineeringP2,
    decimal minEngineeringRange,
    decimal maxEngineeringRange,
    bool isEngineeringRangeRms
);
```

C#

```
public TwoPointScaling(
    decimal electricalP1,
    decimal electricalP2,
    string engineeringUnit,
    decimal engineeringP1,
    decimal engineeringP2,
    decimal minEngineeringRange,
    decimal maxEngineeringRange,
    bool isEngineeringRangeRms
);
```

Visual Basic

```
Public Sub New(
    electricalP1 As decimal,
    electricalP2 As decimal,
    engineeringUnit As string,
    engineeringP1 As decimal,
    engineeringP2 As decimal,
    minEngineeringRange As decimal,
    maxEngineeringRange As decimal,
    isEngineeringRangeRms As bool
)
```

TwoPointScaling Properties

TwoPointScalingクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ElectricalP1	電気値P1。
	ElectricalP2	電気値P2。
	EngineeringP1	工学値P1。
	EngineeringP2	工学値P2。
	EngineeringUnit	物理単位(例: "kN"や"mm")。
R	ScalingType	スケーリングタイプ(読み取り専用)。

TwoPointScaling.ElectricalP1 Property

電気値P1。

C++

```
__property decimal ElectricalP1;
```

C#

```
public decimal ElectricalP1;
```

Visual Basic

```
Public Property ElectricalP1() As decimal
```

備考

スケーリングポイントのリスト内のスケーリングPoint[0].Xに関連付けられます。

TwoPointScaling.ElectricalP2 Property

電気値P2。

C++

```
__property decimal ElectricalP1;
```

C#

```
public decimal ElectricalP1;
```

Visual Basic

```
Public Property ElectricalP1() As decimal
```

備考

スケーリングポイントのリスト内のスケーリングPoint[1].Xに関連付けられます。

TwoPointScaling.EngineeringP1 Property

工学値P1。

C++

```
__property decimal EngineeringP1;
```

C#

```
public decimal EngineeringP1;
```

Visual Basic

```
Public Property EngineeringP1() As decimal
```

備考

スケーリングポイントのリスト内のスケーリングPoint[0].Yに関連付けられます。

TwoPointScaling.EngineeringP2 Property

工学値P2。

C++

```
__property decimal EngineeringP2;
```

C#

```
public decimal EngineeringP2;
```

Visual Basic

```
Public Property EngineeringP2() As decimal
```

備考

スケーリングポイントのリスト内のスケーリングPoint[1].Yに関連付けられます。

TwoPointScaling.ScalingType Property

スケーリングタイプ(読み取り専用)。

C++

```
__property ScalingType ScalingType;
```

C#

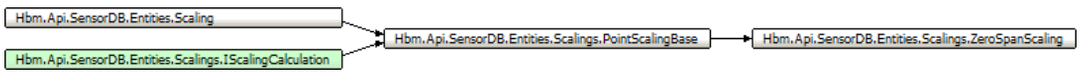
```
public override ScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As ScalingType
```

ZeroSpanScaling Class

ゼロスパンスケーリングを表します。



C++

```
[Serializable]
class ZeroSpanScaling : public PointScalingBase;
```

C#

```
[Serializable]
public class ZeroSpanScaling : PointScalingBase;
```

Visual Basic

```
<Serializable>
Public Class ZeroSpanScaling
Inherits PointScalingBase
```

ファイル

ZeroSpanScaling.cs

備考

スケーリングポイントのリストを内部でカプセル化します。

ZeroSpanScaling Members

次の表は、ZeroSpanScalingによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	SetScalingPoints	内部スケーリングポイントリストを設定します。派生クラスの内部プロパティを初期化して、そのプロパティをスケーリングポイントにマッピングするには、このメソッドをオーバーライドします(例: 2点スケーリング)。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。
	ScalingPoints	内部スケーリングポイントリストを設定します。派生クラスの内部プロパティを初期化して、そのプロパティをスケーリングポイントにマッピングするには、このメソッドをオーバーライドします(例: 2点スケーリング)。

Protected Methods

	名前	説明
	CalculateXFromLinearEquation	2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたy値に対するx値を計算します。
	CalculateYFromLinearEquation	2点形式を使用して、2つの点(x1,y1)と(x2,y2)で定義される線上の指定されたx値に対するy値を計算します。
	PrepareTwoPointList	正確に2つの点を含むリストが格納されるようにスケーリングポイントのリストを準備します。

Public Methods

	名前	説明
	CalculateElectricalValue	現在のスケーリングのスケーリングポイントを使用して、指定した工学値の電気値を計算します。
	CalculateEngineeringValue	現在のスケーリングデータを使用して、指定した電気値の工学値を計算します。
	CalculateEngineeringValue	現在のスケーリングのスケーリングポイントを使用して、指定した電気値の工学値を計算します。

Public Properties

	名前	説明
	ElectricalSensitivity	電気的感度。
	ElectricalZero	電気的ゼロ。
	EngineeringNominal	工学的公称値。
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
R	MaxElectricalRange	計算した最大電気値(読み取り専用)。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
R	ScalingType	スケーリングタイプ(読み取り専用)。

ZeroSpanScaling Properties

ZeroSpanScalingクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ElectricalSensitivity	電気的感度。
	ElectricalZero	電気的ゼロ。
	EngineeringNominal	工学的公称値。
R	MaxElectricalRange	計算した最大電気値(読み取り専用)。
R	ScalingType	スケーリングタイプ(読み取り専用)。

ZeroSpanScaling.ElectricalSensitivity Property

電気的感度。

C++

```
__property decimal ElectricalSensitivity;
```

C#

```
public decimal ElectricalSensitivity;
```

Visual Basic

```
Public Property ElectricalSensitivity() As decimal
```

ZeroSpanScaling.ElectricalZero Property

電気的ゼロ。

C++

```
__property decimal ElectricalZero;
```

C#

```
public decimal ElectricalZero;
```

Visual Basic

```
Public Property ElectricalZero() As decimal
```

ZeroSpanScaling.EngineeringNominal Property

工学的公称値。

C++

```
__property decimal EngineeringNominal;
```

C#

```
public decimal EngineeringNominal;
```

Visual Basic

```
Public Property EngineeringNominal() As decimal
```

ZeroSpanScaling.MaxElectricalRange Property

計算した最大電気値(読み取り専用)。

C++

```
__property decimal MaxElectricalRange;
```

C#

```
public override decimal MaxElectricalRange;
```

Visual Basic

```
Public override Property MaxElectricalRange() As decimal
```

備考

このメソッドは、ZeroSpanスケーリングのScalingPoint値の一部がポイントではなくスパンを定義するため、ZeroSpanスケーリングなどによってオーバーライドする必要があるため、仮想的です。

ZeroSpanScaling.ScalingType Property

スケーリングタイプ(読み取り専用)。

C++

```
__property ScalingType ScalingType;
```

C#

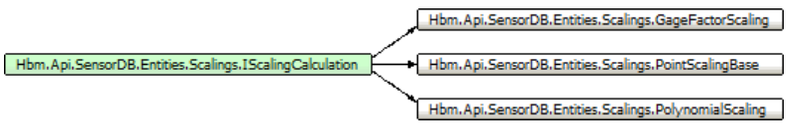
```
public override ScalingType ScalingType;
```

Visual Basic

```
Public override Property ScalingType() As ScalingType
```

IScalingCalculation Interface

電気値と工学値間の計算を使用するスケーリングのインタフェース。



C++

```
__interface IScalingCalculation;
```

C#

```
public interface IScalingCalculation;
```

Visual Basic

```
Public Interface IScalingCalculation
```

ファイル

IScalingCalculation.cs

IScalingCalculation Members

次の表は、ZeroSpanScalingによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CalculateEngineeringValue	現在のスケーリングデータを使用して、指定した電気値の工学値を計算します。

Public Properties

	名前	説明
	MaxElectricalRange	計算した最大電気値(読み取り専用)。

IScalingCalculation Methods

IScalingCalculationクラスのメソッドの一覧です。

Public Methods

	名前	説明
	CalculateEngineeringValue	現在のスケーリングデータを使用して、指定した電気値の工学値を計算します。

IScalingCalculation.CalculateEngineeringValue Method

現在のスケーリングデータを使用して、指定した電気値の工学値を計算します。

C++

```
decimal CalculateEngineeringValue(
    decimal electricalValue
);
```

C#

```
decimal CalculateEngineeringValue(
    decimal electricalValue
);
```

Visual Basic

```
Function CalculateEngineeringValue(
    electricalValue As decimal
)As decimal
```

引数

引数	説明
electricalValue	電気値。

戻り値

計算した工学値。

IScalingCalculation Properties

IScalingCalculationクラスのプロパティの一覧です。

Public Properties

	名前	説明
	MaxElectricalRange	計算した最大電気値(読み取り専用)。

IScalingCalculation.MaxElectricalRange Property

計算した最大電気値(読み取り専用)。

C++

```
__property decimal MaxElectricalRange;
```

C#

```
decimal MaxElectricalRange;
```

Visual Basic

```
Property MaxElectricalRange() As decimal
```

Hbm.Api.SensorDB.Entities.Sensors Namespace

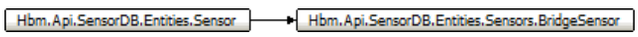
これは名前空間Hbm.Api.SensorDB.Entities.Sensorsです。

クラス

	名前	説明
	BridgeSensor	ブリッジセンサを表します。
	CanBusSensor	CAN(Controller Area Network)バスを介して接続されたセンサを表します。
	CounterSensor	カウンタセンサを表します。
	CurrentProbeSensor	電流プローブセンサを表します。
	CurrentSensor	電流センサを表します。
	FbgAccelerometerSensor	加速度計ブラッググレーティングセンサを表します。
	FbgGenericSensor	汎用ファイバブラッグレーティングセンサを表します。
A	FbgSensor	ファイバブラッググレーティングセンサを表します。
	FbgStrainGageSensor	ファイバブラッググレーティングひずみゲージセンサを表します。
	FbgTemperatureSensor	ファイバブラッググレーティング温度センサを表します。
	FbgWavelengthSensor	ファイバブラッググレーティング波長センサを表します。
	FrequencySensor	周波数センサを表します。
	IepeSensor	IEPE(Integrated Electronics Piezo-Electric)センサを表します。
	LvdtSensor	LVDT(差動変圧器)センサを表します。
	NotSupportedSensor	サポートされていないセンサを表します。
	PiezoPassiveSensor	パッシブピエゾセンサを表します。
	PotentiometerSensor	ポテンショメータセンサを表します。
	PtSensor	プラチナ温度センサを表します。
	PwmSensor	パルス幅変調センサを表します。
	ResistanceSensor	抵抗センサを表します。
	SsiSensor	SSIセンサを表します。
	ThermoCoupleSensor	熱電対センサを表します。
	VoltageProbeSensor	電圧プローブセンサを表します。
	VoltageSensor	電圧センサを表します。

BridgeSensor Class

ブリッジセンサを表します。



C++

```
[Serializable]
class BridgeSensor : public Sensor;
```

C#

```
[Serializable]
public class BridgeSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class BridgeSensor
Inherits Sensor
```

ファイル

BridgeSensor.cs

BridgeSensor Members

次の表は、BridgeSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	BridgeSensor	新しいインスタンスを作成し、データを初期化します。
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。

Public Properties

	名前	説明
	BridgeType	ブリッジセンサの種類。
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	CarrierFrequencyType	キャリア周波数の種類。このプロパティがHbm.Api.SensorDB.Enums.CarrierFrequencyType.Valueと等しい場合、CarrierFrequencyValueに必要な値が含まれます。
	CarrierFrequencyValue	キャリア信号周波数(Hz)。この値は、プロパティCarrierFrequencyTypeがHbm.Api.SensorDB.Enums.CarrierFrequencyType.Valueと等しい場合にのみ使用してください。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Impedance	インピーダンス(オーム単位)。
	Info	センサの追加情報(またはデータ)。
	IsInductive	センサを誘導型としてマークします。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	MaxExcitationVoltage	最大許容励起。
	MinExcitationVoltage	最小許容励起。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	PreferredExcitationVoltage	推奨励起電圧。値はMinExcitationVoltageとMaxExcitationVoltageの間でなければなりません。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。

	名前	説明
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。
	Wiring	物理的なワイヤの数。

BridgeSensor.BridgeSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
BridgeSensor();
```

C#

```
public BridgeSensor();
```

Visual Basic

```
Public Sub New()
```

BridgeSensor Properties

BridgeSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	BridgeType	ブリッジセンサの種類。
	CarrierFrequencyType	キャリア周波数の種類。このプロパティがHbm.Api.SensorDB.Enums.CarrierFrequencyType.Valueと等しい場合、CarrierFrequencyValueに必要な値が含まれます。
	CarrierFrequencyValue	キャリア信号周波数(Hz)。この値は、プロパティCarrierFrequencyTypeがHbm.Api.SensorDB.Enums.CarrierFrequencyType.Valueと等しい場合にのみ使用してください。
	Impedance	インピーダンス(オーム単位)。
	IsInductive	センサを誘導型としてマークします。
	MaxExcitationVoltage	最大許容励起。
	MinExcitationVoltage	最小許容励起。
	PreferredExcitationVoltage	推奨励起電圧。値はMinExcitationVoltageとMaxExcitationVoltageの間でなければなりません。
R	SensorType	センサの種類(読み取り専用)。
	Wiring	物理的なワイヤの数。

BridgeSensor.BridgeType Property

ブリッジセンサの種類。

C++

```
__property BridgeType BridgeType;
```

C#

```
private BridgeType BridgeType;
```

Visual Basic

```
Private Property BridgeType() As BridgeType
```

BridgeSensor.CarrierFrequencyType Property

キャリア周波数の種類。このプロパティがHbm.Api.SensorDB.Enums.CarrierFrequencyType.Valueと等しい場合、CarrierFrequencyValueに必要な値が含まれます。

C++

```
__property CarrierFrequencyType CarrierFrequencyType;
```

C#

```
private CarrierFrequencyType CarrierFrequencyType;
```

Visual Basic

```
Private Property CarrierFrequencyType() As CarrierFrequencyType
```

BridgeSensor.CarrierFrequencyValue Property

キャリア信号周波数(Hz)。この値は、プロパティCarrierFrequencyTypeがHbm.Api.SensorDB.Enums.CarrierFrequencyType.Valueと等しい場合にのみ使用してください。

C++

```
__property decimal CarrierFrequencyValue;
```

C#

```
public decimal CarrierFrequencyValue;
```

Visual Basic

```
Public Property CarrierFrequencyValue() As decimal
```

BridgeSensor.Impedance Property

インピーダンス(オーム単位)。

C++

```
__property decimal Impedance;
```

C#

```
public decimal Impedance;
```

Visual Basic

```
Public Property Impedance() As decimal
```

備考

非誘導ブリッジセンサにのみ使用されます。

BridgeSensor.IsInductive Property

センサを誘導型としてマークします。

C++

```
__property bool IsInductive;
```

C#

```
public bool IsInductive;
```

Visual Basic

```
Public Property IsInductive() As bool
```

BridgeSensor.MaxExcitationVoltage Property

最大許容励起。

C++

```
__property decimal MaxExcitationVoltage;
```

C#

```
public decimal MaxExcitationVoltage;
```

Visual Basic

```
Public Property MaxExcitationVoltage() As decimal
```

BridgeSensor.MinExcitationVoltage Property

最小許容励起。

C++

```
__property decimal MinExcitationVoltage;
```

C#

```
public decimal MinExcitationVoltage;
```

Visual Basic

```
Public Property MinExcitationVoltage() As decimal
```

BridgeSensor.PreferredExcitationVoltage Property

推奨励起電圧。値はMinExcitationVoltageとMaxExcitationVoltageの間でなければなりません。

C++

```
__property decimal PreferredExcitationVoltage;
```

C#

```
public decimal PreferredExcitationVoltage;
```

Visual Basic

```
Public Property PreferredExcitationVoltage() As decimal
```

BridgeSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

BridgeSensor.Wiring Property

物理的なワイヤの数。

C++

```
__property BridgeSensorWiring Wiring;
```

C#

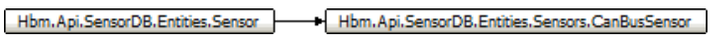
```
public BridgeSensorWiring Wiring;
```

Visual Basic

```
Public Property Wiring() As BridgeSensorWiring
```

CanBusSensor Class

CAN(Controller Area Network)バスを介して接続されたセンサを表します。



C++

```
[Serializable]
class CanBusSensor : public Sensor;
```

C#

```
[Serializable]
public class CanBusSensor: Sensor;
```

Visual Basic

```
<Serializable>
Public Class CanBusSensor
Inherits Sensor
```

ファイル

CanBusSensor.cs

CanBusSensor Members

次の表は、CanBusSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CanBusSensor	新しいインスタンスを作成し、データを初期化します。
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。

Public Properties

	名前	説明
	ActiveQuery	メッセージの内容。
	ActiveQueryRate	クエリが送信される頻度を評価します。
	ActiveSupplyVoltage	有効な供給電圧の値。このプロパティは情報提供のみを目的としています。
	BitSequence	ビッグエンディアンエンコーディング(Motorola)を使用しているかどうかを示します。それ以外の場合は、リトルエンディアンエンコーディング(Intel)を使用します。
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsActiveQueryEnabled	センサを活性化してデータを送信します。
	IsExtendedFrameFormat	拡張フレーム形式(29ビット識別子)を使用しているかどうかを示します。それ以外の場合は、標準フレーム形式(11ビット識別子)を使用します。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	MessageID	センサのCAN busメッセージの識別子。
	ModelName	センサのモデル名。
	ModeSignalBitCount	モード信号のビット数。
	ModeSignalBitSequence	モード信号にビッグエンディアンエンコーディング(Motorola)を使用しているかどうかを示します。それ以外の場合は、リトルエンディアンエンコーディング(Intel)を使用します。
	ModeSignalModeValue	モード信号のモード値。
	ModeSignalRawValueFormat	モード信号の生値の形式。
	ModeSignalStartBit	CAN busメッセージ内のモード信号開始ビットの位置。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PayloadByteCount	メッセージのデータ部のサイズ。

	名前	説明
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	RawValueFormat	生値の形式。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	SignalLength	信号のビット数。
	SignalType	CAN信号の種類。
	StartBit	CAN busメッセージ内のスタートビットの位置。
	StorageLocation	センサを設置している場所。
	Timeout	2つの信号間の待機時間(ミリ秒単位)。`Timeout = 0`の場合、タイムアウトは使用しません。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

CanBusSensor.CanBusSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
CanBusSensor();
```

C#

```
public CanBusSensor();
```

Visual Basic

```
Public Sub New()
```

CanBusSensor Properties

CanBusSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ActiveQuery	メッセージの内容。
	ActiveQueryRate	クエリが送信される頻度を評価します。
	ActiveSupplyVoltage	有効な供給電圧の値。このプロパティは情報提供のみを目的としています。
	BitSequence	ビッグエンディアンエンコーディング(Motorola)を使用しているかどうかを示します。それ以外の場合は、リトルエンディアンエンコーディング(Intel)を使用します。
	IsActiveQueryEnabled	センサを活性化してデータを送信します。
	IsExtendedFrameFormat	拡張フレーム形式(29ビット識別子)を使用しているかどうかを示します。それ以外の場合は、標準フレーム形式(11ビット識別子)を使用します。
	MessageID	センサのCAN busメッセージの識別子。
	ModeSignalBitCount	モード信号のビット数。
	ModeSignalBitSequence	モード信号にビッグエンディアンエンコーディング(Motorola)を使用しているかどうかを示します。それ以外の場合は、リトルエンディアンエンコーディング(Intel)を使用します。
	ModeSignalModeValue	モード信号のモード値。
	ModeSignalRawValueFormat	モード信号の生値の形式。
	ModeSignalStartBit	CAN busメッセージ内のモード信号開始ビットの位置。
	PayloadByteCount	メッセージのデータ部のサイズ。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	RawValueFormat	生値の形式。
R	SensorType	センサの種類(読み取り専用)。
	SignalLength	信号のビット数。
	SignalType	CAN信号の種類。
	StartBit	CAN busメッセージ内のスタートビットの位置。
	Timeout	2つの信号間の待機時間(ミリ秒単位)。`Timeout = 0`の場合、タイムアウトは使用しません。

CanBusSensor.ActiveQuery Property

メッセージの内容。

C++

```
__property string ActiveQuery;
```

C#

```
public string ActiveQuery;
```

Visual Basic

```
Public Property ActiveQuery() As string
```

備考

将来の使用のために予約されています。

CanBusSensor.ActiveQueryRate Property

クエリが送信される頻度を評価します。

C++

```
__property decimal ActiveQueryRate;
```

C#

```
public decimal ActiveQueryRate;
```

Visual Basic

```
Public Property ActiveQueryRate() As decimal
```

備考

将来の使用のために予約されています。

CanBusSensor.ActiveSupplyVoltage Property

有効な供給電圧の値。このプロパティは情報提供のみを目的としています。

C++

```
__property decimal ActiveSupplyVoltage;
```

C#

```
public decimal ActiveSupplyVoltage;
```

Visual Basic

```
Public Property ActiveSupplyVoltage() As decimal
```

備考

これは、このセンサに電力を供給するために必要な電圧です。これは CAN bus(CanConnector)全体に対してのみ設定できるため、この値は直接的な影響を与えず、情報提供のみを目的としています。CanConnectorの供給電圧を設定するには、Hbm.Api.Common.Entities.Connectors.CanConnector.SupplyVoltageプロパティを使用してください。

CanBusSensor.BitSequence Property

ビッグエンディアンエンコーディング(Motorola)を使用しているかどうかを示します。それ以外の場合は、リトルエンディアンエンコーディング(Intel)を使用します。

C++

```
__property BitSequence BitSequence;
```

C#

```
public BitSequence BitSequence;
```

Visual Basic

```
Public Property BitSequence() As BitSequence
```

CanBusSensor.IsActiveQueryEnabled Property

センサを活性化してデータを送信します。

C++

```
__property bool IsActiveQueryEnabled;
```

C#

```
public bool IsActiveQueryEnabled;
```

Visual Basic

```
Public Property IsActiveQueryEnabled() As bool
```

備考

将来の使用のために予約されています。

CanBusSensor.IsExtendedFrameFormat Property

拡張フレーム形式(29ビット識別子)を使用しているかどうかを示します。それ以外の場合は、標準フレーム形式(11ビット識別子)を使用します。

C++

```
__property bool IsExtendedFrameFormat;
```

C#

```
public bool IsExtendedFrameFormat;
```

Visual Basic

```
Public Property IsExtendedFrameFormat() As bool
```

CanBusSensor.MessageID Property

センサのCAN busメッセージの識別子。

C++

```
__property int MessageID;
```

C#

```
public int MessageID;
```

Visual Basic

```
Public Property MessageID() As Integer
```

CanBusSensor.ModeSignalBitCount Property

モード信号のビット数。

C++

```
__property int ModeSignalBitCount;
```

C#

```
public int ModeSignalBitCount;
```

Visual Basic

```
Public Property ModeSignalBitCount() As Integer
```

CanBusSensor.ModeSignalBitSequence Property

モード信号にビッグエンディアンエンコーディング(Motorola)を使用しているかどうかを示します。それ以外の場合は、リトルエンディアンエンコーディング(Intel)を使用します。

C++

```
__property BitSequence ModeSignalBitSequence;
```

C#

```
public BitSequence ModeSignalBitSequence;
```

Visual Basic

```
Public Property ModeSignalBitSequence() As BitSequence
```


現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace -> Hbm.Api.SensorDB.Entities.Sensors Namespace -> CanBusSensor Class -> CanBusSensor.CanBusSensor Constructor

CanBusSensor.ModeSignalModeValue Property

モード信号のモード値。

C++

```
__property int ModeSignalModeValue;
```

C#

```
public int ModeSignalModeValue;
```

Visual Basic

```
Public Property ModeSignalModeValue() As Integer
```

CanBusSensor.ModeSignalRawValueFormat Property

モード信号の生値の形式。

C++

```
__property RawValueFormat ModeSignalRawValueFormat;
```

C#

```
public RawValueFormat ModeSignalRawValueFormat;
```

Visual Basic

```
Public Property ModeSignalRawValueFormat() As RawValueFormat
```

CanBusSensor.ModeSignalStartBit Property

CAN busメッセージ内のモード信号開始ビットの位置。

C++

```
__property int ModeSignalStartBit;
```

C#

```
public int ModeSignalStartBit;
```

Visual Basic

```
Public Property ModeSignalStartBit() As Integer
```

CanBusSensor.PayloadByteCount Property

メッセージのデータ部のサイズ。

C++

```
__property int PayloadByteCount;
```

C#

```
public int PayloadByteCount;
```

Visual Basic

```
Public Property PayloadByteCount() As Integer
```

CanBusSensor.RawValueFormat Property

生値の形式。

C++

```
__property RawValueFormat RawValueFormat;
```

C#

```
public RawValueFormat RawValueFormat;
```

Visual Basic

```
Public Property RawValueFormat() As RawValueFormat
```

CanBusSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```


現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace -> Hbm.Api.SensorDB.Entities.Sensors Namespace -> CanBusSensor Class -> CanBusSensor.CanBusSensor Constructor

CanBusSensor.SignalLength Property

信号のビット数。

C++

```
__property int SignalLength;
```

C#

```
public int SignalLength;
```

Visual Basic

```
Public Property SignalLength() As Integer
```

CanBusSensor.SignalType Property

CAN信号の種類。

C++

```
__property CanSignalType SignalType;
```

C#

```
public CanSignalType SignalType;
```

Visual Basic

```
Public Property SignalType() As CanSignalType
```

CanBusSensor.StartBit Property

CAN busメッセージ内のスタートビットの位置。

C++

```
__property int StartBit;
```

C#

```
public int StartBit;
```

Visual Basic

```
Public Property StartBit() As Integer
```

CanBusSensor.Timeout Property

2つの信号間の待機時間(ミリ秒単位)。Timeout = 0の場合、タイムアウトは使用しません。

C++

```
__property int Timeout;
```

C#

```
public int Timeout;
```

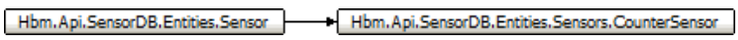
Visual Basic

```
Public Property Timeout() As Integer
```


現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace -> Hbm.Api.SensorDB.Entities.Sensors Namespace -> CanBusSensor Class -> CanBusSensor.CanBusSensor Constructor

CounterSensor Class

カウンタセンサを表します。



C++

```
[Serializable]
class CounterSensor : public Sensor;
```

C#

```
[Serializable]
public class CounterSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class CounterSensor
Inherits Sensor
```

ファイル

CounterSensor.cs

CounterSensor Members

次の表は、CanBusSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CounterSensor	新しいインスタンスを作成し、データを初期化します。
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.CounterSensor.ActiveSupplyTypeによって定義されます。
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	DigitalInputType	センサのデジタル入力の種類。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	GlitchFilter	グリッチフィルタ値(ns単位、0 = off)。
	Info	センサの追加情報(またはデータ)。
	IsInductive	センサを誘導型としてマークします。
	IsQuadFrequencyEnabled	2つの90度信号の使用と、信号の立ち上がりエッジと立ち下がりエッジの解釈(より高い解像度を得るため)を示します。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsTerminationEnabled	終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.CounterSensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	Polarity	極性(TTLロジック)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。

	名前	説明
	PulsesPerRound	このプロパティの解釈は、UseReferencePulseの値によって異なります。UseReferencePulseがtrueの場合、x個の参照パルスの発生後にカウンタ値をリセットし、それ以外の場合はnカウント後にカウンタ値をリセットします。1ラウンド内のパルス数。
	ReferencePulseDivisor	基準パルス除数(n回毎のインデックスリセットのみを考慮します)。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	Signal2Type	Signal2の種類。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。
	UseReferencePulse	基準パルスを使用してカウンタ値をリセットするかどうかを示します。

CounterSensor.CounterSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
CounterSensor();
```

C#

```
public CounterSensor();
```

Visual Basic

```
Public Sub New()
```

CounterSensor Properties

CounterSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.CounterSensor.ActiveSupplyTypeによって定義されます。
	DigitalInputType	センサのデジタル入力の種類。
	GlitchFilter	グリッチフィルタ値(ns単位、0 = off)。
	IsInductive	センサを誘導型としてマークします。
	IsQuadFrequencyEnabled	2つの90度信号の使用と、信号の立ち上がりエッジと立ち下がりエッジの解釈(より高い解像度を得るため)を示します。
	IsTerminationEnabled	終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.CounterSensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。
	Polarity	極性(TTLロジック)。
	PulsesPerRound	このプロパティの解釈は、UseReferencePulseの値によって異なります。UseReferencePulseがtrueの場合、x個の参照パルスの発生後にカウンタ値をリセットし、それ以外の場合はnカウント後にカウンタ値をリセットします。1ラウンド内のパルス数。
	ReferencePulseDivisor	基準パルス除数(n回毎のインデックスリセットのみを考慮します)。
R	SensorType	センサの種類(読み取り専用)。
	Signal2Type	Signal2の種類。
	UseReferencePulse	基準パルスを使用してカウンタ値をリセットするかどうかを示します。

CounterSensor.ActiveSupplyType Property

有効な供給値の種類。

C++

```
__property ActiveSupplyType ActiveSupplyType;
```

C#

```
public ActiveSupplyType ActiveSupplyType;
```

Visual Basic

```
Public Property ActiveSupplyType() As ActiveSupplyType
```

CounterSensor.ActiveSupplyValue Property

有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.CounterSensor.ActiveSupplyTypeによって定義されます。

C++

```
__property decimal ActiveSupplyValue;
```

C#

```
public decimal ActiveSupplyValue;
```

Visual Basic

```
Public Property ActiveSupplyValue() As decimal
```

CounterSensor.DigitalInputType Property

センサのデジタル入力の種類。

C++

```
__property DigitalInputType DigitalInputType;
```

C#

```
public DigitalInputType DigitalInputType;
```

Visual Basic

```
Public Property DigitalInputType() As DigitalInputType
```

CounterSensor.GlitchFilter Property

グリッチフィルタ値(ns単位、0 = off)。

C++

```
__property int GlitchFilter;
```

C#

```
public int GlitchFilter;
```

Visual Basic

```
Public Property GlitchFilter() As Integer
```

CounterSensor.IsInductive Property

センサを誘導型としてマークします。

C++

```
__property bool IsInductive;
```

C#

```
public bool IsInductive;
```

Visual Basic

```
Public Property IsInductive() As bool
```

CounterSensor.IsQuadFrequencyEnabled Property

2つの90度信号の使用と、信号の立ち上がりエッジと立ち下がりエッジの解釈(より高い解像度を得るため)を示します。

C++

```
__property bool IsQuadFrequencyEnabled;
```

C#

```
public bool IsQuadFrequencyEnabled;
```

Visual Basic

```
Public Property IsQuadFrequencyEnabled() As bool
```

CounterSensor.IsTerminationEnabled Property

終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.CounterSensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。

C++

```
__property bool IsTerminationEnabled;
```

C#

```
public bool IsTerminationEnabled;
```

Visual Basic

```
Public Property IsTerminationEnabled() As bool
```

CounterSensor.Polarity Property

極性(TTLロジック)。

C++

```
__property PolarityType Polarity;
```

C#

```
public PolarityType Polarity;
```

Visual Basic

```
Public Property Polarity() As PolarityType
```

CounterSensor.PulsesPerRound Property

このプロパティの解釈は、UseReferencePulseの値によって異なります。UseReferencePulseがtrueの場合、x個の参照パルスの発生後にカウンタ値をリセットし、それ以外の場合はnカウント後にカウンタ値をリセットします。1ラウンド内のパルス数。

C++

```
__property int PulsesPerRound;
```

C#

```
public int PulsesPerRound;
```

Visual Basic

```
Public Property PulsesPerRound() As Integer
```

CounterSensor.ReferencePulseDivisor Property

基準パルス除数(n回毎のインデックスリセットのみを考慮します)。

C++

```
__property int ReferencePulseDivisor;
```

C#

```
public int ReferencePulseDivisor;
```

Visual Basic

```
Public Property ReferencePulseDivisor() As Integer
```

CounterSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

CounterSensor.Signal2Type Property

Signal2の種類。

C++

```
__property Signal2Type Signal2Type;
```

C#

```
public Signal2Type Signal2Type;
```

Visual Basic

```
Public Property Signal2Type() As Signal2Type
```

CounterSensor.UseReferencePulse Property

基準パルスを使用してカウンタ値をリセットするかどうかを示します。

C++

```
__property bool UseReferencePulse;
```

C#

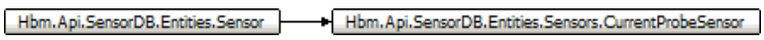
```
public bool UseReferencePulse;
```

Visual Basic

```
Public Property UseReferencePulse() As bool
```

CurrentProbeSensor Class

電流プローブセンサを表します。



C++

```
[Serializable]
class CurrentProbeSensor : public Sensor;
```

C#

```
[Serializable]
public class CurrentProbeSensor: Sensor;
```

Visual Basic

```
<Serializable>
Public Class CurrentProbeSensor
Inherits Sensor
```

ファイル

CurrentProbeSensor.cs

CurrentProbeSensor Members

次の表は、CurrentProbeSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CurrentProbeSensor	新しいインスタンスを作成し、データを初期化します。
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.VoltageSensor.ActiveSupplyTypeによって定義されます。
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合はnullになります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合はnullになります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

CurrentProbeSensor.CurrentProbeSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
CurrentProbeSensor();
```

C#

```
public CurrentProbeSensor();
```

Visual Basic

```
Public Sub New()
```

CurrentProbeSensor Properties

CurrentProbeSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.VoltageSensor.ActiveSupplyTypeによって定義されます。
R	SensorType	センサの種類(読み取り専用)。

CurrentProbeSensor.ActiveSupplyType Property

有効な供給値の種類。

C++

```
__property ActiveSupplyType ActiveSupplyType;
```

C#

```
public ActiveSupplyType ActiveSupplyType;
```

Visual Basic

```
Public Property ActiveSupplyType() As ActiveSupplyType
```

CurrentProbeSensor.ActiveSupplyValue Property

有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.VoltageSensor.ActiveSupplyTypeによって定義されます。

C++

```
__property decimal ActiveSupplyValue;
```

C#

```
public decimal ActiveSupplyValue;
```

Visual Basic

```
Public Property ActiveSupplyValue() As decimal
```

CurrentProbeSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

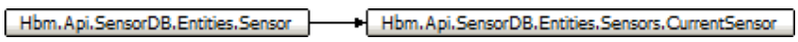
```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

CurrentSensor Class

電流センサを表します。



C++

```
[Serializable]
class CurrentSensor : public Sensor;
```

C#

```
[Serializable]
public class CurrentSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class CurrentSensor
Inherits Sensor
```

ファイル

CurrentSensor.cs

CurrentSensor Members

次の表は、CurrentSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CurrentSensor	新しいインスタンスを作成し、データを初期化します。
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.CurrentSensor.ActiveSupplyTypeによって定義されます。
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	CurrentType	電流の種類。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

CurrentSensor.CurrentSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
CurrentSensor();
```

C#

```
public CurrentSensor();
```

Visual Basic

```
Public Sub New()
```

CurrentSensor Properties

CurrentSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.CurrentSensor.ActiveSupplyTypeによって定義されます。
	CurrentType	電流の種類。
R	SensorType	センサの種類(読み取り専用)。

CurrentSensor.ActiveSupplyType Property

有効な供給値の種類。

C++

```
__property ActiveSupplyType ActiveSupplyType;
```

C#

```
public ActiveSupplyType ActiveSupplyType;
```

Visual Basic

```
Public Property ActiveSupplyType() As ActiveSupplyType
```

CurrentSensor.ActiveSupplyValue Property

有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.CurrentSensor.ActiveSupplyTypeによって定義されます。

C++

```
__property decimal ActiveSupplyValue;
```

C#

```
public decimal ActiveSupplyValue;
```

Visual Basic

```
Public Property ActiveSupplyValue() As decimal
```

CurrentSensor.CurrentType Property

電流の種類。

C++

```
__property CurrentType CurrentType;
```

C#

```
public CurrentType CurrentType;
```

Visual Basic

```
Public Property CurrentType() As CurrentType
```

CurrentSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

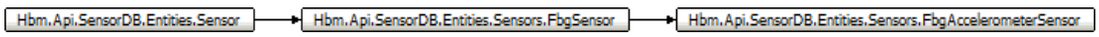
```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

FbgAccelerometerSensor Class

加速度計ブラッググレーティングセンサを表します。



C++

```
[Serializable]
class FbgAccelerometerSensor : public FbgSensor;
```

C#

```
[Serializable]
public class FbgAccelerometerSensor : FbgSensor;
```

Visual Basic

```
<Serializable>
Public Class FbgAccelerometerSensor
Inherits FbgSensor
```

ファイル

FbgAccelerometerSensor.cs

FbgAccelerometerSensor Members

次の表は、FbgAccelerometerSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	FbgAccelerometerSensor	新しいインスタンスを作成し、データを初期化します。
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	FbgSingleFactorCalibration	ファイバブラッググレーティング関連センサの単一係数校正情報。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	ReferenceWavelength	FBGセンサの基準波長。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

FbgAccelerometerSensor.FbgAccelerometerSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
FbgAccelerometerSensor();
```

C#

```
public FbgAccelerometerSensor();
```

Visual Basic

```
Public Sub New()
```

FbgAccelerometerSensor Properties

FbgAccelerometerSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	FbgSingleFactorCalibration	ファイバブラッググレーティング関連センサの単一係数校正情報。
R	SensorType	センサの種類(読み取り専用)。

FbgAccelerometerSensor.FbgSingleFactorCalibration Property

ファイバブラッググレーティング関連センサの単一係数校正情報。

C++

```
__property FbgSingleFactorCalibration FbgSingleFactorCalibration;
```

C#

```
public FbgSingleFactorCalibration FbgSingleFactorCalibration;
```

Visual Basic

```
Public Property FbgSingleFactorCalibration() As FbgSingleFactorCalibration
```

FbgAccelerometerSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

FbgGenericSensor Class

汎用ファイバブラッグレーティングセンサを表します。



C++

```
[Serializable]
class FbgGenericSensor : public FbgSensor;
```

C#

```
[Serializable]
public class FbgGenericSensor : FbgSensor;
```

Visual Basic

```
<Serializable>
Public Class FbgGenericSensor
Inherits FbgSensor
```

ファイル

FbgGenericSensor.cs

FbgGenericSensor Members

次の表は、FbgGenericSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	FbgGenericSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	FbgGenericType	型固有のプロパティを解釈するために使用する汎用型。
	FbgMultiFactorCalibration	ファイバブラッググレーティング関連センサの多係数校正情報。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	ReferenceWavelength	FBGセンサの基準波長。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

FbgGenericSensor.FbgGenericSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
FbgGenericSensor();
```

C#

```
public FbgGenericSensor();
```

Visual Basic

```
Public Sub New()
```

FbgGenericSensor Properties

FbgAccelerometerSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	FbgGenericType	型固有のプロパティを解釈するために使用する汎用型。
	FbgMultiFactorCalibration	ファイバブラッググレーティング関連センサの多係数校正情報。
R	SensorType	センサの種類(読み取り専用)。

FbgGenericSensor.FbgGenericType Property

型固有のプロパティを解釈するために使用する汎用型。

C++

```
__property string FbgGenericType;
```

C#

```
public string FbgGenericType;
```

Visual Basic

```
Public Property FbgGenericType() As string
```

FbgGenericSensor.FbgMultiFactorCalibration Property

ファイバブラッググレーティング関連センサの多係数校正情報。

C++

```
__property FbgMultiFactorCalibration FbgMultiFactorCalibration;
```

C#

```
public FbgMultiFactorCalibration FbgMultiFactorCalibration;
```

Visual Basic

```
Public Property FbgMultiFactorCalibration() As FbgMultiFactorCalibration
```

FbgGenericSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

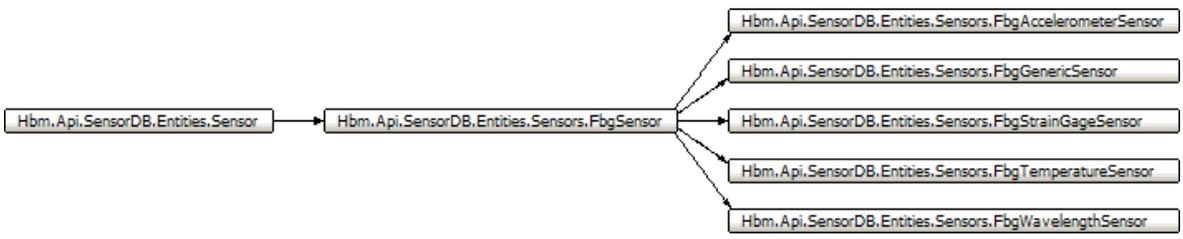
```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

FbgSensor Class

ファイバブラッググレーティングセンサを表します。



C++

```
[Serializable]
class FbgSensor : public Sensor;
```

C#

```
[Serializable]
public abstract class FbgSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public abstract Class FbgSensor
Inherits Sensor
```

ファイル

FbgSensor.cs

FbgSensor Members

次の表は、FbgSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	ReferenceWavelength	FBGセンサの基準波長。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

FbgSensor Properties

FbgSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ReferenceWavelength	FBGセンサの基準波長。

FbgSensor.ReferenceWavelength Property

FBGセンサの基準波長。

C++

```
__property double ReferenceWavelength;
```

C#

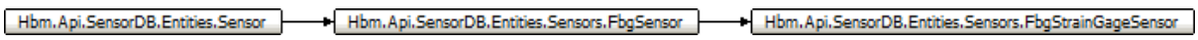
```
public double ReferenceWavelength;
```

Visual Basic

```
Public Property ReferenceWavelength()As double
```

FbgStrainGageSensor Class

ファイバブラッググレーティングひずみゲージセンサを表します。



C++

```
[Serializable]
class FbgStrainGageSensor : public FbgSensor;
```

C#

```
[Serializable]
public class FbgStrainGageSensor : FbgSensor;
```

Visual Basic

```
<Serializable>
Public Class FbgStrainGageSensor
Inherits FbgSensor
```

ファイル

FbgStrainGageSensor.cs

FbgStrainGageSensor Members

次の表は、FbgStrainGageSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	FbgStrainGageSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	FbgSingleFactorCalibration	ファイバブラッググレーティング関連センサの単一係数校正情報。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	ReferenceWavelength	FBGセンサの基準波長。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

FbgStrainGageSensor.FbgStrainGageSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
FbgStrainGageSensor();
```

C#

```
public FbgStrainGageSensor();
```

Visual Basic

```
Public Sub New()
```

FbgStrainGageSensor Properties

FbgStrainGageSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	FbgSingleFactorCalibration	ファイバブラッググレーティング関連センサの単一係数校正情報。
R	SensorType	センサの種類(読み取り専用)。

FbgStrainGageSensor.FbgSingleFactorCalibration Property

ファイバブラッググレーティング関連センサの単一係数校正情報。

C++

```
__property FbgSingleFactorCalibration FbgSingleFactorCalibration;
```

C#

```
public FbgSingleFactorCalibration FbgSingleFactorCalibration;
```

Visual Basic

```
Public Property FbgSingleFactorCalibration() As FbgSingleFactorCalibration
```

FbgStrainGageSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

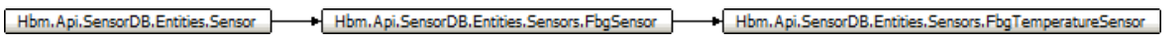
```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

FbgTemperatureSensor Class

ファイバブラッググレーティング温度センサを表します。



C++

```
[Serializable]
class FbgTemperatureSensor : public FbgSensor;
```

C#

```
[Serializable]
public class FbgTemperatureSensor : FbgSensor;
```

Visual Basic

```
<Serializable>
Public Class FbgTemperatureSensor
Inherits FbgSensor
```

ファイル

FbgTemperatureSensor.cs

FbgTemperatureSensor Members

次の表は、FbgTemperatureSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	FbgTemperatureSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	FbgMultiFactorCalibration	ファイバブラッググレーティング関連センサの多係数校正情報。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	ReferenceWavelength	FBGセンサの基準波長。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

FbgTemperatureSensor.FbgTemperatureSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
FbgTemperatureSensor();
```

C#

```
public FbgTemperatureSensor();
```

Visual Basic

```
Public Sub New()
```

FbgTemperatureSensor Properties

FbgTemperatureSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	FbgMultiFactorCalibration	ファイバブラッググレーティング関連センサの多係数校正情報。
R	SensorType	センサの種類(読み取り専用)。

FbgTemperatureSensor.FbgMultiFactorCalibration Property

ファイバブラッググレーティング関連センサの多係数校正情報。

C++

```
__property FbgMultiFactorCalibration FbgMultiFactorCalibration;
```

C#

```
public FbgMultiFactorCalibration FbgMultiFactorCalibration;
```

Visual Basic

```
Public Property FbgMultiFactorCalibration() As FbgMultiFactorCalibration
```

FbgTemperatureSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

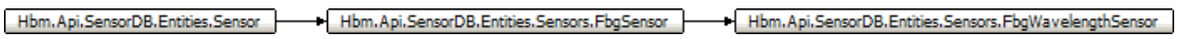
```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

FbgWavelengthSensor Class

ファイバブラッググレーティング波長センサを表します。



C++

```
[Serializable]
class FbgWavelengthSensor : public FbgSensor;
```

C#

```
[Serializable]
public class FbgWavelengthSensor : FbgSensor;
```

Visual Basic

```
<Serializable>
Public Class FbgWavelengthSensor
Inherits FbgSensor
```

ファイル

FbgWavelengthSensor.cs

FbgWavelengthSensor Members

次の表は、FbgWavelengthSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	FbgWavelengthSensor	これは、FbgWavelengthSensorクラスのメンバーであるFbgWavelengthSensorです。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsAbsolute	センサを絶対としてマークします。つまり、測定値は、センサの基準波長(相対)に対する[nm]の差ではなく、[nm]の絶対測定波長に対応します。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	ReferenceWavelength	FBGセンサの基準波長。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

FbgWavelengthSensor.FbgWavelengthSensor Constructor

これは、FbgWavelengthSensorクラスのメンバーであるFbgWavelengthSensorです。

C++

```
FbgWavelengthSensor();
```

C#

```
public FbgWavelengthSensor();
```

Visual Basic

```
Public Sub New()
```

FbgWavelengthSensor Properties

FbgTemperatureSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	IsAbsolute	センサを絶対としてマークします。つまり、測定値は、センサの基準波長(相対)に対する[nm]の差ではなく、[nm]の絶対測定波長に対応します。
R	SensorType	センサの種類(読み取り専用)。

FbgWavelengthSensor.IsAbsolute Property

センサを絶対としてマークします。つまり、測定値は、センサの基準波長(相対)に対する[nm]の差ではなく、[nm]の絶対測定波長に対応します。

C++

```
__property bool IsAbsolute;
```

C#

```
public bool IsAbsolute;
```

Visual Basic

```
Public Property IsAbsolute() As bool
```

FbgWavelengthSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

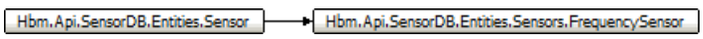
```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

FrequencySensor Class

周波数センサを表します。



C++

```
[Serializable]
class FrequencySensor : public Sensor;
```

C#

```
[Serializable]
public class FrequencySensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class FrequencySensor
Inherits Sensor
```

ファイル

FrequencySensor.cs

FrequencySensor Members

次の表は、FrequencySensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	FrequencySensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.FrequencySensor.ActiveSupplyTypeによって定義されます。
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	DigitalInputType	センサのデジタル入力の種類。このプロパティは、非誘導周波数センサにのみ関連します(Hbm.Api.SensorDB.Entities.Sensors.FrequencySensor.IsInductiveが`false`に設定されている場合)。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	GlitchFilter	グリッチフィルタ値(ns単位、0 = off)。
	Info	センサの追加情報(またはデータ)。
	IsInductive	センサを誘導型としてマークします。
	IsQuadFrequencyEnabled	2つの90度信号の使用と、信号の立ち上がりエッジと立ち下がりエッジの解釈(より高い解像度を得るため)を示します。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsTerminationEnabled	終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.FrequencySensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	Polarity	極性(TTLロジック)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。

	名前	説明
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	Signal2Type	Signal2の種類。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

FrequencySensor.FrequencySensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
FrequencySensor();
```

C#

```
public FrequencySensor();
```

Visual Basic

```
Public Sub New()
```

FrequencySensor Properties

FrequencySensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.FrequencySensor.ActiveSupplyTypeによって定義されます。
	DigitalInputType	センサのデジタル入力の種類。このプロパティは、非誘導周波数センサにのみ関連します(Hbm.Api.SensorDB.Entities.Sensors.FrequencySensor.IsInductiveが`false`に設定されている場合)。
	GlitchFilter	グリッチフィルタ値(ns単位、0 = off)。
	IsInductive	センサを誘導型としてマークします。
	IsQuadFrequencyEnabled	2つの90度信号の使用と、信号の立ち上がりエッジと立ち下がりエッジの解釈(より高い解像度を得るため)を示します。
	IsTerminationEnabled	終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.FrequencySensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。
	Polarity	極性(TTLロジック)。
R	SensorType	センサの種類(読み取り専用)。
	Signal2Type	Signal2の種類。

FrequencySensor.ActiveSupplyType Property

有効な供給値の種類。

C++

```
__property ActiveSupplyType ActiveSupplyType;
```

C#

```
public ActiveSupplyType ActiveSupplyType;
```

Visual Basic

```
Public Property ActiveSupplyType() As ActiveSupplyType
```

FrequencySensor.ActiveSupplyValue Property

有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.FrequencySensor.ActiveSupplyTypeによって定義されます。

C++

```
__property decimal ActiveSupplyValue;
```

C#

```
public decimal ActiveSupplyValue;
```

Visual Basic

```
Public Property ActiveSupplyValue() As decimal
```

FrequencySensor.DigitalInputType Property

センサのデジタル入力の種類。このプロパティは、非誘導周波数センサにのみ関連します(Hbm.Api.SensorDB.Entities.Sensors.FrequencySensor.IsInductiveが`false`に設定されている場合)。

C++

```
__property DigitalInputType DigitalInputType;
```

C#

```
public DigitalInputType DigitalInputType;
```

Visual Basic

```
Public Property DigitalInputType() As DigitalInputType
```

FrequencySensor.GlitchFilter Property

グリッチフィルタ値(ns単位、0 = off)。

C++

```
__property int GlitchFilter;
```

C#

```
public int GlitchFilter;
```

Visual Basic

```
Public Property GlitchFilter() As Integer
```

FrequencySensor.IsInductive Property

センサを誘導型としてマークします。

C++

```
__property bool IsInductive;
```

C#

```
public bool IsInductive;
```

Visual Basic

```
Public Property IsInductive() As bool
```

FrequencySensor.IsQuadFrequencyEnabled Property

2つの90度信号の使用と、信号の立ち上がりエッジと立ち下がりエッジの解釈(より高い解像度を得るため)を示します。

C++

```
__property bool IsQuadFrequencyEnabled;
```

C#

```
public bool IsQuadFrequencyEnabled;
```

Visual Basic

```
Public Property IsQuadFrequencyEnabled() As bool
```

FrequencySensor.IsTerminationEnabled Property

終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.FrequencySensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。

C++

```
__property bool IsTerminationEnabled;
```

C#

```
public bool IsTerminationEnabled;
```

Visual Basic

```
Public Property IsTerminationEnabled() As bool
```

FrequencySensor.Polarity Property

極性(TTLロジック)。

C++

```
__property PolarityType Polarity;
```

C#

```
public PolarityType Polarity;
```

Visual Basic

```
Public Property Polarity() As PolarityType
```

FrequencySensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```


現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace -> Hbm.Api.SensorDB.Entities.Sensors Namespace -> FrequencySensor Class -> FrequencySensor.FrequencySensor Constructor

FrequencySensor.Signal2Type Property

Signal2の種類。

C++

```
__property Signal2Type Signal2Type;
```

C#

```
public Signal2Type Signal2Type;
```

Visual Basic

```
Public Property Signal2Type() As Signal2Type
```

IepeSensor Class

IEPE(Integrated Electronics Piezo-Electric)センサを表します。



C++

```
[Serializable]
class IepeSensor : public Sensor;
```

C#

```
[Serializable]
public class IepeSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class IepeSensor
Inherits Sensor
```

ファイル

IepeSensor.cs

IepeSensor Members

次の表は、IepeSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	IepeSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

IepeSensor.IepeSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
IepeSensor();
```

C#

```
public IepeSensor();
```

Visual Basic

```
Public Sub New()
```

IepeSensor Properties

IepeSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	SensorType	センサの種類(読み取り専用)。

IepeSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

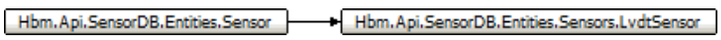
Visual Basic

```
Public override Property SensorType() As SensorType
```


現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace -> Hbm.Api.SensorDB.Entities.Sensors Namespace -> IepeSensor Class -> IepeSensor.IepeSensor Constructor

LvdtSensor Class

LVDT(差動変圧器)センサを表します。



C++

```
[Serializable]
class LvdtSensor : public Sensor;
```

C#

```
[Serializable]
public class LvdtSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class LvdtSensor
Inherits Sensor
```

ファイル

LvdtSensor.cs

LvdtSensor Members

次の表は、LvdtSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	LvdtSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

LvdtSensor.LvdtSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
LvdtSensor();
```

C#

```
public LvdtSensor();
```

Visual Basic

```
Public Sub New()
```

LvdtSensor Properties

LvdtSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	SensorType	センサの種類(読み取り専用)。

LvdtSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

NotSupportedSensor Class

サポートされていないセンサを表します。



C++

```
[Serializable]
class NotSupportedSensor : public Sensor;
```

C#

```
[Serializable]
public class NotSupportedSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class NotSupportedSensor
Inherits Sensor
```

ファイル

NotSupportedSensor.cs

NotSupportedSensor Members

次の表は、NotSupportedSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	NotSupportedSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

NotSupportedSensor.NotSupportedSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
NotSupportedSensor();
```

C#

```
public NotSupportedSensor();
```

Visual Basic

```
Public Sub New()
```

NotSupportedSensor Properties

NotSupportedSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	SensorType	センサの種類(読み取り専用)。

NotSupportedSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

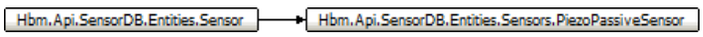
```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

PiezoPassiveSensor Class

パッシブピエゾセンサを表します。



C++

```
[Serializable]
class PiezoPassiveSensor : public Sensor;
```

C#

```
[Serializable]
public class PiezoPassiveSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class PiezoPassiveSensor
Inherits Sensor
```

ファイル

PiezoPassiveSensor.cs

PiezoPassiveSensor Members

次の表は、PiezoPassiveSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	PiezoPassiveSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	DischargeTimeType	放電時間の種類。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

PiezoPassiveSensor.PiezoPassiveSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
PiezoPassiveSensor();
```

C#

```
public PiezoPassiveSensor();
```

Visual Basic

```
Public Sub New()
```

PiezoPassiveSensor Properties

PiezoPassiveSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	DischargeTimeType	放電時間の種類。
R	SensorType	センサの種類(読み取り専用)。

PiezoPassiveSensor.DischargeTimeType Property

放電時間の種類。

C++

```
__property DischargeTimeType DischargeTimeType;
```

C#

```
public DischargeTimeType DischargeTimeType;
```

Visual Basic

```
Public Property DischargeTimeType() As DischargeTimeType
```

PiezoPassiveSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

PotentiometerSensor Class

ポテンショメータセンサを表します。



C++

```
[Serializable]
class PotentiometerSensor : public Sensor;
```

C#

```
[Serializable]
public class PotentiometerSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class PotentiometerSensor
Inherits Sensor
```

ファイル

PotentiometerSensor.cs

PotentiometerSensor Members

次の表は、PotentiometerSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	PotentiometerSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	NominalResistance	公称抵抗(オーム単位)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	PreferredExcitationVoltage	推奨励起電圧。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。
	Wiring	物理的なワイヤの本数。

PotentiometerSensor.PotentiometerSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
PotentiometerSensor();
```

C#

```
public PotentiometerSensor();
```

Visual Basic

```
Public Sub New()
```

PotentiometerSensor Properties

PotentiometerSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	NominalResistance	公称抵抗(オーム単位)。
	PreferredExcitationVoltage	推奨励起電圧。
R	SensorType	センサの種類(読み取り専用)。
	Wiring	物理的なワイヤの本数。

PotentiometerSensor.NominalResistance Property

公称抵抗(オーム単位)。

C++

```
__property decimal NominalResistance;
```

C#

```
public decimal NominalResistance;
```

Visual Basic

```
Public Property NominalResistance() As decimal
```

PotentiometerSensor.PreferredExcitationVoltage Property

推奨励起電圧。

C++

```
__property decimal PreferredExcitationVoltage;
```

C#

```
public decimal PreferredExcitationVoltage;
```

Visual Basic

```
Public Property PreferredExcitationVoltage() As decimal
```

PotentiometerSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

PotentiometerSensor.Wiring Property

物理的なワイヤの本数。

C++

```
__property PotentiometerSensorWiring Wiring;
```

C#

```
public PotentiometerSensorWiring Wiring;
```

Visual Basic

```
Public Property Wiring() As PotentiometerSensorWiring
```

PtSensor Class

プラチナ温度センサを表します。



C++

```
[Serializable]
class PtSensor : public Sensor;
```

C#

```
[Serializable]
public class PtSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class PtSensor
Inherits Sensor
```

ファイル

PtSensor.cs

PtSensor Members

次の表は、PtSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ConvertFromDegreeCelsius	指定した°Cの値を要求された単位に変換します(°C、°F、°Reamur、°Rankine、Kをサポートしています)。
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	GetMaxEngineeringOfPt	現在の工学単位(°C、°F、K、°R、°Rankのみ可能)に従って、PTセンサの最大工学範囲を返します。
	GetMinEngineeringOfPt	現在の工学単位(°C、°F、K、°R、°Rankのみ可能)に従って、PTセンサの最小工学範囲を返します。
	PtSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	NominalResistance	公称抵抗(オーム単位)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。
	Wiring	物理的なワイヤの本数。

PtSensor.PtSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
PtSensor();
```

C#

```
public PtSensor();
```

Visual Basic

```
Public Sub New()
```

PtSensor Methods

PtSensorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ConvertFromDegreeCelsius	指定した°Cの値を要求された単位に変換します(°C、°F、°Reamur、°Rankine、Kをサポートしています)。
	GetMaxEngineeringOfPt	現在の工学単位(°C、°F、K、°R、°Rankのみ可能)に従って、PTセンサの最大工学範囲を返します。
	GetMinEngineeringOfPt	現在の工学単位(°C、°F、K、°R、°Rankのみ可能)に従って、PTセンサの最小工学範囲を返します。

PtSensor.ConvertFromDegreeCelsius Method

指定した°Cの値を要求された単位に変換します(°C、°F、°Reamur、°Rankine、Kをサポートしています)。

C++

```
decimal ConvertFromDegreeCelsius(
    decimal valueInDegreeCelcius,
    string demandedUnit
);
```

C#

```
public decimal ConvertFromDegreeCelsius(
    decimal valueInDegreeCelcius,
    string demandedUnit
);
```

Visual Basic

```
Public Function ConvertFromDegreeCelsius(
    valueInDegreeCelcius As decimal,
    demandedUnit As string
)As decimal
```

引数

引数	説明
valueInDegreeCelcius	摂氏単位の値。
demandedUnit	要求された単位。

戻り値

要求した単位で摂氏値を指定します。

PtSensor.GetMaxEngineeringOfPt Method

現在の工学単位(°C、°F、K、°R、°Rankのみ可能)に従って、PTセンサの最大工学範囲を返します。

C++

```
decimal GetMaxEngineeringOfPt();
```

C#

```
public decimal GetMaxEngineeringOfPt();
```

Visual Basic

```
Public Function GetMaxEngineeringOfPt()As decimal
```

戻り値

現在の工学単位に応じたPTセンサの最大工学範囲。

PtSensor.GetMinEngineeringOfPt Method

現在の工学単位(°C、°F、K、°R、°Rankのみ可能)に従って、PTセンサの最小工学範囲を返します。

C++

```
decimal GetMinEngineeringOfPt();
```

C#

```
public decimal GetMinEngineeringOfPt();
```

Visual Basic

```
Public Function GetMinEngineeringOfPt()As decimal
```

戻り値

現在の工学単位に応じたPTセンサの最小工学範囲。

PtSensor Properties

PtSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	NominalResistance	公称抵抗(オーム単位)。
R	SensorType	センサの種類(読み取り専用)。
	Wiring	物理的なワイヤの本数。

PtSensor.NominalResistance Property

公称抵抗(オーム単位)。

C++

```
__property int NominalResistance;
```

C#

```
public int NominalResistance;
```

Visual Basic

```
Public Property NominalResistance() As Integer
```

備考

現時点では、次の値のみが許可されています: 10、100、500、または1000オーム。
セッタにはユーザ入力を制限するための追加の検証はありません。

PtSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```


現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace -> Hbm.Api.SensorDB.Entities.Sensors Namespace -> PtSensor Class -> PtSensor Properties

PtSensor.Wiring Property

物理的なワイヤの本数。

C++

```
__property PtSensorWiring Wiring;
```

C#

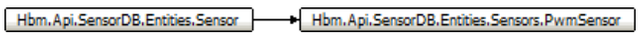
```
public PtSensorWiring Wiring;
```

Visual Basic

```
Public Property Wiring() As PtSensorWiring
```

PwmSensor Class

パルス幅変調センサを表します。



C++

```
[Serializable]
class PwmSensor : public Sensor;
```

C#

```
[Serializable]
public class PwmSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class PwmSensor
Inherits Sensor
```

ファイル

PwmSensor.cs

PwmSensor Members

次の表は、PwmSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	PwmSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.PwmSensor.ActiveSupplyTypeによって定義されます。
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	DigitalInputType	センサのデジタル入力の種類。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	GlitchFilter	グリッチフィルタ値(ns単位、0 = off)。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsTerminationEnabled	終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.PwmSensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	Polarity	極性(TTLロジック)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	PwmType	PWMセンサの種類。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。

	名前	説明
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

PwmSensor.PwmSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
PwmSensor();
```

C#

```
public PwmSensor();
```

Visual Basic

```
Public Sub New()
```

PwmSensor Properties

PwmSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.PwmSensor.ActiveSupplyTypeによって定義されます。
	DigitalInputType	センサのデジタル入力の種類。
	GlitchFilter	グリッチフィルタ値(ns単位、0 = off)。
	IsTerminationEnabled	終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.PwmSensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。
	Polarity	極性(TTLロジック)。
	PwmType	PWMセンサの種類。
R	SensorType	センサの種類(読み取り専用)。

PwmSensor.ActiveSupplyType Property

有効な供給値の種類。

C++

```
__property ActiveSupplyType ActiveSupplyType;
```

C#

```
public ActiveSupplyType ActiveSupplyType;
```

Visual Basic

```
Public Property ActiveSupplyType() As ActiveSupplyType
```

PwmSensor.ActiveSupplyValue Property

有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.PwmSensor.ActiveSupplyTypeによって定義されます。

C++

```
__property decimal ActiveSupplyValue;
```

C#

```
public decimal ActiveSupplyValue;
```

Visual Basic

```
Public Property ActiveSupplyValue() As decimal
```

PwmSensor.DigitalInputType Property

センサのデジタル入力の種類。

C++

```
__property DigitalInputType DigitalInputType;
```

C#

```
public DigitalInputType DigitalInputType;
```

Visual Basic

```
Public Property DigitalInputType() As DigitalInputType
```

PwmSensor.GlitchFilter Property

グリッチフィルタ値(ns単位、0 = off)。

C++

```
__property int GlitchFilter;
```

C#

```
public int GlitchFilter;
```

Visual Basic

```
Public Property GlitchFilter() As Integer
```

PwmSensor.IsTerminationEnabled Property

終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.PwmSensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。

C++

```
__property bool IsTerminationEnabled;
```

C#

```
public bool IsTerminationEnabled;
```

Visual Basic

```
Public Property IsTerminationEnabled() As bool
```

PwmSensor.Polarity Property

極性(TTLロジック)。

C++

```
__property PolarityType Polarity;
```

C#

```
public PolarityType Polarity;
```

Visual Basic

```
Public Property Polarity() As PolarityType
```

PwmSensor.PwmType Property

PWMセンサの種類。

C++

```
__property PwmType PwmType;
```

C#

```
public PwmType PwmType;
```

Visual Basic

```
Public Property PwmType() As PwmType
```

PwmSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

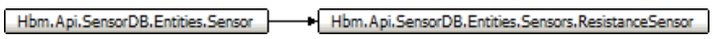
```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

ResistanceSensor Class

抵抗センサを表します。



C++

```
[Serializable]
class ResistanceSensor : public Sensor;
```

C#

```
[Serializable]
public class ResistanceSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class ResistanceSensor
Inherits Sensor
```

ファイル

ResistanceSensor.cs

ResistanceSensor Members

次の表は、ResistanceSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	ResistanceSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	NominalResistance	公称抵抗(オーム単位)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。
	Wiring	物理的なワイヤの本数。

ResistanceSensor.ResistanceSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
ResistanceSensor();
```

C#

```
public ResistanceSensor();
```

Visual Basic

```
Public Sub New()
```

ResistanceSensor Properties

ResistanceSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	NominalResistance	公称抵抗(オーム単位)。
R	SensorType	センサの種類(読み取り専用)。
	Wiring	物理的なワイヤの本数。

ResistanceSensor.NominalResistance Property

公称抵抗(オーム単位)。

C++

```
__property int NominalResistance;
```

C#

```
public int NominalResistance;
```

Visual Basic

```
Public Property NominalResistance() As Integer
```

備考

現時点では、次の値のみが許可されています: 10、100、500、または1000オーム。
セッタにはユーザ入力を制限するための追加の検証はありません。

ResistanceSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

ResistanceSensor.Wiring Property

物理的なワイヤの本数。

C++

```
__property ResistanceSensorWiring Wiring;
```

C#

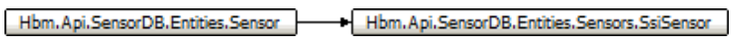
```
public ResistanceSensorWiring Wiring;
```

Visual Basic

```
Public Property Wiring() As ResistanceSensorWiring
```

SsiSensor Class

SSIセンサを表します。



C++

```
[Serializable]
class SsiSensor : public Sensor;
```

C#

```
[Serializable]
public class SsiSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class SsiSensor
Inherits Sensor
```

ファイル

SsiSensor.cs

SsiSensor Members

次の表は、SsiSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	SsiSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.SsiSensor.ActiveSupplyTypeによって定義されます。
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	ClockFrequency	クロック周波数(kHz)。
	CodingType	SSIコーディングの種類。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	DigitalInputType	センサのデジタル入力の種類。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsTerminationEnabled	終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.SsiSensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	ResolutionBits	解像度(ビット数)。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

SsiSensor.SsiSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
SsiSensor();
```

C#

```
public SsiSensor();
```

Visual Basic

```
Public Sub New()
```

SsiSensor Properties

SsiSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.SsiSensor.ActiveSupplyTypeによって定義されます。
	ClockFrequency	クロック周波数(kHz)。
	CodingType	SSIコーディングの種類。
	DigitalInputType	センサのデジタル入力の種類。
	IsTerminationEnabled	終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.SsiSensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。
	ResolutionBits	解像度(ビット数)。
R	SensorType	センサの種類(読み取り専用)。

SsiSensor.ActiveSupplyType Property

有効な供給値の種類。

C++

```
__property ActiveSupplyType ActiveSupplyType;
```

C#

```
public ActiveSupplyType ActiveSupplyType;
```

Visual Basic

```
Public Property ActiveSupplyType() As ActiveSupplyType
```

SsiSensor.ActiveSupplyValue Property

有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.SsiSensor.ActiveSupplyTypeによって定義されます。

C++

```
__property decimal ActiveSupplyValue;
```

C#

```
public decimal ActiveSupplyValue;
```

Visual Basic

```
Public Property ActiveSupplyValue() As decimal
```

SsiSensor.ClockFrequency Property

クロック周波数(kHz)。

C++

```
__property int ClockFrequency;
```

C#

```
public int ClockFrequency;
```

Visual Basic

```
Public Property ClockFrequency() As Integer
```

SsiSensor.CodingType Property

SSIコーディングの種類。

C++

```
__property SsiCodingType CodingType;
```

C#

```
public SsiCodingType CodingType;
```

Visual Basic

```
Public Property CodingType() As SsiCodingType
```

SsiSensor.DigitalInputType Property

センサのデジタル入力の種類。

C++

```
__property DigitalInputType DigitalInputType;
```

C#

```
public DigitalInputType DigitalInputType;
```

Visual Basic

```
Public Property DigitalInputType() As DigitalInputType
```

SsiSensor.IsTerminationEnabled Property

終了が有効かどうかを示します。このプロパティは、Hbm.Api.SensorDB.Entities.Sensors.SsiSensor.DigitalInputTypeがHbm.Api.SensorDB.Enums.DigitalInputType.Differentialに設定されている場合にのみ関連します。

C++

```
__property bool IsTerminationEnabled;
```

C#

```
public bool IsTerminationEnabled;
```

Visual Basic

```
Public Property IsTerminationEnabled() As bool
```

SsiSensor.ResolutionBits Property

解像度(ビット数)。

C++

```
__property int ResolutionBits;
```

C#

```
public int ResolutionBits;
```

Visual Basic

```
Public Property ResolutionBits() As Integer
```

SsiSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

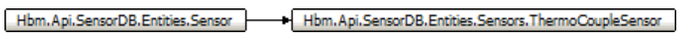
```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

ThermoCoupleSensor Class

熱電対センサを表します。



C++

```
[Serializable]
class ThermoCoupleSensor : public Sensor;
```

C#

```
[Serializable]
public class ThermoCoupleSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class ThermoCoupleSensor
Inherits Sensor
```

ファイル

ThermoCoupleSensor.cs

ThermoCoupleSensor Members

次の表は、ThermoCoupleSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ConvertFromDegreeCelsius	指定した°Cの値を要求された単位に変換します(°C、°F、°Reamur、°Rankine、Kをサポートしています)。
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	GetMaxEngineering	現在のThermoCoupleTypeとスケーリングの現在の工学単位に従って、最大工学範囲を返します
	GetMinEngineering	現在のThermoCoupleTypeとスケーリングの現在の工学単位に従って、最小工学範囲を返します。
	ThermoCoupleSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合はnullになります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合はnullになります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	ThermoCoupleType	熱電対センサの種類。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

ThermoCoupleSensor.ThermoCoupleSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
ThermoCoupleSensor();
```

C#

```
public ThermoCoupleSensor();
```

Visual Basic

```
Public Sub New()
```

ThermoCoupleSensor Methods

SsiSensorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ConvertFromDegreeCelsius	指定した°Cの値を要求された単位に変換します(°C、°F、°Reamur、°Rankine、Kをサポートしています)。
	GetMaxEngineering	現在のThermoCoupleTypeとスケーリングの現在の工学単位に従って、最大工学範囲を返します
	GetMinEngineering	現在のThermoCoupleTypeとスケーリングの現在の工学単位に従って、最小工学範囲を返します。
	ThermoCoupleSensor	新しいインスタンスを作成し、データを初期化します。

ThermoCoupleSensor.ConvertFromDegreeCelsius Method

指定した°Cの値を要求された単位に変換します(°C、°F、°Reamur、°Rankine、Kをサポートしています)。

C++

```
decimal ConvertFromDegreeCelsius(
    decimal valueInDegreeCelcius,
    string demandedUnit
);
```

C#

```
public decimal ConvertFromDegreeCelsius(
    decimal valueInDegreeCelcius,
    string demandedUnit
);
```

Visual Basic

```
Public Function ConvertFromDegreeCelsius(
    valueInDegreeCelcius As decimal,
    demandedUnit As string
)As decimal
```

引数

引数	説明
valueInDegreeCelcius	摂氏単位の値。
demandedUnit	要求された単位。

戻り値

要求した単位で摂氏値を指定します。

現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace -> Hbm.Api.SensorDB.Entities.Sensors Namespace -> ThermoCoupleSensor Class -> ThermoCoupleSensor.ThermoCoupleSensor Constructor

ThermoCoupleSensor.GetMaxEngineering Method

現在のThermoCoupleTypeとスケーリングの現在の工学単位に従って、最大工学範囲を返します。

C++

```
decimal GetMaxEngineering();
```

C#

```
public decimal GetMaxEngineering();
```

Visual Basic

```
Public Function GetMaxEngineering()As decimal
```

戻り値

現在のThermoCoupleTypeとスケーリングの現在の工学単位に従った、最大工学範囲。

ThermoCoupleSensor.GetMinEngineering Method

現在のThermoCoupleTypeとスケーリングの現在の工学単位に従って、最小工学範囲を返します。

C++

```
decimal GetMinEngineering();
```

C#

```
public decimal GetMinEngineering();
```

Visual Basic

```
Public Function GetMinEngineering()As decimal
```

戻り値

現在のThermoCoupleTypeとスケーリングの現在の工学単位に従った、最小工学範囲。

ThermoCoupleSensor Properties

ThermoCoupleSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	SensorType	センサの種類(読み取り専用)。
	ThermoCoupleType	熱電対センサの種類。

ThermoCoupleSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

ThermoCoupleSensor.ThermoCoupleType Property

熱電対センサの種類。

C++

```
__property string ThermoCoupleType;
```

C#

```
public string ThermoCoupleType;
```

Visual Basic

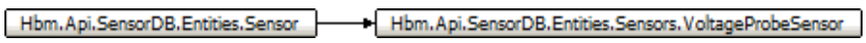
```
Public Property ThermoCoupleType() As string
```

備考

入力はトリム処理し、大文字に変換します。APIコードを変更せずに、さらに多くの型をリポジトリに追加できるため、このプロパティには列挙がありません。

VoltageProbeSensor Class

電圧プローブセンサを表します。



C++

```
[Serializable]
class VoltageProbeSensor : public Sensor;
```

C#

```
[Serializable]
public class VoltageProbeSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class VoltageProbeSensor
Inherits Sensor
```

ファイル

VoltageProbeSensor.cs

VoltageProbeSensor Members

次の表は、VoltageProbeSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	VoltageProbeSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.VoltageSensor.ActiveSupplyTypeによって定義されます。
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	MaxCompensationRange	最大補償値(F(ファラッド)単位)。
	MinCompensationRange	最小補償値(F(ファラッド)単位)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。
	VoltageProbeType	電圧プローブセンサの種類。

VoltageProbeSensor.VoltageProbeSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
VoltageProbeSensor();
```

C#

```
public VoltageProbeSensor();
```

Visual Basic

```
Public Sub New()
```

VoltageProbeSensor Properties

VoltageProbeSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.VoltageSensor.ActiveSupplyTypeによって定義されます。
	MaxCompensationRange	最大補償値(F(ファラッド)単位)。
	MinCompensationRange	最小補償値(F(ファラッド)単位)。
R	SensorType	センサの種類(読み取り専用)。
	VoltageProbeType	電圧プローブセンサの種類。

VoltageProbeSensor.ActiveSupplyType Property

有効な供給値の種類。

C++

```
__property ActiveSupplyType ActiveSupplyType;
```

C#

```
public ActiveSupplyType ActiveSupplyType;
```

Visual Basic

```
Public Property ActiveSupplyType() As ActiveSupplyType
```

VoltageProbeSensor.ActiveSupplyValue Property

有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.VoltageSensor.ActiveSupplyTypeによって定義されます。

C++

```
__property decimal ActiveSupplyValue;
```

C#

```
public decimal ActiveSupplyValue;
```

Visual Basic

```
Public Property ActiveSupplyValue() As decimal
```

VoltageProbeSensor.MaxCompensationRange Property

最大補償値(F(ファラッド)単位)。

C++

```
__property decimal MaxCompensationRange;
```

C#

```
public decimal MaxCompensationRange;
```

Visual Basic

```
Public Property MaxCompensationRange() As decimal
```

VoltageProbeSensor.MinCompensationRange Property

最小補償値(F(ファラッド)単位)。

C++

```
__property decimal MinCompensationRange;
```

C#

```
public decimal MinCompensationRange;
```

Visual Basic

```
Public Property MinCompensationRange() As decimal
```

VoltageProbeSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```


現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace -> Hbm.Api.SensorDB.Entities.Sensors Namespace -> VoltageProbeSensor Class -> VoltageProbeSensor.VoltageProbeSensor Constructor

VoltageProbeSensor.VoltageProbeType Property

電圧プローブセンサの種類。

C++

```
__property VoltageProbeType VoltageProbeType;
```

C#

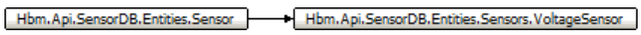
```
public VoltageProbeType VoltageProbeType;
```

Visual Basic

```
Public Property VoltageProbeType() As VoltageProbeType
```

VoltageSensor Class

電圧センサを表します。



C++

```
[Serializable]
class VoltageSensor : public Sensor;
```

C#

```
[Serializable]
public class VoltageSensor : Sensor;
```

Visual Basic

```
<Serializable>
Public Class VoltageSensor
Inherits Sensor
```

ファイル

VoltageSensor.cs

VoltageSensor Members

次の表は、VoltageSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	VoltageSensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.VoltageSensor.ActiveSupplyTypeによって定義されます。
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合はnullになります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合はnullになります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	CurrentType	電流の種類。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
R	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

VoltageSensor.VoltageSensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
VoltageSensor();
```

C#

```
public VoltageSensor();
```

Visual Basic

```
Public Sub New()
```

VoltageSensor Properties

VoltageSensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	ActiveSupplyValue	有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.VoltageSensor.ActiveSupplyTypeによって定義されます。
	CurrentType	電流の種類。
R	SensorType	センサの種類(読み取り専用)。

VoltageSensor.ActiveSupplyType Property

有効な供給値の種類。

C++

```
__property ActiveSupplyType ActiveSupplyType;
```

C#

```
public ActiveSupplyType ActiveSupplyType;
```

Visual Basic

```
Public Property ActiveSupplyType() As ActiveSupplyType
```

VoltageSensor.ActiveSupplyValue Property

有効な電源の値。この値の型は、Hbm.Api.SensorDB.Entities.Sensors.VoltageSensor.ActiveSupplyTypeによって定義されます。

C++

```
__property decimal ActiveSupplyValue;
```

C#

```
public decimal ActiveSupplyValue;
```

Visual Basic

```
Public Property ActiveSupplyValue() As decimal
```

VoltageSensor.CurrentType Property

電流の種類。

C++

```
__property CurrentType CurrentType;
```

C#

```
public CurrentType CurrentType;
```

Visual Basic

```
Public Property CurrentType() As CurrentType
```


現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace -> Hbm.Api.SensorDB.Entities.Sensors Namespace -> VoltageSensor Class -> VoltageSensor.VoltageSensor Constructor

VoltageSensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public override SensorType SensorType;
```

Visual Basic

```
Public override Property SensorType() As SensorType
```

Category Class

センサのカテゴリを表します。



C++

```
[Serializable]
class Category;
```

C#

```
[Serializable]
public class Category;
```

Visual Basic

```
<Serializable>
Public Class Category
```

ファイル

Category.cs

Category Members

次の表は、Categoryによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	CategoryID	リポジトリ内のカテゴリの主キー。
	Description	カテゴリを説明する名前。
	InternalID	データベースに依存しない番号を使用してカテゴリを一意に識別するための内部使用のID(例: カテゴリ"My Sensors"をプログラムで識別する場合)。
	IsDeletable	カテゴリを削除できないことをGUIに示すフラグ。
	IsLocked	カテゴリを変更できないことをGUIに示すフラグ。
	ParentCategoryID	カテゴリが属する親カテゴリのID、またはカテゴリがルートカテゴリの場合は `null`。
	PositionNo	親カテゴリ内でのカテゴリの位置。

Category Properties

Categoryクラスのプロパティの一覧です。

Public Properties

	名前	説明
	CategoryID	リポジトリ内のカテゴリの主キー。
	Description	カテゴリを説明する名前。
	InternalID	データベースに依存しない番号を使用してカテゴリを一意に識別するための内部使用のID(例: カテゴリ"My Sensors"をプログラムで識別する場合)。
	IsDeletable	カテゴリを削除できないことをGUIに示すフラグ。
	IsLocked	カテゴリを変更できないことをGUIに示すフラグ。
	ParentCategoryID	カテゴリが属する親カテゴリのID、またはカテゴリがルートカテゴリの場合は`null`。
	PositionNo	親カテゴリ内でのカテゴリの位置。

Category.CategoryID Property

リポジトリ内のカテゴリの主キー。

C++

```
[Key]
[ReadOnly(true)]
__property int CategoryID;
```

C#

```
[Key]
[ReadOnly(true)]
public int CategoryID;
```

Visual Basic

```
<Key>
<ReadOnly(true)>
Public Property CategoryID() As Integer
```

Category.Description Property

カテゴリを説明する名前。。

C++

```
__property string Description;
```

C#

```
public string Description;
```

Visual Basic

```
Public Property Description() As string
```

Category.InternalID Property

データベースに依存しない番号を使用してカテゴリを一意に識別するための内部使用のID(例: カテゴリ"My Sensors"をプログラムで識別する場合)。

C++

```
__property int? InternalID;
```

C#

```
public int? InternalID;
```

Visual Basic

```
Public Property InternalID() As int?
```

Category.IsDeletable Property

カテゴリを削除できないことをGUIに示すフラグ。

C++

```
__property bool IsDeletable;
```

C#

```
public bool IsDeletable;
```

Visual Basic

```
Public Property IsDeletable() As bool
```

Category.IsLocked Property

カテゴリを変更できないことをGUIに示すフラグ。

C++

```
__property bool IsLocked;
```

C#

```
public bool IsLocked;
```

Visual Basic

```
Public Property IsLocked() As bool
```

Category.ParentCategoryID Property

カテゴリが属する親カテゴリのID、またはカテゴリがルートカテゴリの場合は`null`。

C++

```
__property int? ParentCategoryID;
```

C#

```
public int? ParentCategoryID;
```

Visual Basic

```
Public Property ParentCategoryID() As int?
```

Category.PositionNo Property

親カテゴリ内でのカテゴリの位置。

C++

```
__property int PositionNo;
```

C#

```
public int PositionNo;
```

Visual Basic

```
Public Property PositionNo() As Integer
```

Descriptor Class

バイナリドキュメントなど、センサの記述子を表します。

Hbm.Api.SensorDB.Entities.Descriptor

C++

```
[Serializable]
class Descriptor;
```

C#

```
[Serializable]
public class Descriptor;
```

Visual Basic

```
<Serializable>
Public Class Descriptor
```

ファイル

Descriptor.cs

Descriptor Members

次の表は、Descriptorによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	ContentType	バイナリファイルのコンテンツタイプ(大文字のファイル拡張子)を取得します。
	Description	コンテンツの説明。
	DescriptorID	リポジトリ内の記述子の主キー。
	FileName	ファイル拡張子を含むコンテンツファイルのファイル名(パス情報なし)。
	SensorID	記述子が属するセンサのID。

Descriptor Properties

Descriptorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ContentType	バイナリファイルのコンテンツタイプ(大文字のファイル拡張子)を取得します。
	Description	コンテンツの説明。
	DescriptorID	リポジトリ内の記述子の主キー。
	FileName	ファイル拡張子を含むコンテンツファイルのファイル名(パス情報なし)。
	SensorID	記述子が属するセンサのID。

Descriptor.ContentType Property

バイナリファイルのコンテンツタイプ(大文字のファイル拡張子)を取得します。

C++

```
__property string ContentType;
```

C#

```
public string ContentType;
```

Visual Basic

```
Public Property ContentType() As string
```

Descriptor.Description Property

コンテンツの説明。

C++

```
__property string Description;
```

C#

```
public string Description;
```

Visual Basic

```
Public Property Description() As string
```

Descriptor.DescriptorID Property

リポジトリ内の記述子の主キー。

C++

```
[Key]
[ReadOnly(true)]
__property int DescriptorID;
```

C#

```
[Key]
[ReadOnly(true)]
public int DescriptorID;
```

Visual Basic

```
<Key>
[ReadOnly(true)]
Public Property DescriptorID() As Integer
```

Descriptor.FileName Property

ファイル拡張子を含むコンテンツファイルのファイル名(パス情報なし)。

C++

```
__property string FileName;
```

C#

```
public string FileName;
```

Visual Basic

```
Public Property FileName() As string
```

Descriptor.SensorID Property

記述子が属するセンサのID。

C++

```
[Key]
[ReadOnly(true)]
__property int SensorID;
```

C#

```
[Key]
[ReadOnly(true)]
public int SensorID;
```

Visual Basic

```
<Key>
[ReadOnly(true)]
Public Property SensorID() As Integer
```

HierarchicalCategory Class

カプセル化されたカテゴリの階層ビューを表します。カプセル化されたカテゴリを、階層的な子カテゴリのリストと子センサのリストに拡張します。



C++

```
[Serializable]
class HierarchicalCategory;
```

C#

```
[Serializable]
public class HierarchicalCategory;
```

Visual Basic

```
<Serializable>
Public Class HierarchicalCategory
```

ファイル

HierarchicalCategory.cs

HierarchicalCategory Members

次の表は、HierarchicalCategoryによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	HierarchicalCategory	これは、HierarchicalCategoryコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	Category	階層カテゴリが属するカテゴリ。
	HierarchicalCategoryChilds	カテゴリに基づいた階層的な子カテゴリ。
	Level	階層レベルを表します(ルート = 0から開始)。レベルが不明な場合は-1です。
	ParentHierarchicalCategory	このカテゴリの親カテゴリ。
	Sensors	カテゴリに準じた子センサ。

HierarchicalCategory Constructor

これは、HierarchicalCategoryコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	HierarchicalCategory. HierarchicalCategory ()	新しいインスタンスを作成します。
	HierarchicalCategory. HierarchicalCategory (Category, int)	新しいインスタンスを作成し、指定したカテゴリを保持します。

HierarchicalCategory.HierarchicalCategory Constructor ()

新しいインスタンスを作成します。

C++

```
HierarchicalCategory();
```

C#

```
public HierarchicalCategory();
```

Visual Basic

```
Public Sub New()
```

備考

シリアライゼーションに必要です。

HierarchicalCategory.HierarchicalCategory Constructor (Category, int)

新しいインスタンスを作成し、指定したカテゴリを保持します。

C++

```
HierarchicalCategory(
    Category category,
    int hierarchicalLevel
);
```

C#

```
public HierarchicalCategory(
    Category category,
    int hierarchicalLevel
);
```

Visual Basic

```
Public Sub New(
    category As Category,
    hierarchicalLevel As Integer
)
```

引数

引数	説明
category	カプセル化されるカテゴリ。
hierarchicalLevel	カテゴリの階層レベル。

HierarchicalCategory Properties

HierarchicalCategoryクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Category	階層カテゴリが属するカテゴリ。
	HierarchicalCategoryChilds	カテゴリに基づいた階層的な子カテゴリ。
	Level	階層レベルを表します(ルート = 0から開始)。レベルが不明な場合は-1です。
	ParentHierarchicalCategory	このカテゴリの親カテゴリ。
	Sensors	カテゴリに準じた子センサ。

HierarchicalCategory.Category Property

階層カテゴリが属するカテゴリ。

C++

```
__property Category Category;
```

C#

```
public Category Category;
```

Visual Basic

```
Public Property Category() As Category
```

HierarchicalCategory.HierarchicalCategoryChilds Property

カテゴリに基づいた階層的な子カテゴリ。

C++

```
__property List<HierarchicalCategory> HierarchicalCategoryChilds;
```

C#

```
public List<HierarchicalCategory> HierarchicalCategoryChilds;
```

Visual Basic

```
Public Property HierarchicalCategoryChilds() As List<HierarchicalCategory>
```

HierarchicalCategory.Level Property

階層レベルを表します(ルート = 0から開始)。レベルが不明な場合は-1です。

C++

```
__property int Level;
```

C#

```
public int Level;
```

Visual Basic

```
Public Property Level() As Integer
```

HierarchicalCategory.ParentHierarchicalCategory Property

このカテゴリの親カテゴリ。

]C++

```
__property HierarchicalCategory ParentHierarchicalCategory;
```

C#

```
public HierarchicalCategory ParentHierarchicalCategory;
```

Visual Basic

```
Public Property ParentHierarchicalCategory() As HierarchicalCategory
```

HierarchicalCategory.Sensors Property

カテゴリに準じた子センサ。

C++

```
__property List<Sensor> Sensors;
```

C#

```
public List<Sensor> Sensors;
```

Visual Basic

```
Public Property Sensors()As List<Sensor>
```

Language Class

翻訳に使用する言語をカプセル化します。



C++

```
[Serializable]
class Language;
```

C#

```
[Serializable]
public class Language;
```

Visual Basic

```
<Serializable>
Public Class Language
```

ファイル

Language.cs

Language Members

次の表は、Languageによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	IsFallbackLanguage	言語をフォールバック言語としてマークします。現在のユーザの言語がサポートされていない場合は、フォールバック言語を翻訳に使用する必要があります。フォールバック言語としてマークされている言語は1つだけであることが重要です。(内部使用のみ)。

Public Methods

	名前	説明
	Language	これは、Languageコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	Description	言語の説明(例: "English")。
	TwoLetterISOLanguageName	2文字のISO言語名(例: 英語の場合は"en")。リポジトリ内の言語の主キー。

Language Constructor

これは、Languageコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	Language.Language ()	新しいインスタンスを作成します。
	Language.Language (string, string)	新しいインスタンスを作成し、指定したパラメータを使用して新しいインスタンスを初期化します。

Language.Language Constructor ()

新しいインスタンスを作成します。

C++

```
Language();
```

C#

```
public Language();
```

Visual Basic

```
Public Sub New()
```

Language.Language Constructor (string, string)

新しいインスタンスを作成し、指定したパラメータを使用して新しいインスタンスを初期化します。

C++

```
Language(
    string twoLetterISOLanguageName,
    string description
);
```

C#

```
public Language(
    string twoLetterISOLanguageName,
    string description
);
```

Visual Basic

```
Public Sub New(
    twoLetterISOLanguageName As string,
    description As string
)
```

引数

引数	説明
twoLetterISOLanguageName	2文字のISO言語名。
description	言語の説明。

Language Properties

Languageクラスのプロパティの一覧です。

Internal Properties

	名前	説明
	IsFallbackLanguage	言語をフォールバック言語としてマークします。現在のユーザの言語がサポートされていない場合は、フォールバック言語を翻訳に使用する必要があります。フォールバック言語としてマークされている言語は1つだけであることが重要です。(内部使用のみ)。

Public Properties

	名前	説明
	Description	言語の説明(例: "English")。
	TwoLetterISOLanguageName	2文字のISO言語名(例: 英語の場合は"en")。リポジトリ内の言語の主キー。

Language.Description Property

言語の説明(例: "English")。

C++

```
__property string Description;
```

C#

```
public string Description;
```

Visual Basic

```
Public Property Description() As string
```

Language.IsFallbackLanguage Property

言語をフォールバック言語としてマークします。現在のユーザの言語がサポートされていない場合は、フォールバック言語を翻訳に使用する必要があります。フォールバック言語としてマークされている言語は1つだけであることが重要です。(内部使用のみ)。

C++

```
__property bool IsFallbackLanguage;
```

C#

```
internal bool IsFallbackLanguage;
```

Visual Basic

```
Friend Property IsFallbackLanguage() As bool
```

Language.TwoLetterISOLanguageName Property

2文字のISO言語名(例: 英語の場合は"en")。リポジトリ内の言語の主キー。

C++

```
[Key]
[ReadOnly(true)]
__property string TwoLetterISOLanguageName;
```

C#

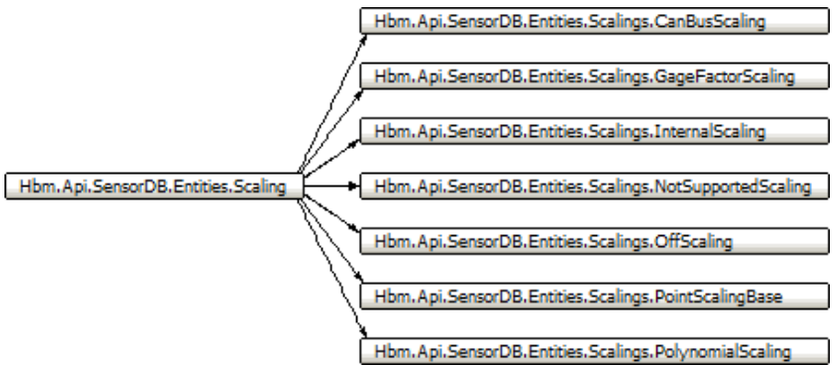
```
[Key]
[ReadOnly(true)]
public string TwoLetterISOLanguageName;
```

Visual Basic

```
<Key>
[ReadOnly(true)]
Public Property TwoLetterISOLanguageName() As Integer
```

Scaling Class

センサ値のスケーリングを表します。



C++

```
[Serializable]
class Scaling;
```

C#

```
[Serializable]
public abstract class Scaling;
```

Visual Basic

```
<Serializable>
Public abstract Class Scaling
```

ファイル

Scaling.cs

現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities Namespace

Scaling Members

次の表は、Scalingによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。

Public Properties

	名前	説明
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
	ScalingType	スケーリングタイプ(読み取り専用)。

Scaling Properties

Languageクラスのプロパティの一覧です。。

Internal Properties

	名前	説明
	ScalingID	リポジトリ内のスケーリングの主キー。

Public Properties

	名前	説明
	EngineeringUnit	物理単位(例: "kN"や"mm")。
	IsEngineeringRangeRms	エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。
	MaxEngineeringRange	最大の物理値。
	MaxHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最大の物理値。
	MinEngineeringRange	最小の物理値。
	MinHardwareEngineeringRange	現在設定されている測定範囲内でデバイスが測定できる最小の物理値。
	ScalingType	スケーリングタイプ(読み取り専用)。

Scaling.EngineeringUnit Property

物理単位(例: "kN"や"mm")。

C++

```
__property string EngineeringUnit;
```

C#

```
public string EngineeringUnit;
```

Visual Basic

```
Public Property EngineeringUnit() As string
```

Scaling.IsEngineeringRangeRms Property

エンジニアリング範囲(MinEngineeringRangeおよびMaxEngineeringRange)が二乗平均平方根値として指定されているかどうかを示します。

C++

```
__property bool IsEngineeringRangeRms;
```

C#

```
public bool IsEngineeringRangeRms;
```

Visual Basic

```
Public Property IsEngineeringRangeRms() As bool
```

Scaling.MaxEngineeringRange Property

最大の物理値。

C++

```
__property decimal MaxEngineeringRange;
```

C#

```
public decimal MaxEngineeringRange;
```

Visual Basic

```
Public Property MaxEngineeringRange() As decimal
```

Scaling.MaxHardwareEngineeringRange Property

現在設定されている測定範囲内でデバイスが測定できる最大の物理値。

C++

```
__property decimal MaxHardwareEngineeringRange;
```

C#

```
public decimal MaxHardwareEngineeringRange;
```

Visual Basic

```
Public Property MaxHardwareEngineeringRange() As decimal
```

Scaling.MinEngineeringRange Property

最小の物理値。

C++

```
__property decimal MinEngineeringRange;
```

C#

```
public decimal MinEngineeringRange;
```

Visual Basic

```
Public Property MinEngineeringRange() As decimal
```

Scaling.MinHardwareEngineeringRange Property

現在設定されている測定範囲内でデバイスが測定できる最小の物理値。

C++

```
__property decimal MinHardwareEngineeringRange;
```

C#

```
public decimal MinHardwareEngineeringRange;
```

Visual Basic

```
Public Property MinHardwareEngineeringRange() As decimal
```

Scaling.ScalingID Property

リポジトリ内のスケーリングの主キー。

C++

```
[XmlIgnore]
[JsonIgnore]
__property int ScalingID;
```

C#

```
[XmlIgnore]
[JsonIgnore]
internal int ScalingID;
```

Visual Basic

```
<XmlIgnore>
<JsonIgnore>
Friend Property ScalingID() As Integer
```

Scaling.ScalingType Property

スケーリングタイプ(読み取り専用)。

C++

```
__property ScalingType ScalingType;
```

C#

```
public abstract ScalingType ScalingType;
```

Visual Basic

```
Public abstract Property ScalingType() As ScalingType
```

備考

派生クラスによって実装する必要があります。

Sensor Class

温度、ブリッジ、電圧などの物理センサを表します。



C++

```
[Serializable]
class Sensor;
```

C#

```
[Serializable]
public abstract class Sensor;
```

Visual Basic

```
<Serializable>
Public abstract Class Sensor
```

ファイル

Sensor.cs

Sensor Members

次の表は、ResistanceSensorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。
	Sensor	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
A	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

Sensor.Sensor Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
Sensor();
```

C#

```
public Sensor();
```

Visual Basic

```
Public Sub New()
```

Sensor Methods

Sensorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Copy	現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。

Sensor.Copy Method

現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。

Sensor.Copy Method
現在のセンサの新しいディープコピーを作成します。新しいコピーでは、データベースに依存するIDがリセットされ、新しい独自の名前が生成されます。センサはユーザ定義としてマークされ、最終更新日が更新されます。

C++

```
Sensor Copy();
```

C#

```
public Sensor Copy();
```

Visual Basic

```
Public Function Copy()As Sensor
```

戻り値

現在のセンサのコピーとしての新しいセンサインスタンス。

備考

新しいセンサには記述子がありません。

Sensor Properties

Sensorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	CalibratedBy	センサの校正実行者の名前。
	CalibrationDate	校正の日付。指定されていない場合は`null`になります。
	CalibrationExpirationDate	校正の有効期限。指定されていない場合は`null`になります。
	Category	センサが属するカテゴリ。
	CategoryPositionNo	カテゴリ内のセンサの位置を並べ替えます。
	Comment	センサについてのコメント。
	Description	センサを説明する名前。
	Descriptors	センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。
	ElectricalUnit	センサの電気単位(読み取り専用)。
	ExpectedShuntValue	シャントの期待値。
	Info	センサの追加情報(またはデータ)。
	IsSingleUse	センサを単一チャネルにのみ割り当てられるかどうかを示します。
	IsUserDefined	このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。
	LastModifiedDate	データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。
	ManufacturerName	センサの製造元名(例: HBM)。
	ModelName	センサのモデル名。
	NeedsScalingAdaptation	事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。
	PreferredChannelName	センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。
	Scaling	必要なすべてのパラメータを使用してスケーリングします。
	SensorID	リポジトリ内のセンサの主キー。
	SensorType	センサの種類(読み取り専用)。
	SerialNumber	実在するセンサのシリアル番号。
	StorageLocation	センサを設置している場所。
	UniqueName	センサの独自のプログラム識別子(例: GUIDなど)。

Sensor.CalibratedBy Property

センサの校正実行者の名前。

C++

```
__property string CalibratedBy;
```

C#

```
public string CalibratedBy;
```

Visual Basic

```
Public Property CalibratedBy() As string
```

Sensor.CalibrationDate Property

校正の日付。指定されていない場合はnullになります。

C++

```
__property DateTime? CalibrationDate;
```

C#

```
public DateTime? CalibrationDate;
```

Visual Basic

```
Public Property CalibrationDate() As DateTime?
```

Sensor.CalibrationExpirationDate Property

校正の有効期限。指定されていない場合はnullになります。

C++

```
__property DateTime? CalibrationExpirationDate;
```

C#

```
public DateTime? CalibrationExpirationDate;
```

Visual Basic

```
Public Property CalibrationExpirationDate() As DateTime?
```

Sensor.Category Property

センサが属するカテゴリ。

C++

```
__property Category Category;
```

C#

```
public Category Category;
```

Visual Basic

```
Public Property Category() As Category
```

Sensor.CategoryPositionNo Property

カテゴリ内のセンサの位置を並べ替えます。

C++

```
__property int CategoryPositionNo;
```

C#

```
public int CategoryPositionNo;
```

Visual Basic

```
Public Property CategoryPositionNo() As Integer
```

Sensor.Comment Property

センサについてのコメント。

C++

```
__property string Comment;
```

C#

```
public string Comment;
```

Visual Basic

```
Public Property Comment() As string
```

Sensor.Description Property

センサを説明する名前。

C++

```
__property string Description;
```

C#

```
public string Description;
```

Visual Basic

```
Public Property Description() As string
```

Sensor.Descriptors Property

センサの記述子の読み取り専用コレクション。リストの変更はリポジトリに保存されないため、リスト項目を追加または変更することはできません。ただし、リストのインスタンスは変更される場合があります。

C++

```
[XmlIgnore]
[JsonIgnore]
__property ReadOnlyCollection<Descriptor> Descriptors;
```

C#

```
[XmlIgnore]
[JsonIgnore]
public ReadOnlyCollection<Descriptor> Descriptors;
```

Visual Basic

```
<XmlIgnore>
<JsonIgnore>
Public Property Descriptors() As ReadOnlyCollection<Descriptor>
```

Sensor.ElectricalUnit Property

センサの電気単位(読み取り専用)。

C++

```
[ReadOnly(true)]
[XmlIgnore]
[JsonIgnore]
__property string ElectricalUnit;
```

C#

```
[ReadOnly(true)]
[XmlIgnore]
[JsonIgnore]
public string ElectricalUnit;
```

Visual Basic

```
<ReadOnly(true)>
<XmlIgnore>
<JsonIgnore>
Public Property ElectricalUnit() As string
```

備考

読み取り専用プロパティはXMLにシリアライズできないため、XmlIgnoreとしてマークします。

Sensor.ExpectedShuntValue Property

シャントの期待値。

C++

```
__property decimal ExpectedShuntValue;
```

C#

```
public decimal ExpectedShuntValue;
```

Visual Basic

```
Public Property ExpectedShuntValue() As decimal
```

Sensor.Info Property

センサの追加情報(またはデータ)。

C++

```
__property string Info;
```

C#

```
public string Info;
```

Visual Basic

```
Public Property Info() As string
```

Sensor.IsSingleUse Property

センサを単一チャネルにのみ割り当てられるかどうかを示します。

C++

```
__property bool IsSingleUse;
```

C#

```
public bool IsSingleUse;
```

Visual Basic

```
Public Property IsSingleUse() As bool
```

Sensor.IsUserDefined Property

このセンサがHBMではなくユーザによって定義されているかどうかを示します。重要: このプロパティはHBMによる内部使用のみを目的としています。値を変更しないでください! このプロパティは、データベースの更新中の同期のために内部的に使用されます。

C++

```
__property bool IsUserDefined;
```

C#

```
public bool IsUserDefined;
```

Visual Basic

```
Public Property IsUserDefined() As bool
```

Sensor.LastModifiedDate Property

データベースの最終変更日。このプロパティは内部使用専用です。値を変更しないでください。

C++

```
__property DateTime LastModifiedDate;
```

C#

```
public DateTime LastModifiedDate;
```

Visual Basic

```
Public Property LastModifiedDate() As DateTime
```

備考

このプロパティのセッタは、シリアライゼーションの目的でのみパブリックです

Sensor.ManufacturerName Property

センサの製造元名(例: HBM)。

C++

```
__property string ManufacturerName;
```

C#

```
public string ManufacturerName;
```

Visual Basic

```
Public Property ManufacturerName() As string
```

Sensor.ModelName Property

センサのモデル名。

C++

```
__property string ModelName;
```

C#

```
public string ModelName;
```

Visual Basic

```
Public Property ModelName() As string
```

Sensor.NeedsScalingAdaptation Property

事前のスケーリングを適応させずにセンサを使用すべきかどうかを示します(trueの場合)。

C++

```
__property bool NeedsScalingAdaptation;
```

C#

```
public bool NeedsScalingAdaptation;
```

Visual Basic

```
Public Property NeedsScalingAdaptation() As bool
```

Sensor.PreferredChannelName Property

センサが属するチャネルの推奨名。チャネルに名前が付けられていない場合は、この名前をチャネル名として使用する必要があります。

C++

```
__property string PreferredChannelName;
```

C#

```
public string PreferredChannelName;
```

Visual Basic

```
Public Property PreferredChannelName() As string
```

Sensor.Scaling Property

必要なすべてのパラメータを使用してスケーリングします。

C++

```
__property Scaling Scaling;
```

C#

```
public Scaling Scaling;
```

Visual Basic

```
Public Property Scaling() As Scaling
```

Sensor.SensorID Property

リポジトリ内のセンサの主キー。

C++

```
[Key]
[ReadOnly(true)]
__property int SensorID;
```

C#

```
[Key]
[ReadOnly(true)]
public int SensorID;
```

Visual Basic

```
<Key>
<ReadOnly(true)>
Public Property SensorID() As Integer
```

Sensor.SensorType Property

センサの種類(読み取り専用)。

C++

```
__property SensorType SensorType;
```

C#

```
public abstract SensorType SensorType;
```

Visual Basic

```
Public abstract Property SensorType() As SensorType
```

備考

派生クラスによって実装する必要があります。

Sensor.SerialNumber Property

実在するセンサのシリアル番号。

C++

```
__property string SerialNumber;
```

C#

```
public string SerialNumber;
```

Visual Basic

```
Public Property SerialNumber() As string
```

Sensor.StorageLocation Property

センサを設置している場所。

C++

```
__property string StorageLocation;
```

C#

```
public string StorageLocation;
```

Visual Basic

```
Public Property StorageLocation() As string
```

Sensor.UniqueName Property

センサの独自のプログラム識別子(例: GUIDなど)。

C++

```
__property string UniqueName;
```

C#

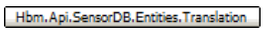
```
public string UniqueName;
```

Visual Basic

```
Public Property UniqueName() As string
```

Translation Class

翻訳済みエンティティをカプセル化します。



C++

```
[Serializable]
class Translation;
```

C#

```
[Serializable]
public class Translation;
```

Visual Basic

```
<Serializable>
Public Class Translation
```

ファイル

Translation.cs

Translation Members

次の表は、Translationによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Translation	これは、Translationコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	Description	説明(翻訳した項目)。
	TranslationID	リポジトリ内の翻訳の主キー。
	TwoLetterISOLanguageName	翻訳に使用する2文字のISO言語名。リポジトリに保存している言語と一致する必要があります。

Translation Constructor

これは、Translationコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	Translation.Translation ()	新しいインスタンスを作成します。
	Translation.Translation (int, string, string)	新しいインスタンスを作成し、指定したパラメータを使用して新しいインスタンスを初期化します。

Translation.Translation Constructor ()

新しいインスタンスを作成します。

C++

```
Translation();
```

C#

```
public Translation();
```

Visual Basic

```
Public Sub New()
```

Translation.Translation Constructor (int, string, string)

新しいインスタンスを作成し、指定したパラメータを使用して新しいインスタンスを初期化します。

C++

```
Translation(
    int translationID,
    string twoLetterISOLanguageName,
    string description
);
```

C#

```
public Translation(
    int translationID,
    string twoLetterISOLanguageName,
    string  description
);
```

Visual Basic

```
Public Sub New(
    translationID As Integer,
    twoLetterISOLanguageName As string,
    description As string
)
```

引数

引数	説明
translationID	翻訳のID。
twoLetterISOLanguageName	データベーステーブル"Language"内の言語の2文字のISO言語名。
description	説明(翻訳した項目)。

Translation Properties

Translationクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Description	説明(翻訳した項目)。
	TranslationID	リポジトリ内の翻訳の主キー。
	TwoLetterISOLanguageName	翻訳に使用する2文字のISO言語名。リポジトリに保存している言語と一致する必要があります。

Translation.Description Property

説明(翻訳した項目)。

C++

```
__property string Description;
```

C#

```
public string Description;
```

Visual Basic

```
Public Property Description() As string
```

Translation.TranslationID Property

リポジトリ内の翻訳の主キー。

C++

```
[Key]
[ReadOnly(true)]
__property int TranslationID;
```

C#

```
[Key]
[ReadOnly(true)]
public int TranslationID;
```

Visual Basic

```
<Key>
<ReadOnly(true)>
Public Property TranslationID() As Integer
```

Translation.TwoLetterISOLanguageName Property

翻訳に使用する2文字のISO言語名。リポジトリに保存している言語と一致する必要があります。

C++

```
__property string TwoLetterISOLanguageName;
```

C#

```
public string TwoLetterISOLanguageName;
```

Visual Basic

```
Public Property TwoLetterISOLanguageName() As string
```

VersionInfo Class

センサデータベースのバージョン情報をカプセル化します。

Hbm.Api.SensorDB.Entities.VersionInfo

C++

```
[Serializable]
class VersionInfo;
```

C#

```
[Serializable]
public class VersionInfo;
```

Visual Basic

```
<Serializable>
Public Class VersionInfo
```

ファイル

VersionInfo.cs

VersionInfo Members

次の表は、VersionInfoによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	VersionInfo	新しいインスタンスを作成します。

Public Properties

	名前	説明
	Description	バージョンの説明。
	IsHbmDb	データベースの種類: HBMまたはUser。
	VersionNumber	バージョン番号。

VersionInfo.VersionInfo Constructor

新しいインスタンスを作成します。

C++

```
VersionInfo();
```

C#

```
public VersionInfo();
```

Visual Basic

```
Public Sub New()
```

VersionInfo Properties

VersionInfoクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Description	バージョンの説明。
	IsHbmDb	データベースの種類: HBMまたはUser。
	VersionNumber	バージョン番号。

VersionInfo.Description Property

バージョンの説明。

C++

```
__property string Description;
```

C#

```
public string Description;
```

Visual Basic

```
Public Property Description() As string
```

VersionInfo.IsHbmDb Property

データベースの種類: HBMまたはUser。

C++

```
__property bool IsHbmDb;
```

C#

```
public bool IsHbmDb;
```

Visual Basic

```
Public Property IsHbmDb() As bool
```

VersionInfo.VersionNumber Property

バージョン番号。

C++

```
__property string VersionNumber;
```

C#

```
public string VersionNumber;
```

Visual Basic

```
Public Property VersionNumber() As string
```

Hbm.Api.SensorDB.Entities.FbgCalibration Namespace

これは名前空間Hbm.Api.SensorDB.Entities.FbgCalibrationです。

クラス

名前	説明
FbgMultiFactorCalibration	ファイバーブラッググレーティング関連センサの係数校正情報。
FbgSingleFactorCalibration	ファイバーブラッググレーティング関連センサの係数校正情報。

FbgMultiFactorCalibration Class

ファイバーブラッググレーティング関連センサの係数校正情報。

Hbm.Api.SensorDB.Entities.FbgCalibration.FbgMultiFactorCalibration

C++

```
[Serializable]
class FbgMultiFactorCalibration;
```

C#

```
[Serializable]
public class FbgMultiFactorCalibration;
```

Visual Basic

```
<Serializable>
Public Class FbgMultiFactorCalibration
```

ファイル

FbgMultiFactorCalibration.cs

現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Entities.FbgCalibration Namespace

FbgMultiFactorCalibration Members

次の表は、FbgMultiFactorCalibrationによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	CalibrationFactorX0	校正係数X0 (f(x)= X0 + X1*x + X2*x^2)。
	CalibrationFactorX1	校正係数X1 (f(x)= X0 + X1*x + X2*x^2)。
	CalibrationFactorX2	校正係数X2 (f(x)= X0 + X1*x + X2*x^2)。

FbgMultiFactorCalibration Properties

FbgMultiFactorCalibrationクラスのプロパティの一覧です。

Public Properties

	名前	説明
	CalibrationFactorX0	校正係数X0 (f(x)= X0 + X1*x + X2*x^2)。
	CalibrationFactorX1	校正係数X1 (f(x)= X0 + X1*x + X2*x^2)。
	CalibrationFactorX2	校正係数X2 (f(x)= X0 + X1*x + X2*x^2)。

FbgMultiFactorCalibration.CalibrationFactorX0 Property

校正係数X0 (f(x)= X0 + X1*x + X2*x^2)。

C++

```
__property double CalibrationFactorX0;
```

C#

```
public double CalibrationFactorX0;
```

Visual Basic

```
Public Property CalibrationFactorX0() As double
```

FbgMultiFactorCalibration.CalibrationFactorX1 Property

校正係数X1 (f(x)= X0 + X1*x + X2*x^2)。

C++

```
__property double CalibrationFactorX1;
```

C#

```
public double CalibrationFactorX1;
```

Visual Basic

```
Public Property CalibrationFactorX1() As double
```

FbgMultiFactorCalibration.CalibrationFactorX2 Property

校正係数X2 (f(x)= X0 + X1*x + X2*x^2)。

C++

```
__property double CalibrationFactorX2;
```

C#

```
public double CalibrationFactorX2;
```

Visual Basic

```
Public Property CalibrationFactorX2() As double
```

FbgSingleFactorCalibration Class

ファイバーブラッググレーティング関連センサの係数校正情報。

Hbm.Api.SensorDB.Entities.FbgCalibration.FbgSingleFactorCalibration

C++

```
[Serializable]
class FbgSingleFactorCalibration;
```

C#

```
[Serializable]
public class FbgSingleFactorCalibration;
```

Visual Basic

```
<Serializable>
Public Class FbgSingleFactorCalibration
```

ファイル

FbgSingleFactorCalibration.cs

FbgSingleFactorCalibration Members

次の表は、FbgSingleFactorCalibrationによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	CalibrationFactor	校正係数。

FbgSingleFactorCalibration Properties

FbgSingleFactorCalibrationクラスのプロパティの一覧です。

Public Properties

	名前	説明
	CalibrationFactor	校正係数。

FbgSingleFactorCalibration.CalibrationFactor Property

校正係数。

C++

```
__property double CalibrationFactor;
```

C#

```
public double CalibrationFactor;
```

Visual Basic

```
Public Property CalibrationFactor() As double
```

Hbm.Api.SensorDB.Enums Namespace

これは名前空間Hbm.Api.SensorDB.Enumsです。

列挙型

	名前	説明
	ActiveSupplyType	有効な供給値の種類。
	BitSequence	CANデータのビットシーケンスの種類。
	BridgeSensorWiring	BridgeSensorのワイヤの本数。
	BridgeType	ブリッジの種類。
	CanSignalType	CAN 信号の種類。
	CarrierFrequencyType	キャリア周波数の種類。
	CurrentType	電流の種類。
	DigitalInputType	デジタル入力の種類。
	DischargeTimeType	放電時間の種類。
	FrameFormat	CanSensorのフレーム形式。
	PolarityType	極性の種類(TTLロジック)。
	PotentiometerSensorWiring	PotentiometerSensorのワイヤの本数。
	PtSensorWiring	PtSensorのワイヤの本数。
	PwmType	PWMの種類。
	RawValueFormat	CAN信号からの生値の形式。
	ResistanceSensorWiring	ResistanceSensorのワイヤの本数。
	ScalingType	センサのスケーリングの種類。
	SensorType	センサの種類。
	Signal2Type	Signal2の種類。
	SsiCodingType	コードの種類。
	VoltageProbeType	VoltageProbeセンサの種類。

Hbm.Api.SensorDB.Enums.ActiveSupplyType Enumeration

有効な供給値の種類。

C++

```
enum ActiveSupplyType : public int {
  Voltage = 0,
  Current = 1
};
```

C#

```
public enum ActiveSupplyType : int {
  Voltage = 0,
  Current = 1
}
```

Visual Basic

```
Public Enum ActiveSupplyType
Inherits int
  Voltage = 0
  Current = 1
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
Voltage	有効な電圧の供給。
Current	有効な電流の供給。

備考

注意: この列挙体の有効な供給値の種類は、テーブルActiveSupplyType内のActiveSupplyTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.BitSequence Enumeration

CANデータのビットシーケンスの種類。

C++

```
enum BitSequence : public int {
  Intel = 0,
  Motorola = 1
};
```

C#

```
public enum BitSequence : int {
  Intel = 0,
  Motorola = 1
}
```

Visual Basic

```
Public Enum BitSequence
Inherits int
  Intel = 0
  Motorola = 1
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
Intel	Intel(リトルエンディアン)。
Motorola	Motorola(ビッグエンディアン)。な電流の供給。

Hbm.Api.SensorDB.Enums.BridgeSensorWiring Enumeration

BridgeSensorのワイヤの本数。

C++

```
enum BridgeSensorWiring : public int {
  NotSet = 0,
  ThreeWire = 3,
  FourWire = 4,
  FiveWire = 5,
  SixWire = 6
};
```

C#

```
public enum BridgeSensorWiring : int {
  NotSet = 0,
  ThreeWire = 3,
  FourWire = 4,
  FiveWire = 5,
  SixWire = 6
}
```

Visual Basic

```
Public Enum BridgeSensorWiring
Inherits int
  NotSet = 0
  ThreeWire = 3
  FourWire = 4
  FiveWire = 5
  SixWire = 6
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
NotSet	未設定の場合の初期値。
ThreeWire	3線。
FourWire	4線。
FiveWire	5線。
SixWire	6線。

備考

格納にはint値のみが使用されます。リポジトリ内に対応するテーブルはありません。

Hbm.Api.SensorDB.Enums.BridgeType Enumeration

ブリッジの種類。

C++

```
enum BridgeType : public int {
  FullBridge = 0,
  HalfBridge = 1,
  QuarterBridge = 2
};
```

C#

```
public enum BridgeType : int {
  FullBridge = 0,
  HalfBridge = 1,
  QuarterBridge = 2
}
```

Visual Basic

```
Public Enum BridgeType
Inherits int
  FullBridge = 0
  HalfBridge = 1
  QuarterBridge = 2
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
FullBridge	フルブリッジ(未設定の場合の初期値)。
HalfBridge	ハーフブリッジ。
QuarterBridge	クォータブリッジ。

備考

注意: この列挙体のブリッジは、BridgeTypeテーブル内のBridgeTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.CanSignalType Enumeration

CAN 信号の種類。

C++

```
enum CanSignalType : public int {
  Standard = 0,
  ModeDependent = 1,
  Mode = 2
};
```

C#

```
public enum CanSignalType : int {
  Standard = 0,
  ModeDependent = 1,
  Mode = 2
}
```

Visual Basic

```
Public Enum CanSignalType
Inherits int
  Standard = 0
  ModeDependent = 1
  Mode = 2
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
Standard	標準信号(未設定の場合の初期値)。
ModeDependent	モード依存信号。
Mode	モード信号。

備考

注意: この列挙体の信号は、SignalTypeテーブル内のSignalTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.CarrierFrequencyType Enumeration

キャリア周波数の種類。

C++

```
enum CarrierFrequencyType : public int {
  Auto = 0,
  AutoAc = 1,
  Dc = 2,
  Value = 3
};
```

C#

```
public enum CarrierFrequencyType : int {
  Auto = 0,
  AutoAc = 1,
  Dc = 2,
  Value = 3
}
```

Visual Basic

```
Public Enum CarrierFrequencyType
Inherits int
  Auto = 0
  AutoAc = 1
  Dc = 2
  Value = 3
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
Auto	自動。アンプに応じて、キャリア周波数または直流電圧のいずれかが励起に使用されます。
AutoAc	キャリアを使用しますが、周波数はアンプに委ねられます。
Dc	DCを使用します。
Value	キャリア周波数として、プロパティHbm.Api.SensorDB.Entities.Sensors.BridgeSensor.CarrierFrequencyValueの値を使用します。

Hbm.Api.SensorDB.Enums.CurrentType Enumeration

電流の種類。

C++

```
enum CurrentType : public int {
  DC = 0,
  AC = 1
};
```

C#

```
public enum CurrentType : int {
  DC = 0,
  AC = 1
}
```

Visual Basic

```
Public Enum CurrentType
Inherits int
  DC = 0
  AC = 1
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
DC	直流(DC)(未設定の場合の初期値)。
AC	交流(AC)。

備考

注意: この列挙体の値は、テーブルCurrentTypeのCurrentTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.DigitalInputType Enumeration

デジタル入力の種類。

C++

```
enum DigitalInputType : public int {
  Differential = 0,
  SingleEnded = 1
};
```

C#

```
public enum DigitalInputType : int {
  Differential = 0,
  SingleEnded = 1
}
```

Visual Basic

```
Public Enum DigitalInputType
Inherits int
  Differential = 0
  SingleEnded = 1
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
Differential	差動デジタル入力(未設定の場合の初期値)。
SingleEnded	シングルエンドデジタル信号。

備考

注意: この列挙体の値は、テーブルDigitalInputTypeのDigitalInputTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.DischargeTimeType Enumeration

放電時間の種類。

C++

```
enum DischargeTimeType : public int {
  Slow = 0,
  Medium = 1,
  Fast = 2
};
```

C#

```
public enum DischargeTimeType : int {
  Slow = 0,
  Medium = 1,
  Fast = 2
}
```

Visual Basic

```
Public Enum DischargeTimeType
Inherits int
  Slow = 0
  Medium = 1
  Fast = 2
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
Slow	遅い放電時間(未設定の場合の初期値)。
Medium	中程度の放電時間。
Fast	速い放電時間。

備考

注意: この列挙体の放電時間は、テーブルDischargeTimeのDischargeTimeTypeIDと一致している必要があります。

Hbm.Api.SensorDB.Enums.FrameFormat Enumeration

CanSensorのフレーム形式。

C++

```
enum FrameFormat {
  Standard = 0,
  Extended = 1
};
```

C#

```
public enum FrameFormat {
  Standard = 0,
  Extended = 1
}
```

Visual Basic

```
Public Enum FrameFormat
  Standard = 0
  Extended = 1
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
Standard	未設定の場合の初期値。
Extended	3線。

備考

格納にはint値のみが使用されます。リポジトリ内に対応するテーブルはありません。

Hbm.Api.SensorDB.Enums.PolarityType Enumeration

極性の種類(TTLロジック)。

C++

```
enum PolarityType : public int {
  NotInverted = 0,
  Inverted = 1
};
```

C#

```
public enum PolarityType : int {
  NotInverted = 0,
  Inverted = 1
}
```

Visual Basic

```
Public Enum PolarityType
Inherits int
  NotInverted = 0
  Inverted = 1
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
NotInverted	極性反転なし(未設定の場合の初期値)。
Inverted	極性反転。

備考

注意: この列挙体の極性タイプは、テーブルPolarityTypeのPolarityTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.PotentiometerSensorWiring Enumeration

PotentiometerSensorのワイヤの本数。

C++

```
enum PotentiometerSensorWiring : public int {
  NotSet = 0,
  ThreeWire = 3,
  FourWire = 4,
  FiveWire = 5,
  SixWire = 6
};
```

C#

```
public enum PotentiometerSensorWiring : int {
  NotSet = 0,
  ThreeWire = 3,
  FourWire = 4,
  FiveWire = 5,
  SixWire = 6
}
```

Visual Basic

```
Public Enum PotentiometerSensorWiring
Inherits int
  NotSet = 0
  ThreeWire = 3
  FourWire = 4
  FiveWire = 5
  SixWire = 6
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
NotSet	未設定の場合の初期値。
ThreeWire	3線。
FourWire	4線。
FiveWire	5線。
SixWire	6線。

備考

格納にはint値のみが使用されます。リポジトリ内に対応するテーブルはありません。

Hbm.Api.SensorDB.Enums.PtSensorWiring Enumeration

PtSensorのワイヤの本数。

C++

```
enum PtSensorWiring : public int {
  NotSet = 0,
  ThreeWire = 3,
  FourWire = 4
};
```

C#

```
public enum PtSensorWiring : int {
  NotSet = 0,
  ThreeWire = 3,
  FourWire = 4
}
```

Visual Basic

```
Public Enum PtSensorWiring
Inherits int
  NotSet = 0
  ThreeWire = 3
  FourWire = 4
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
NotSet	未設定の場合の初期値。
ThreeWire	3線。
FourWire	4線。

備考

格納にはint値のみが使用されます。リポジトリ内に対応するテーブルはありません。

Hbm.Api.SensorDB.Enums.PwmType Enumeration

PWMの種類。

C++

```
enum PwmType : public int {
  DutyCycle = 0,
  Duration = 1,
  Period = 2
};
```

C#

```
public enum PwmType : int {
  DutyCycle = 0,
  Duration = 1,
  Period = 2
}
```

Visual Basic

```
Public Enum PwmType
Inherits int
  DutyCycle = 0
  Duration = 1
  Period = 2
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
DutyCycle	デューティ比(未設定の場合の初期値)。
Duration	期間。
Period	周期。

備考

注意: この列挙体の値は、テーブルPwmTypeのPwmTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.RawValueFormat Enumeration

CAN信号からの生値の形式。

C++

```
enum RawValueFormat : public int {
  UnsignedInt32 = 0,
  SignedInt32 = 1,
  UnsignedInt64 = 2,
  SignedInt64 = 3,
  Float32 = 4,
  Float64 = 5
};
```

C#

```
public enum RawValueFormat : int {
  UnsignedInt32 = 0,
  SignedInt32 = 1,
  UnsignedInt64 = 2,
  SignedInt64 = 3,
  Float32 = 4,
  Float64 = 5
}
```

Visual Basic

```
Public Enum RawValueFormat
Inherits int
  UnsignedInt32 = 0
  SignedInt32 = 1
  UnsignedInt64 = 2
  SignedInt64 = 3
  Float32 = 4
  Float64 = 5
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
UnsignedInt32	符号なし32ビット整数(未設定の場合の初期値)。
SignedInt32	符号付き32ビット整数。
UnsignedInt64	符号なし64ビット整数。
SignedInt64	符号付き64ビット整数。
Float32	32ビット浮動小数点。
Float64	64ビット浮動小数点。

備考

注意: この列挙体の値は、RawValueFormatテーブル内のRawValueFormatIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.ResistanceSensorWiring Enumeration

抵抗センサのワイヤの本数。

C++

```
enum ResistanceSensorWiring : public int {
  NotSet = 0,
  ThreeWire = 3,
  FourWire = 4
};
```

C#

```
public enum ResistanceSensorWiring : int {
  NotSet = 0,
  ThreeWire = 3,
  FourWire = 4
}
```

Visual Basic

```
Public Enum ResistanceSensorWiring
Inherits int
  NotSet = 0
  ThreeWire = 3
  FourWire = 4
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
NotSet	未設定の場合の初期値。
ThreeWire	3線。
FourWire	4線。

備考

格納にはint値のみが使用されます。リポジトリ内に対応するテーブルはありません。

Hbm.Api.SensorDB.Enums.ScalingType Enumeration

センサのスケーリングの種類。

C++

```
enum ScalingType : public int {
  NotSupported = -1,
  TwoPoint = 0,
  ZeroSpan = 1,
  Table = 2,
  Polynomial = 3,
  Gage = 4,
  Internal = 5,
  CanBus = 6,
  Off = 7
};
```

C#

```
public enum ScalingType : int {
  NotSupported = -1,
  TwoPoint = 0,
  ZeroSpan = 1,
  Table = 2,
  Polynomial = 3,
  Gage = 4,
  Internal = 5,
  CanBus = 6,
  Off = 7
}
```

Visual Basic

```
Public Enum ScalingType
Inherits int
  NotSupported = -1
  TwoPoint = 0
  ZeroSpan = 1
  Table = 2
  Polynomial = 3
  Gage = 4
  Internal = 5
  CanBus = 6
  Off = 7
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
NotSupported	スケーリングはサポートしていません。
TwoPoint	2点スケーリング(未設定の場合の初期値)。
ZeroSpan	ゼロスパンスケーリング。
Table	テーブルスケーリング。
Polynomial	多項式スケーリング。
Gage	ゲージ係数スケーリング。
Internal	内部スケーリング。
CanBus	CAN busスケーリング。
Off	スケーリングを使用しません(オフ)。

備考

注意: この列挙体の値は、ScalingTypeテーブル内のScalingTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.SensorType Enumeration

センサの種類。

C++

```
enum SensorType : public int {
  NotSupported = -1,
  Bridge = 0,
  CanBus = 1,
  Counter = 2,
  Current = 3,
  Frequency = 4,
  Iepe = 5,
  Lvdt = 6,
  PiezoPassive = 8,
  Potentiometer = 9,
  Pt = 10,
  Pwm = 11,
  Resistance = 12,
  Ssi = 13,
  ThermoCouple = 14,
  Voltage = 15,
  VoltageProbe = 16,
  CurrentProbe = 17,
  FbgGeneric = 18,
  FbgStrainGage = 19,
  FbgTemperature = 20,
  FbgWavelength = 21,
  FbgAccelerometer = 22
};
```

C#

```
public enum SensorType : int {
  NotSupported = -1,
  Bridge = 0,
  CanBus = 1,
  Counter = 2,
  Current = 3,
  Frequency = 4,
  Iepe = 5,
  Lvdt = 6,
  PiezoPassive = 8,
  Potentiometer = 9,
  Pt = 10,
  Pwm = 11,
  Resistance = 12,
  Ssi = 13,
  ThermoCouple = 14,
  Voltage = 15,
  VoltageProbe = 16,
  CurrentProbe = 17,
  FbgGeneric = 18,
  FbgStrainGage = 19,
  FbgTemperature = 20,
  FbgWavelength = 21,
  FbgAccelerometer = 22
}
```

Visual Basic

```
Public Enum SensorType
Inherits int
  NotSupported = -1
  Bridge = 0
  CanBus = 1
  Counter = 2
  Current = 3
  Frequency = 4
  Iepe = 5
  Lvdt = 6
  PiezoPassive = 8
  Potentiometer = 9
  Pt = 10
  Pwm = 11
  Resistance = 12
  Ssi = 13
  ThermoCouple = 14
  Voltage = 15
  VoltageProbe = 16
  CurrentProbe = 17
  FbgGeneric = 18
  FbgStrainGage = 19
  FbgTemperature = 20
  FbgWavelength = 21
  FbgAccelerometer = 22
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
NotSupported	サポートしていないセンサ。
Bridge	ブリッジセンサ(未設定の場合の初期値)。
CanBus	CAN busセンサ。
Counter	カウンタセンサ。
Current	電流センサ。
Frequency	加速度センサセンサ。
Iepe	IEPEセンサ。
Lvdt	LVDTセンサ。
PiezoPassive	アクティブピエゾセンサおよびパッシブピエゾセンサ。
Potentiometer	ポテンショメータセンサ。
Pt	PTセンサ。
Pwm	PWMセンサ。
Resistance	抵抗温度センサ。
Ssi	SSIセンサ。
ThermoCouple	熱電対。
Voltage	電圧センサ。
VoltageProbe	電圧プローブ。
CurrentProbe	電流プローブ。
FbgGeneric	一般的ファイバブラッググレーティングセンサ。
FbgStrainGage	FBGひずみゲージ。
FbgTemperature	FBG温度センサ。
FbgWavelength	FBG波長センサ。
FbgAccelerometer	FBG加速度センサ。

備考

注意: この列挙体の値は、テーブルSensorTypeのSensorTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.Signal2Type Enumeration

Signal2の種類。

C++

```
enum Signal2Type : public int {
  Off = 0,
  Dynamic = 1,
  Static = 2
};
```

C#

```
public enum Signal2Type : int {
  Off = 0,
  Dynamic = 1,
  Static = 2
}
```

Visual Basic

```
Public Enum Signal2Type
Inherits int
  Off = 0
  Dynamic = 1
  Static = 2
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
Off	F1+F2信号を評価します。
Dynamic	方向は90°位相遅延されたF2信号によって決定されます。
Static	方向は、F2 信号の静的値(高/低)によって決定されます。

備考

注意: この列挙体の値は、Signal2Typeテーブル内のSignal2TypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.SsiCodingType Enumeration

コードの種類。

C++

```
enum SsiCodingType : public int {
  Gray = 0,
  Dual = 1
};
```

C#

```
public enum SsiCodingType : int {
  Gray = 0,
  Dual = 1
}
```

Visual Basic

```
Public Enum SsiCodingType
Inherits int
  Gray = 0
  Dual = 1
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
Gray	グレイコード(未設定の場合の初期値)。
Dual	差動コード。

備考

注意: この列挙体の値は、テーブルCodingTypeのCodingTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Enums.VoltageProbeType Enumeration

VoltageProbeセンサの種類。

C++

```
enum VoltageProbeType : public int {
  PassiveSingleEnded = 0,
  PassiveSingleEndedWithIsolation = 1,
  PassiveSingleEndedWithIsolationXT = 2,
  PassiveDifferential = 3,
  ActiveDifferential = 4,
  CurrentWithVoltageOut = 5
};
```

C#

```
public enum VoltageProbeType : int {
  PassiveSingleEnded = 0,
  PassiveSingleEndedWithIsolation = 1,
  PassiveSingleEndedWithIsolationXT = 2,
  PassiveDifferential = 3,
  ActiveDifferential = 4,
  CurrentWithVoltageOut = 5
}
```

Visual Basic

```
Public Enum VoltageProbeType
Inherits int
  PassiveSingleEnded = 0
  PassiveSingleEndedWithIsolation = 1
  PassiveSingleEndedWithIsolationXT = 2
  PassiveDifferential = 3
  ActiveDifferential = 4
  CurrentWithVoltageOut = 5
End Enum
```

ファイル

SensorEnums.cs

メンバー

メンバー	説明
PassiveSingleEnded	F1+F2信号を評価します。
PassiveSingleEndedWithIsolation	方向は90°位相遅延されたF2信号によって決定されます。
PassiveSingleEndedWithIsolationXT	方向は、F2信号の静的値(高/低)によって決定されます。
PassiveDifferential	PAS_DIFF、パッシブ差動。
ActiveDifferential	ACT_DIFF、アクティブ差動。
CurrentWithVoltageOut	電流、電圧出力付き電流。

備考

注意: この列挙体の値は、テーブルVoltageProbeTypeのVoltageProbeTypeIDと一致する必要があります。

Hbm.Api.SensorDB.Exceptions Namespace

これは名前空間Hbm.Api.SensorDB.Exceptionsです。

クラス

	名前	説明
	AttachmentToLargeException	この例外は、添付ファイルが大きすぎてデータベースに挿入できないことを示します。
	CategoryHasChildsException	この例外は、カテゴリにまだ子要素があることを示します。
	CategoryNotAssignedException	この例外は、センサにカテゴリインスタンスが割り当てられていないことを示します。
	FallbackLanguageDeleteException	この例外は、現在のフォールバック言語を削除しようとしたことを示します。
	InvalidCANFileFormatException	この例外は、CANファイルのファイル形式が無効であることを示します。
	LanguageNotFoundException	この例外は、sensor databaseに言語が見つからなかったことを示します。
	RootCategoryNotFoundException	この例外は、必要なルートカテゴリがデータベースに存在しないことを示します。
	ScalingNotAssignedException	この例外は、センサにスケーリングインスタンスが割り当てられていないことを示します。
	ScalingNotSupportedException	この例外は、データベース操作でサポートされていないスケーリングが使用されたことを示します。
	SensorDBDataException	sensor database APIの一般的な(非具体的な)例外のデータ例外(例:データベースエラーのカプセル化など)。
	SensorDBException	sensor databaseAPIのカスタム例外の基本クラス。
	SensorNotSupportedException	この例外は、データベース操作でサポートしていないセンサを使用したことを示します。

AttachmentToLargeException Class

この例外は、添付ファイルが大きすぎてデータベースに挿入できないことを示します。



C++

```
[Serializable]
class AttachmentToLargeException : public SensorDBException;
```

C#

```
[Serializable]
public class AttachmentToLargeException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class AttachmentToLargeException
Inherits SensorDBException
```

ファイル

AttachmentToLargeException.cs

AttachmentToLargeException Members

次の表は、AttachmentToLargeExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	AttachmentToLargeException	これは、AttachmentToLargeExceptionコンストラクタの多重定義の概要です。
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

AttachmentToLargeException Constructor

これは、AttachmentToLargeExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	AttachmentToLargeException. AttachmentToLargeException ()	新しいインスタンスを作成します。
	AttachmentToLargeException. AttachmentToLargeException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	AttachmentToLargeException. AttachmentToLargeException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

AttachmentToLargeException.AttachmentToLargeException Constructor ()

新しいインスタンスを作成します。

C++

```
AttachmentToLargeException();
```

C#

```
public AttachmentToLargeException();
```

Visual Basic

```
Public Sub New()
```

AttachmentToLargeException.AttachmentToLargeException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
AttachmentToLargeException(
    string message
);
```

C#

```
public AttachmentToLargeException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

AttachmentToLargeException.AttachmentToLargeException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
AttachmentToLargeException(
    string message,
    System.Exception inner
);
```

C#

```
public AttachmentToLargeException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

CategoryHasChildsException Class

この例外は、カテゴリにまだ子要素があることを示します。



C++

```
[Serializable]
class CategoryHasChildsException : public SensorDBException;
```

C#

```
[Serializable]
public class CategoryHasChildsException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class CategoryHasChildsException
Inherits SensorDBException
```

ファイル

CategoryHasChildsException.cs

CategoryHasChildsException Members

次の表は、CategoryHasChildsExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CategoryHasChildsException	これは、CategoryHasChildsExceptionコンストラクタの多重定義の概要です。
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

CategoryHasChildsException Constructor

これは、CategoryHasChildsExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	CategoryHasChildsException. CategoryHasChildsException ()	新しいインスタンスを作成します。
	CategoryHasChildsException. CategoryHasChildsException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	CategoryHasChildsException. CategoryHasChildsException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

CategoryHasChildsException.CategoryHasChildsException Constructor ()

新しいインスタンスを作成します。

C++

```
CategoryHasChildsException();
```

C#

```
public CategoryHasChildsException();
```

Visual Basic

```
Public Sub New()
```

CategoryHasChildsException.CategoryHasChildsException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
CategoryHasChildsException(
    string message
);
```

C#

```
public CategoryHasChildsException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

CategoryHasChildsException.CategoryHasChildsException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
CategoryHasChildsException(
    string message,
    System.Exception inner
);
```

C#

```
public CategoryHasChildsException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

CategoryNotAssignedException Class

この例外は、センサにカテゴリインスタンスが割り当てられていないことを示します。



C++

```
[Serializable]
class CategoryNotAssignedException : public SensorDBException;
```

C#

```
[Serializable]
public class CategoryNotAssignedException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class CategoryNotAssignedException
Inherits SensorDBException
```

ファイル

CategoryNotAssignedException.cs

CategoryNotAssignedException Members

次の表は、CategoryNotAssignedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	CategoryNotAssignedException	これは、CategoryNotAssignedExceptionコンストラクタの多重定義の概要です。
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

CategoryNotAssignedException Constructor

これは、CategoryNotAssignedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	CategoryNotAssignedException. CategoryNotAssignedException ()	新しいインスタンスを作成します。
	CategoryNotAssignedException. CategoryNotAssignedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	CategoryNotAssignedException. CategoryNotAssignedException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

CategoryNotAssignedException.CategoryNotAssignedException Constructor ()

新しいインスタンスを作成します。

C++

```
CategoryNotAssignedException();
```

C#

```
public CategoryNotAssignedException();
```

Visual Basic

```
Public Sub New()
```

CategoryNotAssignedException.CategoryNotAssignedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
CategoryNotAssignedException(
    string message
);
```

C#

```
public CategoryNotAssignedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

CategoryNotAssignedException.CategoryNotAssignedException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
CategoryNotAssignedException(
    string message,
    System.Exception inner
);
```

C#

```
public CategoryNotAssignedException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

FallbackLanguageDeleteException Class

この例外は、現在のフォールバック言語を削除しようとしたことを示します。



C++

```
[Serializable]
class FallbackLanguageDeleteException : public SensorDBException;
```

C#

```
[Serializable]
public class FallbackLanguageDeleteException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class FallbackLanguageDeleteException
Inherits SensorDBException
```

ファイル

FallbackLanguageDeleteException.cs

FallbackLanguageDeleteException Members

次の表は、FallbackLanguageDeleteExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	FallbackLanguageDeleteException	これは、FallbackLanguageDeleteExceptionコンストラクタの多重定義の概要です。
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

FallbackLanguageDeleteException Constructor

これは、FallbackLanguageDeleteExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	FallbackLanguageDeleteException. FallbackLanguageDeleteException ()	新しいインスタンスを作成します。
	FallbackLanguageDeleteException. FallbackLanguageDeleteException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	FallbackLanguageDeleteException. FallbackLanguageDeleteException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

FallbackLanguageDeleteException.FallbackLanguageDeleteException Constructor ()

新しいインスタンスを作成します。

C++

```
FallbackLanguageDeleteException();
```

C#

```
public FallbackLanguageDeleteException();
```

Visual Basic

```
Public Sub New()
```

FallbackLanguageDeleteException.FallbackLanguageDeleteException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
FallbackLanguageDeleteException(
    string message
);
```

C#

```
public FallbackLanguageDeleteException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

FallbackLanguageDeleteException.FallbackLanguageDeleteException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
FallbackLanguageDeleteException(
    string message,
    System.Exception inner
);
```

C#

```
public FallbackLanguageDeleteException(
    string message,
    System.Exception inner
);
```

Visual Basic

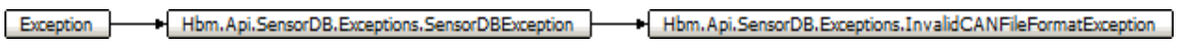
```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

InvalidCANFileFormatException Class

この例外は、CANファイルのファイル形式が無効であることを示します。



C++

```
[Serializable]
class InvalidCANFileFormatException : public SensorDBException;
```

C#

```
[Serializable]
public class InvalidCANFileFormatException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class InvalidCANFileFormatException
Inherits SensorDBException
```

ファイル

InvalidCANFileFormatException.cs

InvalidCANFileFormatException Members

次の表は、InvalidCANFileFormatExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	InvalidCANFileFormatException	これは、InvalidCANFileFormatExceptionコンストラクタの多重定義の概要です。
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

InvalidCANFileFormatException Constructor

これは、InvalidCANFileFormatExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	InvalidCANFileFormatException. InvalidCANFileFormatException ()	新しいインスタンスを作成します。
	InvalidCANFileFormatException. InvalidCANFileFormatException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	InvalidCANFileFormatException. InvalidCANFileFormatException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

InvalidCANFileFormatException.InvalidCANFileFormatException Constructor ()

新しいインスタンスを作成します。

C++

```
InvalidCANFileFormatException();
```

C#

```
public InvalidCANFileFormatException();
```

Visual Basic

```
Public Sub New()
```

InvalidCANFileFormatException.InvalidCANFileFormatException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
InvalidCANFileFormatException(
    string message
);
```

C#

```
public InvalidCANFileFormatException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

InvalidCANFileFormatException.InvalidCANFileFormatException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
InvalidCANFileFormatException(
    string message,
    System.Exception inner
);
```

C#

```
public InvalidCANFileFormatException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

LanguageNotFoundException Class

この例外は、sensor databaseに言語が見つからなかったことを示します。



C++

```
[Serializable]
class LanguageNotFoundException : public SensorDBException;
```

C#

```
[Serializable]
public class LanguageNotFoundException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class LanguageNotFoundException
Inherits SensorDBException
```

ファイル

LanguageNotFoundException.cs

LanguageNotFoundException Members

次の表は、LanguageNotFoundExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	LanguageNotFoundException	これは、LanguageNotFoundExceptionコンストラクタの多重定義の概要です。
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

LanguageNotFoundException Constructor

これは、LanguageNotFoundExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	LanguageNotFoundException. LanguageNotFoundException ()	新しいインスタンスを作成します。
	LanguageNotFoundException. LanguageNotFoundException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	LanguageNotFoundException. LanguageNotFoundException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

LanguageNotFoundException.LanguageNotFoundException Constructor ()

新しいインスタンスを作成します。

C++

```
LanguageNotFoundException();
```

C#

```
public LanguageNotFoundException();
```

Visual Basic

```
Public Sub New()
```

LanguageNotFoundException.LanguageNotFoundException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
LanguageNotFoundException(
    string message
);
```

C#

```
public LanguageNotFoundException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

LanguageNotFoundException.LanguageNotFoundException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
LanguageNotFoundException(
    string message,
    System.Exception inner
);
```

C#

```
public LanguageNotFoundException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

RootCategoryNotFoundException Class

この例外は、必要なルートカテゴリがデータベースに存在しないことを示します。



C++

```
[Serializable]
class RootCategoryNotFoundException : public SensorDBException;
```

C#

```
[Serializable]
public class RootCategoryNotFoundException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class RootCategoryNotFoundException
Inherits SensorDBException
```

ファイル

RootCategoryNotFoundException.cs

RootCategoryNotFoundException Members

次の表は、RootCategoryNotFoundExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	RootCategoryNotFoundException	これは、RootCategoryNotFoundExceptionコンストラクタの多重定義の概要です。
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

RootCategoryNotFoundException Constructor

これは、RootCategoryNotFoundExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	RootCategoryNotFoundException. RootCategoryNotFoundException ()	新しいインスタンスを作成します。
	RootCategoryNotFoundException. RootCategoryNotFoundException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	RootCategoryNotFoundException. RootCategoryNotFoundException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

RootCategoryNotFoundException.RootCategoryNotFoundException Constructor ()

新しいインスタンスを作成します。

C++

```
RootCategoryNotFoundException();
```

C#

```
public RootCategoryNotFoundException();
```

Visual Basic

```
Public Sub New()
```

RootCategoryNotFoundException.RootCategoryNotFoundException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
RootCategoryNotFoundException(
    string message
);
```

C#

```
public RootCategoryNotFoundException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

RootCategoryNotFoundException.RootCategoryNotFoundException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
RootCategoryNotFoundException(
    string message,
    System.Exception inner
);
```

C#

```
public RootCategoryNotFoundException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

ScalingNotAssignedException Class

この例外は、センサにスケーリングインスタンスが割り当てられていないことを示します。



C++

```
[Serializable]
class ScalingNotAssignedException : public SensorDBException;
```

C#

```
[Serializable]
public class ScalingNotAssignedException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class ScalingNotAssignedException
Inherits SensorDBException
```

ファイル

ScalingNotAssignedException.cs

ScalingNotAssignedException Members

次の表は、ScalingNotAssignedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ScalingNotAssignedException	これは、ScalingNotAssignedExceptionコンストラクタの多重定義の概要です。
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

ScalingNotAssignedException Constructor

これは、ScalingNotAssignedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	ScalingNotAssignedException. ScalingNotAssignedException ()	新しいインスタンスを作成します。
	ScalingNotAssignedException. ScalingNotAssignedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	ScalingNotAssignedException. ScalingNotAssignedException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

ScalingNotAssignedException.ScalingNotAssignedException Constructor ()

新しいインスタンスを作成します。

C++

```
ScalingNotAssignedException();
```

C#

```
public ScalingNotAssignedException();
```

Visual Basic

```
Public Sub New()
```

ScalingNotAssignedException.ScalingNotAssignedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
ScalingNotAssignedException(
    string message
);
```

C#

```
public ScalingNotAssignedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

ScalingNotAssignedException.ScalingNotAssignedException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
ScalingNotAssignedException(
    string message,
    System.Exception inner
);
```

C#

```
public ScalingNotAssignedException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

ScalingNotSupportedException Class

この例外は、データベース操作でサポートされていないスケーリングが使用されたことを示します。



C++

```
[Serializable]
class ScalingNotSupportedException : public SensorDBException;
```

C#

```
[Serializable]
public class ScalingNotSupportedException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class ScalingNotSupportedException
Inherits SensorDBException
```

ファイル

ScalingNotSupportedException.cs

ScalingNotSupportedException Members

次の表は、ScalingNotSupportedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ScalingNotSupportedException	これは、ScalingNotSupportedExceptionコンストラクタの多重定義の概要です。
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

ScalingNotSupportedException Constructor

これは、ScalingNotSupportedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	ScalingNotSupportedException. ScalingNotSupportedException ()	新しいインスタンスを作成します。
	ScalingNotSupportedException. ScalingNotSupportedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	ScalingNotSupportedException. ScalingNotSupportedException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

ScalingNotSupportedException.ScalingNotSupportedException Constructor ()

新しいインスタンスを作成します。

C++

```
ScalingNotSupportedException();
```

C#

```
public ScalingNotSupportedException();
```

Visual Basic

```
Public Sub New()
```

ScalingNotSupportedException.ScalingNotSupportedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
ScalingNotSupportedException(
    string message
);
```

C#

```
public ScalingNotSupportedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

ScalingNotSupportedException.ScalingNotSupportedException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
ScalingNotSupportedException(
    string message,
    System.Exception inner
);
```

C#

```
public ScalingNotSupportedException(
    string message,
    System.Exception inner
);
```

Visual Basic

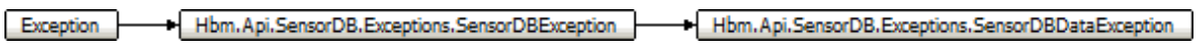
```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

SensorDBDataException Class

sensor database APIの一般的な(非具体的な)例外のデータ例外(例: データベースエラーのカプセル化など)。



C++

```
[Serializable]
class SensorDBDataException : public SensorDBException;
```

C#

```
[Serializable]
public class SensorDBDataException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class SensorDBDataException
Inherits SensorDBException
```

ファイル

SensorDBDataException.cs

SensorDBDataException Members

次の表は、SensorDBDataExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	SensorDBDataException	これは、SensorDBDataExceptionコンストラクタの多重定義の概要です。
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

SensorDBDataException Constructor

これは、SensorDBDataExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	SensorDBDataException. SensorDBDataException ()	新しいインスタンスを作成します。
	SensorDBDataException. SensorDBDataException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	SensorDBDataException. SensorDBDataException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

SensorDBDataException.SensorDBDataException Constructor ()

新しいインスタンスを作成します。

C++

```
SensorDBDataException();
```

C#

```
public SensorDBDataException();
```

Visual Basic

```
Public Sub New()
```

SensorDBDataException.SensorDBDataException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
SensorDBDataException(
    string message
);
```

C#

```
public SensorDBDataException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

SensorDBDataException.SensorDBDataException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
SensorDBDataException(
    string message,
    System.Exception inner
);
```

C#

```
public SensorDBDataException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

SensorDBException Class

sensor databaseAPIのカスタム例外の基本クラス。



C++

```
[Serializable]
class SensorDBException : public Exception;
```

C#

```
[Serializable]
public abstract class SensorDBException : Exception;
```

Visual Basic

```
<Serializable>
Public abstract Class SensorDBException
Inherits Exception
```

ファイル

SensorDBException.cs

SensorDBException Members

次の表は、SensorDBExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。

SensorDBException Constructor

これは、SensorDBExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	SensorDBException. SensorDBException ()	新しいインスタンスを作成します。
	SensorDBException. SensorDBException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	SensorDBException. SensorDBException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

SensorDBException.SensorDBException Constructor ()

新しいインスタンスを作成します。

C++

```
SensorDBException();
```

C#

```
public SensorDBException();
```

Visual Basic

```
Public Sub New()
```

SensorDBException.SensorDBException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
SensorDBException(
    string message
);
```

C#

```
public SensorDBException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

SensorDBException.SensorDBException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
SensorDBException(
    string message,
    System.Exception inner
);
```

C#

```
public SensorDBException(
    string message,
    System.Exception inner
);
```

Visual Basic

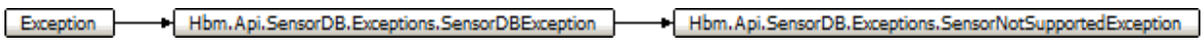
```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

SensorNotSupportedException Class

この例外は、データベース操作でサポートしていないセンサを使用したことを示します。



C++

```
[Serializable]
class SensorNotSupportedException : public SensorDBException;
```

C#

```
[Serializable]
public class SensorNotSupportedException : SensorDBException;
```

Visual Basic

```
<Serializable>
Public Class SensorNotSupportedException
Inherits SensorDBException
```

ファイル

SensorNotSupportedException.cs

現在位置: Hbm.Api.SensorDB Namespace -> Hbm.Api.SensorDB.Exceptions Namespace

SensorNotSupportedException Members

次の表は、SensorNotSupportedExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	SensorDBException	これは、SensorDBExceptionコンストラクタの多重定義の概要です。
	SensorNotSupportedException	これは、SensorNotSupportedExceptionコンストラクタの多重定義の概要です。

SensorNotSupportedException Constructor

これは、SensorNotSupportedExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	SensorNotSupportedException. SensorNotSupportedException ()	新しいインスタンスを作成します。
	SensorNotSupportedException. SensorNotSupportedException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	SensorNotSupportedException. SensorNotSupportedException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

SensorNotSupportedException.SensorNotSupportedException Constructor ()

新しいインスタンスを作成します。

C++

```
SensorNotSupportedException();
```

C#

```
public SensorNotSupportedException();
```

Visual Basic

```
Public Sub New()
```

SensorNotSupportedException.SensorNotSupportedException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
SensorNotSupportedException(
    string message
);
```

C#

```
public SensorNotSupportedException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーのメッセージ。

SensorNotSupportedException.SensorNotSupportedException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
SensorNotSupportedException(
    string message,
    System.Exception inner
);
```

C#

```
public SensorNotSupportedException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーのメッセージ。
inner	エラーの内部例外。

Hbm.Api.Logging Namespace

目次

Hbm.Api.Logging Namespace ・・・

LoggerBase.Error Method (Exception, object) ・・・

LoggerBase.Name Property ・・・

LoggerEmpty.WarnFormat Method (Exception, string, params object[]) 2302
LoggerEmpty.WarnFormat Method (LogContext, string, params object[]) 2303
LoggerEmpty.WarnFormat Method (string, params object[]) 2304
WarnXml Method 2305
LoggerEmpty.WarnXml Method (Exception, string, object) 2306
LoggerEmpty.WarnXml Method (LogContext, Exception, string, object) 2307
LoggerEmpty.WarnXml Method (LogContext, string, object) 2308
LoggerEmpty.WarnXml Method (string, object) 2309
LoggerLoupe Class 2310
LoggerLoupe Members 2311
LoggerLoupe.LoggerLoupe Constructor 2313
LoggerLoupe Classes 2314
LoggerLoupe.LoggerLoupeMessageSourceProvider Class 2315
LoggerLoupeMessageSourceProvider Members 2316
LoggerLoupe.LoggerLoupeMessageSourceProvider.LoggerLoupeMessageSourceProvider Constructor 2317
LoggerLoupeMessageSourceProvider Properties 2318
LoggerLoupe.LoggerLoupeMessageSourceProvider.ClassName Property 2319
LoggerLoupe.LoggerLoupeMessageSourceProvider.FileName Property 2320
LoggerLoupe.LoggerLoupeMessageSourceProvider.LineNumber Property 2321
LoggerLoupe.LoggerLoupeMessageSourceProvider.MethodName Property 2322
LoggerLoupe Methods 2323
Debug Method 2324
LoggerLoupe.Debug Method (Func<string>) 2325
LoggerLoupe.Debug Method (LogContext, Func<string>) 2326
LoggerLoupe.Debug Method (LogContext, object) 2327
LoggerLoupe.Debug Method (object) 2328
DebugFormat Method 2329
LoggerLoupe.DebugFormat Method (LogContext, string, params object[]) 2330
LoggerLoupe.DebugFormat Method (string, params object[]) 2331
DebugXml Method 2332
LoggerLoupe.DebugXml Method (LogContext, string, object) 2333
LoggerLoupe.DebugXml Method (string, object) 2334
Error Method 2335
LoggerLoupe.Error Method (Exception, object) 2336
LoggerLoupe.Error Method (Func<string>) 2337
LoggerLoupe.Error Method (Func<string>, Exception) 2338
LoggerLoupe.Error Method (LogContext, Exception, Func<string>) 2339
LoggerLoupe.Error Method (LogContext, Exception, object) 2340
LoggerLoupe.Error Method (LogContext, Func<string>) 2341
LoggerLoupe.Error Method (LogContext, object) 2342
LoggerLoupe.Error Method (object) 2343
ErrorFormat Method 2344
LoggerLoupe.ErrorFormat Method (Exception, string, params object[]) 2345
LoggerLoupe.ErrorFormat Method (LogContext, Exception, string, params object[]) 2346
LoggerLoupe.ErrorFormat Method (LogContext, string, params object[]) 2347
LoggerLoupe.ErrorFormat Method (string, params object[]) 2348
ErrorXml Method 2349
LoggerLoupe.ErrorXml Method (Exception, string, object) 2350
LoggerLoupe.ErrorXml Method (LogContext, Exception, string, object) 2351
LoggerLoupe.ErrorXml Method (LogContext, string, object) 2352
LoggerLoupe.ErrorXml Method (string, object) 2353
Info Method 2354
LoggerLoupe.Info Method (Func<string>) 2355
LoggerLoupe.Info Method (LogContext, Func<string>) 2356
LoggerLoupe.Info Method (LogContext, object) 2357
LoggerLoupe.Info Method (object) 2358
InfoFormat Method 2359
LoggerLoupe.InfoFormat Method (LogContext, string, params object[]) 2360
LoggerLoupe.InfoFormat Method (string, params object[]) 2361
InfoXml Method 2362
LoggerLoupe.InfoXml Method (LogContext, string, object) 2363
LoggerLoupe.InfoXml Method (string, object) 2364
Trace Method 2365

LoggerLoupe.Trace Method (Func<string>) ・・・

LoggerNlog.Info Method (object) ・・・

ILogger.ErrorXml Method (LogContext, Exception, string, object) ・・・

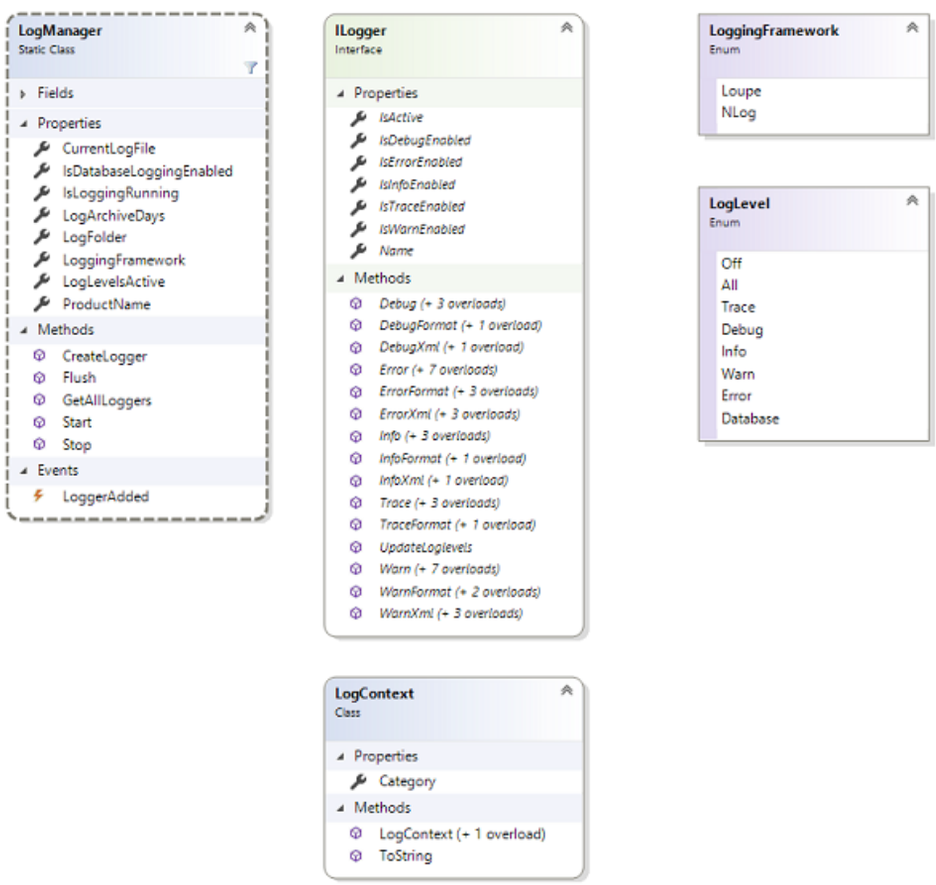
Hbm.Api.Logging名前空間には、さまざまなログフレームワークを使用してメッセージをログに記録できるようにする機能が含まれています(これまではNLogとLoupeをサポートしていました)。

ログエントリをコンテキスト(データベースアクセス、デバイス通信など)に応じてグループ化するには、さまざまなLogContextを定義して、ログ エントリが挿入される階層構造を構築します。

ログ記録の一般的な使用方法は次のとおりです:

- 特定のLoggingFrameworkとLogLevelsを選択してログ記録を開始
- LogManager.CreateLoggerを使用して、少なくとも1つのロガーオブジェクト(Ilogger)を作成
- ログ エントリを分類するために必要なだけLogContextsを作成
- アプリケーション実行中に、特定のロガーLogContextsを使用してログエントリを生成
- アプリケーション終了時にログ記録を停止

次の図は、この名前空間の関連クラスと列挙型の概要を要約したものです。



例

次のコード スニペットは、その方法を示しています

```
// Start logging with a certain LoggingFramework (Loupe or NLog) and a combination of "ORed"
LogLevels:
// e.g.: LogManager.Start(LoggingFramework.Loupe, LogLevel.Error | LogLevel.Warn | LogLevel.
Info );
// e.g.: LogManager.Start(LoggingFramework.Loupe, LogLevel.All);
// e.g.: LogManager.Start(LoggingFramework.NLog, LogLevel.Error | LogLevel.Warn | LogLevel.
Info);
// The directory, in which the logfiles will be saved, can be queried using the LogManager:
// LogManager.LogFolder returns the path that is used to save the logfiles.
// e.g.: C:\Users\Username\AppData\Local\HBM\Logs\Hbm.Api.DemoProject
LogManager.Start(LoggingFramework.NLog, LogLevel.Error | LogLevel.Warn | LogLevel.Info);

// See http://www.gibraltarsoftware.com/ for a free logging viewer for the Loupe logging
framework
// (otherwise you can not read the logfile since Loupe saves the logfile in a binary format)

// NLog logs into a textfile and you do not need a special viewer. LogEntries with low
LogLevels like
// LogLevel.Info or LogLevel.Trace will be buffered until a certain number of entries has
been
// accumulated or an entry with a higher LogLevel (e.g.LogLevel.Error) will be logged.

// Create a logger object that has to be used to log messages
ILogger logger = LogManager.CreateLogger("ApiDemo");

// We want to group our log entries into different parts. Logging uses LogContext objects to
define these groups.
// Notice: The Hbm.CommonAPI will also log messages additional to the messages that will be
logged here!

// E.g.: One group to collect entries concerning measurement tasks:
LogContext logContextApiDemoMeasuring = new LogContext("ApiDemo.Measurement");

// E.g.: One group to collect entries concerning problems:
LogContext logContextApiDemoProblems = new LogContext("ApiDemo.Problems");

// The resulting hierarchy into which the entries can be logged (using Loupe) is now:
// Categories
//  - ApiDemo
// - Measurement
//   - Error: Log entry number 0 for measurement context from ApiDemo!
//   - Info: Log entry number 1 for measurement context from ApiDemo!
//   ...
// - Problems
//   - Debug: Log entry number 2 for problems context from ApiDemo!
//   - Info: Log entry number 3 for problems context from ApiDemo!
//   ...
// using Nlog, these hierarchy infos are included in the line that will be appended to the
logfile
// e.g.: 2015-09-10 09:37:25.1234 [10] ERROR |ApiDemo.Measurement| ApiDemo: Error: Log entry
number 0 for measurement context from ApiDemo!
int logNumberDummy=0; // counter for log entries...

// Generate some log entries (under Measurement)
// this entry will be logged because we started logging with LogManager.
Start(LoggingFramework.NLog, LogLevel.Error | LogLevel.Warn | LogLevel.Info)
logger.ErrorFormat(logContextApiDemoMeasuring, "Error: Log entry number {0} for measurement
context from ApiDemo!", logNumberDummy++);

// this entry will *** N O T *** be logged because we started logging with LogManager.
Start(LoggingFramework.NLog, LogLevel.Error | LogLevel.Warn | LogLevel.Info)
```

```
// change the start command to LogManager.Start(LoggingFramework.NLog, LogLevel.Error |
LogLevel.Warn | LogLevel.Info | LogLevel.Debug) to log the following entry!
logger.DebugFormat(logContextApiDemoMeasuring, "Debug: Log entry number {0} for measurement
context from ApiDemo!", logNumberDummy++);

// Generate log entry (under Problems)
logger.InfoFormat(logContextApiDemoProblems, "Info: Log entry number {0} for problems context
from ApiDemo!", logNumberDummy++);

// End logging.
LogManager.Stop();

// You can find the directory and the created logfile under
Debug.Print("Directory of the logfile: "+LogManager.LogFolder);
Debug.Print("Fullt path to the logfile: "+ LogManager.CurrentLogFile);
```

名前空間

名前	説明
Hbm.Api.Logging	これは、名前空間Hbm.Api.Loggingです。
Hbm.Api.Logging.Enums	これは、名前空間Hbm.Api.Logging.Enumsです。
Hbm.Api.Logging.Logger	これは、名前空間Hbm.Api.Logging.Loggerです。

Hbm.Api.Logging Namespace

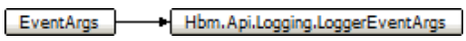
これは名前空間Hbm.Api.Loggingです。

クラス

	名前	説明
	LoggerEventArgs	ILoggerインスタンスを含むイベント引数。
S	LogManager	Common Device APIのログサービスを開始および停止するマネージャクラス。CreateLogger(string)を使用して、ILoggerタイプのロガーオブジェクトをインスタンス化します。

LoggerEventArgs Class

ILoggerインスタンスを含むイベント引数。



C++

```
class LoggerEventArgs : public EventArgs;
```

C#

```
public class LoggerEventArgs : EventArgs;
```

Visual Basic

```
Public Class LoggerEventArgs
Inherits EventArgs
```

ファイル

LoggerEventArgs.cs

関連するトピック

System.EventArgs

LoggerEventArgs Members

次の表は、LoggerEventArgsによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	LoggerEventArgs	LoggerEventArgsクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	Logger	ロガーを取得します。

LoggerEventArgs.LoggerEventArgs Constructor

LoggerEventArgsクラスの新しいインスタンスを初期化します。

C++

```
LoggerEventArgs(
    ILogger logger
);
```

C#

```
public LoggerEventArgs(
    ILogger logger
);
```

Visual Basic

```
Public Sub New(
    logger As ILogger
)
```

引数

引数	説明
logger	ロガー。

LoggerEventArgs Properties

LoggerEventArgsクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Logger	ロガーを取得します。

LoggerEventArgs.Logger Property

ロガーを取得します。

C++

```
__property ILogger Logger;
```

C#

```
public ILogger Logger;
```

Visual Basic

```
Public Property Logger() As ILogger
```

説明

ロガー。

LogManager Class

Common Device APIのログサービスを開始および停止するマネージャクラス。CreateLogger(string)を使用して、ILogger タイプのロガー オブジェクトをインスタンス化します。



C++

```
class LogManager EventArgs;
```

C#

```
public static class LogManager;
```

Visual Basic

```
Public static Class LogManager
```

ファイル

LogManager.cs

備考

最良の方法: 完全修飾型名を使用して、各クラスのロガーを作成します。

LogManager Members

次の表は、LogManagerによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	RaiseLoggerAddedEventAsync	ロガー追加イベントを発生させます。

Public Events

	名前	説明
	LoggerAdded	新しいロガーが追加された時に発生します。

Public Methods

	名前	説明
	CreateLogger	ロガー オブジェクトを作成します。
	Flush	バッファ内の保留中のログメッセージをディスクにフラッシュします。
	GetAllLoggers	すべてのロガーを取得します。
	Start	ログサービスを開始します。ログがすでにアクティブになっている場合(Hbm.Api.Logging.LogManager.IsLoggingRunningも参照)、呼び出しは効果がありません。
	Stop	ログサービスが有効になっている場合は停止します。Hbm.Api.Logging.LogManager.IsLoggingRunningを参照してください。

Public Properties

	名前	説明
	CurrentLogFile	現在のログファイルを絶対ファイルパスとして取得します。
	IsDatabaseLoggingEnabled	データベース操作のロギングが有効かどうかを示すフラグを取得します。
	IsLoggingRunning	ロギングが有効で、使用準備できているかどうかを示すフラグを取得します。
	LogArchiveDays	ログメッセージをアーカイブに格納する最大日数を取得または設定します。
	LogFolder	ログファイルのフォルダの場所を取得します。
	LoggingFramework	使用しているログフレームワークを取得します。
	LogLevelsActive	メッセージが記録されるアクティブなログレベルを取得します。
	ProductName	実行中のアプリケーションのカスタム製品名を取得または設定します。製品名は、ロギングセッションを区別するのに役立ちます。

LogManager Methods

LogManagerクラスのメソッドの一覧です。

Internal Methods

	名前	説明
	RaiseLoggerAddedEventAsync	ロガー追加イベントを発生させます。

Public Methods

	名前	説明
	CreateLogger	ロガー オブジェクトを作成します。
	Flush	バッファ内の保留中のログメッセージをディスクにフラッシュします。
	GetAllLoggers	すべてのロガーを取得します。
	Start	ログサービスを開始します。ログがすでにアクティブになっている場合(Hbm.Api.Logging.LogManager.IsLoggingRunningも参照)、呼び出しは効果がありません。
	Stop	ログサービスが有効になっている場合は停止します。Hbm.Api.Logging.LogManager.IsLoggingRunningを参照してください。

LogManager.CreateLogger Method

ロガーオブジェクトを作成します。

C++

```
ILogger CreateLogger(
    string name
);
```

C#

```
public static ILogger CreateLogger(
    string name
);
```

Visual Basic

```
Public static Function CreateLogger(
    name As string,
) As ILogger
```

引数

引数	説明
name	ロガーの名前(完全修飾クラス名など)。

戻り値

メッセージをログに記録するために使用できるLoggerオブジェクト。Hbm.Api.Logging.Logger.ILoggerを参照してください。Hbm.Api.Logging.LogManager.IsLoggingRunningがfalseの場合でも、空のロガーが返されます。この空のロガーは何も実行しないので、何かをログに記録するときにnullチェックを行う必要はありません。

LogManager.Flush Method

バッファ内の保留中のログメッセージをディスクにフラッシュします。

C++

```
void Flush();
```

C#

```
public static void Flush();
```

Visual Basic

```
Public static Function Flush() As void
```

LogManager.GetAllLoggers Method

すべてのロガーを取得します。

C++

```
List<ILogger> GetAllLoggers();
```

C#

```
public static List<ILogger> GetAllLoggers();
```

Visual Basic

```
Public static Function GetAllLoggers() As List<ILogger>
```

戻り値

すべてのロガーのリスト。

LogManager.RaiseLoggerAddedEventAsync Method

ロガー追加イベントを発生させます。

C++

```
void RaiseLoggerAddedEventAsync(
    ILogger logger
);
```

C#

```
internal static void RaiseLoggerAddedEventAsync(
    ILogger logger
);
```

Visual Basic

```
Friend static Function RaiseLoggerAddedEventAsync(
    logger As ILogger,
) As void
```

引数

引数	説明
logger	ロガー。

LogManager.Start Method

ログサービスを開始します。ログがすでにアクティブになっている場合(Hbm.Api.Logging.LogManager.IsLoggingRunningも参照)、呼び出しは効果がありません。

C++

```
void Start(
    LoggingFramework loggingFramework = LoggingFramework.Loupe,
    LogLevel logLevels = LogLevel.Info | LogLevel.Warn | LogLevel.Error,
    string logFolder = ""
);
```

C#

```
public static void Start(
    LoggingFramework loggingFramework,
    LogLevel logLevels,
    string logFolder
);
```

Visual Basic

```
Public static Function Start(
    loggingFramework As LoggingFramework = LoggingFramework.Loupe,
    logLevels As LogLevel = LogLevel.Info | LogLevel.Warn | LogLevel.Error,
    logFolder As = string = ""
) As void
```

引数

引数	説明
loggingFramework	どのログフレームワークがアプリケーションに接続されているかを決定します。詳細については、hbm.api.logging.enums.loggingframeworkを参照してください。
logLevels	どのレベルメッセージが記録されるかを決定します。hbm.api.logging.enums.loglevelを参照してください。
logFolder	LogManagerが有効なフォルダを生成するためにロギングまたは空に使用する有効なディレクトリ。

備考

hbm.api.logging.enums.loglevel.allが設定されていても、hbm.api.logging.enums.loglevel.offが設定されている場合、ロギングサービスは開始しません。

LogManager.Stop Method

ログサービスが有効になっている場合は停止します。Hbm.Api.Logging.LogManager.IsLoggingRunningを参照してください。

C++

```
void Stop();
```

C#

```
public static void Stop();
```

Visual Basic

```
Public static Function Stop() As void
```

LogManager Properties

LogManagerクラスのプロパティの一覧です。

Public Properties

	名前	説明
R S	CurrentLogFile	現在のログファイルを絶対ファイルパスとして取得します。
S	IsDatabaseLoggingEnabled	データベース操作のロギングが有効かどうかを示すフラグを取得します。
S	IsLoggingRunning	ロギングが有効で、使用準備できているかどうかを示すフラグを取得します。
S	LogArchiveDays	ログメッセージをアーカイブに格納する最大日数を取得または設定します。
R S	LogFolder	ログファイルのフォルダの場所を取得します。
S	LoggingFramework	使用しているログフレームワークを取得します。
S	LogLevelsActive	メッセージが記録されるアクティブなログレベルを取得します。
S	ProductName	実行中のアプリケーションのカスタム製品名を取得または設定します。製品名は、ロギングセッションを区別するのに役立ちます。

LogManager.CurrentLogFile Property

現在のログファイルを絶対ファイルパスとして取得します。

C++

```
__property string CurrentLogFile;
```

C#

```
public static string CurrentLogFile;
```

Visual Basic

```
Public static Property CurrentLogFile() As string
```

LogManager.IsDatabaseLoggingEnabled Property

データベース操作のロギングが有効かどうかを示すフラグを取得します。

C++

```
__property bool IsDatabaseLoggingEnabled;
```

C#

```
public static bool IsDatabaseLoggingEnabled;
```

Visual Basic

```
Public static Property IsDatabaseLoggingEnabled() As bool
```

LogManager.IsLoggingRunning Property

ロギングが有効で、使用準備できているかどうかを示すフラグを取得します。

C++

```
__property bool IsLoggingRunning;
```

C#

```
public static bool IsLoggingRunning;
```

Visual Basic

```
Public static Property IsLoggingRunning() As bool
```

LogManager.LogArchiveDays Property

ログメッセージをアーカイブに格納する最大日数を取得または設定します。

C++

```
__property int LogArchiveDays;
```

C#

```
public static int LogArchiveDays;
```

Visual Basic

```
Public static Property LogArchiveDays() As Integer
```

LogManager.LogFolder Property

ログファイルのフォルダの場所を取得します。

C++

```
__property string LogFolder;
```

C#

```
public static string LogFolder;
```

Visual Basic

```
Public static Property LogFolder() As string
```

LogManager.LoggingFramework Property

使用しているログフレームワークを取得します。

C++

```
__property LoggingFramework LoggingFramework;
```

C#

```
public static LoggingFramework LoggingFramework;
```

Visual Basic

```
Public static Property LoggingFramework() As LoggingFramework
```

LogManager.LogLevelsActive Property

メッセージが記録されるアクティブなログレベルを取得します。

C++

```
__property LogLevel LogLevelsActive;
```

C#

```
public static LogLevel LogLevelsActive;
```

Visual Basic

```
Public static Property LogLevelsActive() As LogLevel
```

LogManager.ProductName Property

実行中のアプリケーションのカスタム製品名を取得または設定します。製品名は、ロギングセッションを区別するのに役立ちます。

C++

```
__property string ProductName;
```

C#

```
public static string ProductName;
```

Visual Basic

```
Public static Property ProductName() As string
```

備考

この値はオプションです。設定しない場合は、実行するアセンブリの製品名を使用します。

例外

例外	説明
System.ArgumentNullException	値がnullまたは空の文字列の場合、スローされます。

LogManager Events

LogManagerクラスのイベントの一覧です。

Public Events

	名前	説明
	LoggerAdded	新しいロガーが追加された時に発生します。

LogManager.LoggerAdded Event

新しいロガーが追加された時に発生します。

C++

```
__property EventHandler<LoggerEventArgs> LoggerAdded;
```

C#

```
public static event EventHandler<LoggerEventArgs> LoggerAdded;
```

Visual Basic

```
Public static Event LoggerAdded() As EventHandler<LoggerEventArgs>
```

Hbm.Api.Logging.Enums Namespace

これは名前空間Hbm.Api.Logging.Enumsです。

列挙型

	名前	説明
	LoggingFramework	サポートしているログフレームワーク。
	LogLevel	利用可能なログレベル。

Hbm.Api.Logging.Enums.LoggingFramework Enumeration

サポートしているログフレームワーク。

C++

```
enum LoggingFramework {
  Loupe = 0,
  NLog = 1
};
```

C#

```
public enum LoggingFramework {
  Loupe = 0,
  NLog = 1
}
```

Visual Basic

```
Public Enum LoggingFramework
  Loupe = 0
  NLog = 1
End Enum
```

ファイル

LoggingFramework.cs

メンバー

メンバー	説明
Loupe	Gibraltar Loupe (http://www.gibraltarsoftware.com/loupe)。
NLog	NLog (http://nlog-project.org/)

現在位置: Hbm.Api.Logging Namespace -> Hbm.Api.Logging.Enums Namespace

Hbm.Api.Logging.Enums.LogLevel Enumeration

利用可能なログレベル。

C++

```
[Flags]
enum LogLevel {
  Off = 0,
  All = 1,
  Trace = 2,
  Debug = 4,
  Info = 8,
  Warn = 16,
  Error = 32,
  Database = 64
};
```

C#

```
[Flags]
public enum LogLevel {
  Off = 0,
  All = 1,
  Trace = 2,
  Debug = 4,
  Info = 8,
  Warn = 16,
  Error = 32,
  Database = 64
}
```

Visual Basic

```
<Flags>
Public Enum LogLevel
  Off = 0
  All = 1
  Trace = 2
  Debug = 4
  Info = 8
  Warn = 16
  Error = 32
  Database = 64
End Enum
```

ファイル

LogLevel.cs

メンバー

メンバー	説明
Off	ログレベルオフ。
All	すべてのログレベル。
Trace	トレースのログレベル。
Debug	デバッグのログレベル。
Info	情報のログレベル。
Warn	警告のログレベル。
Error	エラーのログレベル。
Database	データベース操作のログレベル。

Hbm.Api.Logging.Logger Namespace

これは名前空間Hbm.Api.Logging.Loggerです。

クラス

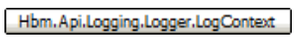
	名前	説明
	LogContext	LogContextは、階層的な方法でログメッセージを分類します。"database.queries.select"など、意味のある分類にする必要があります。
	LoggerBase	すべてのロガーのベースクラス。
	LoggerEmpty	何も記録していないデフォルトのロガーオブジェクト。何らかのログレベルが有効になっているかどうかを決定するすべてのプロパティはfalseです。
	LoggerLoupe	Loupeロガーをラップします。
	LoggerNlog	Nlogロガーをラップします。

インタフェース

	名前	説明
	ILogger	特定のログフレームワーク(NLog、Loupeなど)をカプセル化するロガーオブジェクトのインタフェース。

LogContext Class

LogContextは、階層的な方法でログメッセージを分類します。"database.queries.select"など、意味のある分類にする必要があります。



C++

```
class LogContext;
```

C#

```
public class LogContext;
```

Visual Basic

```
Public Class LogContext
```

ファイル

LogContext.cs

LogContext Members

次の表は、LogContextによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	LogContext	これは、LogContextコンストラクタの多重定義の概要です。
	ToString	Hbm.Api.Logging.Logger.LogContext.Category文字列のプロパティを返します。

Public Properties

	名前	説明
	Category	ログメッセージのログカテゴリ。値はドットで区切る必要があります。例: "Database.Queries.Select"

LogContext Constructor

これは、LogContextコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	LogContext.LogContext ()	Hbm.Api.Logging.Logger.LogContext型のインスタンスを初期化するコンストラクタ。
	LogContext.LogContext (string)	Hbm.Api.Logging.Logger.LogContext型のインスタンスを初期化するコンストラクタ。

LogContext.LogContext Constructor ()

Hbm.Api.Logging.Logger.LogContext型のインスタンスを初期化するコンストラクタ。

C++

```
LogContext();
```

C#

```
public LogContext();
```

Visual Basic

```
Public Sub New()
```

LogContext.LogContext Constructor (string)

Hbm.Api.Logging.Logger.LogContext型のインスタンスを初期化するコンストラクタ。

C++

```
LogContext(
    string category
);
```

C#

```
public LogContext(
    string category
);
```

Visual Basic

```
Public Sub New(
    category As string
)
```

引数

引数	説明
category	ログメッセージのログカテゴリ。値はドットで区切る必要があります。例: "Database.Queries.Select"。

LogContext Methods

LogContextクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ToString	Hbm.Api.Logging.Logger.LogContext.Category文字列のプロパティを返します。

LogContext.ToString Method

Hbm.Api.Logging.Logger.LogContext.Category文字列のプロパティを返します。

C++

```
string ToString();
```

C#

```
public override string ToString();
```

Visual Basic

```
Public override Function ToString() As string
```

LogContext Properties

LogContextクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Category	ログメッセージのログカテゴリ。値はドットで区切る必要があります。例: "Database.Queries.Select"

LogContext.Category Property

ログメッセージのログカテゴリ。値はドットで区切る必要があります。例: "Database.Queries.Select"

C++

```
__property string Category;
```

C#

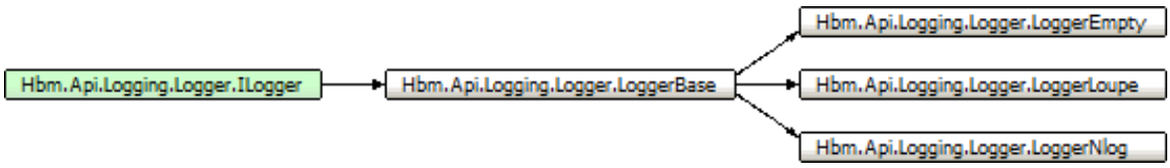
```
public string Category;
```

Visual Basic

```
Public Property Category() As string
```

LoggerBase Class

すべてのロガーのベースクラス。



C++

```
class LoggerBase : public ILogger;
```

C#

```
public abstract class LoggerBase : ILogger;
```

Visual Basic

```
Public abstract Class LoggerBase
Inherits ILogger
```

ファイル

LoggerBase.cs

LoggerBase Members

次の表は、LoggerBaseによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Debug	これは、Debugメソッドの多重定義の概要です。
	Debug	これは、Debugメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	LoggerBase	抽象ロガークラスのコンストラクタ。
	Trace	これは、Traceメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	UpdateLoglevels	現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。
	Warn	これは、Warnメソッドの多重定義の概要です。
	Warn	これは、Warnメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。

Public Properties

	名前	説明
	IsActive	ロガーが有効かを示すフラグ。
	IsDebugEnabled	ログレベルのデバッグが有効かどうかを示すフラグ。
	IsErrorEnabled	ログレベルエラーが有効かどうかを示すフラグ。
	IsInfoEnabled	ログレベル情報が有効かどうかを示すフラグ。
	IsTraceEnabled	ログレベルトレースが有効かどうかを示すフラグ。
	IsWarnEnabled	ログレベルの警告が有効かどうかを示すフラグ。
	Name	名前を取得します。

LoggerBase.LoggerBase Constructor

抽象ロガークラスのコンストラクタ。

C++

```
LoggerBase();
```

C#

```
public LoggerBase();
```

Visual Basic

```
Public Sub New()
```

LoggerBase Methods

LoggerBaseクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Debug	これは、Debugメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	UpdateLoglevels	現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。
	Warn	これは、Warnメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。

Debug Method

これは、Debugメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.Debug (Func<string>)	LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerBase.Debug (LogContext, Func<string>)	LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerBase.Debug (LogContext, object)	LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerBase.Debug (object)	LogLevel.Debugレベルでメッセージをログに記録します。

LoggerBase.Debug Method (Func<string>)

LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Debug(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Debug(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Debug Method (LogContext, Func<string>)

LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Debug(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Debug(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Debug Method (LogContext, object)

LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    LogContext context,
    object message
);
```

C#

```
public abstract void Debug(
    LogContext context,
    object message
);
```

Visual Basic

```
Public abstract Function Debug(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerBase.Debug Method (object)

LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    object message
);
```

C#

```
public abstract void Debug(
    object message
);
```

Visual Basic

```
Public abstract Function Debug(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

DebugFormat Method

これは、DebugFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.DebugFormat (LogContext, string, params object[])	LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerBase.DebugFormat (string, params object[])	LogLevel.Debugレベルでメッセージをログに記録します。

LoggerBase.DebugFormat Method (LogContext, string, params object[])

LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public abstract void DebugFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function DebugFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerBase.DebugFormat Method (string, params object[])

LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugFormat(
    string format,
    params object[] args
);
```

C#

```
public abstract void DebugFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function DebugFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

DebugXml Method

これは、DebugXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.DebugXml (LogContext, string, object)	LogLevel.DebugレベルでXMLメッセージをログに記録します。
	LoggerBase.DebugXml (string, object)	LogLevel.DebugレベルでXML文字列をログに記録します。

LoggerBase.DebugXml Method (LogContext, string, object)

LogLevel.DebugレベルでXMLメッセージをログに記録します。

C++

```
void DebugXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public abstract void DebugXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function DebugXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerBase.DebugXml Method (string, object)

LogLevel.Debugレベルで XML 文字列をログに記録します。

C++

```
void DebugXml(
    string xml,
    object message
);
```

C#

```
public abstract void DebugXml(
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function DebugXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Error Method

これは、Errorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.Error (Exception, object)	LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerBase.Error (Func<string>)	LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerBase.Error (Func<string>, Exception)	LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerBase.Error (LogContext, Exception, Func<string>)	LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerBase.Error (LogContext, Exception, object)	LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerBase.Error (LogContext, Func<string>)	LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerBase.Error (LogContext, object)	LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerBase.Error (object)	LogLevel.Errorレベルでメッセージをログに記録します。

LoggerBase.Error Method (Exception, object)

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Exception ex,
    object message
);
```

C#

```
public abstract void Error(
    Exception ex,
    object message
);
```

Visual Basic

```
Public abstract Function Error(
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerBase.Error Method (Func<string>)

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Error(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Error(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Error Method (Func<string>, Exception)

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

C#

```
public abstract void Error(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

Visual Basic

```
Public abstract Function Error(
    messageBuilderCallback As Func<string>,
    ex As Exception
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。
ex	ログの例外。

LoggerBase.Error Method (LogContext, Exception, Func<string>)

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Error(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Error(
    context As LogContext,
    ex As Exception,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Error Method (LogContext, Exception, object)

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Exception ex,
    object message
);
```

C#

```
public abstract void Error(
    LogContext context,
    Exception ex,
    object message
);
```

Visual Basic

```
Public abstract Function Error(
    context As LogContext,
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerBase.Error Method (LogContext, Func<string>)

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Error(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Error(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Error Method (LogContext, object)

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    object message
);
```

C#

```
public abstract void Error(
    LogContext context,
    object message
);
```

Visual Basic

```
Public abstract Function Error(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerBase.Error Method (object)

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    object message
);
```

C#

```
public abstract void Error(
    object message
);
```

Visual Basic

```
Public abstract Function Error(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

ErrorFormat Method

これは、ErrorFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.ErrorFormat (Exception, string, params object[])	LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerBase.ErrorFormat (LogContext, Exception, string, params object[])	LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerBase.ErrorFormat (LogContext, string, params object[])	LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerBase.ErrorFormat (string, params object[])	LogLevel.Errorレベルでメッセージをログに記録します。

LoggerBase.ErrorFormat Method (Exception, string, params object[])

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public abstract void ErrorFormat(
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function ErrorFormat(
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerBase.ErrorFormat Method (LogContext, Exception, string, params object[])

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    LogContext context,
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public abstract void ErrorFormat(
    LogContext context,
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function ErrorFormat(
    context As LogContext,
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerBase.ErrorFormat Method (LogContext, string, params object[])

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public abstract void ErrorFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function ErrorFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerBase.ErrorFormat Method (string, params object[])

LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    string format,
    params object[] args
);
```

C#

```
public abstract void ErrorFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function ErrorFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ErrorXml Method

これは、ErrorXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.ErrorXml (Exception, string, object)	LogLevel.Errorレベルで XML文字列をログに記録します。
	LoggerBase.ErrorXml (LogContext, Exception, string, object)	LogLevel.Errorレベルで XML文字列をログに記録します。
	LoggerBase.ErrorXml (LogContext, string, object)	LogLevel.Errorレベルで XML文字列をログに記録します。
	LoggerBase.ErrorXml (string, object)	LogLevel.Errorレベルで XML文字列をログに記録します。

LoggerBase.ErrorXml Method (Exception, string, object)

LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    Exception ex,
    string xml,
    object message
);
```

C#

```
public abstract void ErrorXml(
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function ErrorXml(
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerBase.ErrorXml Method (LogContext, Exception, string, object)

LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

C#

```
public abstract void ErrorXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function ErrorXml(
    context As LogContext,
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerBase.ErrorXml Method (LogContext, string, object)

LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public abstract void ErrorXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function ErrorXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerBase.ErrorXml Method (string, object)

LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    string xml,
    object message
);
```

C#

```
public abstract void ErrorXml(
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function ErrorXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

現在位置: Hbm.Api.Logging Namespace -> Hbm.Api.Logging.Logger Namespace ->
LoggerBase Class -> LoggerBase Methods ->
ErrorXml Method -> LoggerBase.ErrorXml Method (string, object)

Info Method

これは、Infoメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.Info (Func<string>)	LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerBase.Info (LogContext, Func<string>)	LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerBase.Info (LogContext, object)	LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerBase.Info (object)	LogLevel.Infoレベルでメッセージをログに記録します。

LoggerBase.Info Method (Func<string>)

LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Info(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Info(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Info Method (LogContext, Func<string>)

LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Info(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Info(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Info Method (LogContext, object)

LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    LogContext context,
    object message
);
```

C#

```
public abstract void Info(
    LogContext context,
    object message
);
```

Visual Basic

```
Public abstract Function Info(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerBase.Info Method (object)

LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    object message
);
```

C#

```
public abstract void Info(
    object message
);
```

Visual Basic

```
Public abstract Function Info(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

InfoFormat Method

これは、InfoFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.InfoFormat (LogContext, string, params object[])	LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerBase.InfoFormat (string, params object[])	LogLevel.Infoレベルでメッセージをログに記録します。

LoggerBase.InfoFormat Method (LogContext, string, params object[])

LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void InfoFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public abstract void InfoFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function InfoFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerBase.InfoFormat Method (string, params object[])

LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void InfoFormat(
    string format,
    params object[] args
);
```

C#

```
public abstract void InfoFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function InfoFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

InfoXml Method

これは、InfoXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.InfoXml (LogContext, string, object)	LogLevel.InfoレベルでXMLメッセージをログに記録します。
	LoggerBase.InfoXml (string, object)	LogLevel.InfoレベルでXML文字列をログに記録します。

LoggerBase.InfoXml Method (LogContext, string, object)

LogLevel.InfoレベルでXMLメッセージをログに記録します。

C++

```
void InfoXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public abstract void InfoXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function InfoXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerBase.InfoXml Method (string, object)

LogLevel.Infoレベルで XML文字列をログに記録します。

C++

```
void InfoXml(
    string xml,
    object message
);
```

C#

```
public abstract void InfoXml(
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function InfoXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Trace Method

これは、Traceメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.Trace (Func<string>)	LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerBase.Trace (LogContext, Func<string>)	LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerBase.Trace (LogContext, object)	LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerBase.Trace (object)	LogLevel.Traceレベルでメッセージをログに記録します。

LoggerBase.Trace Method (Func<string>)

LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Trace(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Trace(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Trace Method (LogContext, Func<string>)

LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Trace(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Trace(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Trace Method (LogContext, object)

LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    LogContext context,
    object  message
);
```

C#

```
public abstract void Trace(
    LogContext context,
    object message
);
```

Visual Basic

```
Public abstract Function Trace(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerBase.Trace Method (object)

LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    object  message
);
```

C#

```
public abstract void Trace(
    object message
);
```

Visual Basic

```
Public abstract Function Trace(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

TraceFormat Method

これは、TraceFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.TraceFormat (LogContext, string, params object[])	LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerBase.TraceFormat (string, params object[])	LogLevel.Traceレベルでメッセージをログに記録します。

LoggerBase.TraceFormat Method (LogContext, string, params object[])

LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void TraceFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public abstract void TraceFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function TraceFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerBase.TraceFormat Method (string, params object[])

LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void TraceFormat(
    string format,
    params object[] args
);
```

C#

```
public abstract void TraceFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function TraceFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerBase.UpdateLoglevels Method

現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。

C++

```
void UpdateLoglevels();
```

C#

```
public void UpdateLoglevels();
```

Visual Basic

```
Public Function UpdateLoglevels() As void
```

Warn Method

これは、Warnメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.Warn (Exception, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerBase.Warn (Func<string>)	LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerBase.Warn (Func<string>, Exception)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerBase.Warn (LogContext, Exception, Func<string>)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerBase.Warn (LogContext, Exception, object)	LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerBase.Warn (LogContext, Func<string>)	LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerBase.Warn (LogContext, object)	LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerBase.Warn (object)	LogLevel.Warnレベルでメッセージをログに記録します。

LoggerBase.Warn Method (Exception, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Exception ex,
    object message
);
```

C#

```
public abstract void Warn(
    Exception ex,
    object message
);
```

Visual Basic

```
Public abstract Function Warn(
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerBase.Warn Method (Func<string>)

LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Warn(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Warn(
    messageBuilderCallback As Func<string>,
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Warn Method (Func<string>, Exception)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

C#

```
public abstract void Warn(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

Visual Basic

```
Public abstract Function Warn(
    messageBuilderCallback As Func<string>,
    ex As Exception
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。
ex	ログの例外。

LoggerBase.Warn Method (LogContext, Exception, Func<string>)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Warn(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Warn(
    context As LogContext,
    ex As Exception,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Warn Method (LogContext, Exception, object)

LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Exception ex,
    object message
);
```

C#

```
public abstract void Warn(
    LogContext context,
    Exception ex,
    object message
);
```

Visual Basic

```
Public abstract Function Warn(
    context As LogContext,
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
message	ログのメッセージオブジェクト。

現在位置: Hbm.Api.Logging Namespace -> Hbm.Api.Logging.Logger Namespace -> LoggerBase Class -> LoggerBase Methods -> LoggerBase.UpdateLoglevels Method

LoggerBase.Warn Method (LogContext, Func<string>)

LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public abstract void Warn(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public abstract Function Warn(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerBase.Warn Method (LogContext, object)

LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    object message
);
```

C#

```
public abstract void Warn(
    LogContext context,
    object message
);
```

Visual Basic

```
Public abstract Function Warn(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerBase.Warn Method (object)

LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    object message
);
```

C#

```
public abstract void Warn(
    object message
);
```

Visual Basic

```
Public abstract Function Warn(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

WarnFormat Method

これは、WarnFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.WarnFormat (Exception, string, params object[])	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerBase.WarnFormat (LogContext, string, params object[])	LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerBase.WarnFormat (string, params object[])	LogLevel.Warnレベルでメッセージをログに記録します。

LoggerBase.WarnFormat Method (Exception, string, params object[])

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public abstract void WarnFormat(
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function WarnFormat(
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerBase.WarnFormat Method (LogContext, string, params object[])

LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public abstract void WarnFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function WarnFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerBase.WarnFormat Method (string, params object[])

LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    string format,
    params object[] args
);
```

C#

```
public abstract void WarnFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public abstract Function WarnFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

WarnXml Method

これは、WarnXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerBase.WarnXml (Exception, string, object)	Enums.LogLevel.Warnレベルで XML文字列をログに記録します。
	LoggerBase.WarnXml (LogContext, Exception, string, object)	Enums.LogLevel.Warnレベルで XMLメッセージをログに記録します。
	LoggerBase.WarnXml (LogContext, string, object)	LogLevel.Warnレベルで XMLメッセージをログに記録します。
	LoggerBase.WarnXml (string, object)	LogLevel.Warnレベルで XML文字列をログに記録します。

LoggerBase.WarnXml Method (Exception, string, object)

Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

C++

```
void WarnXml(
    Exception ex,
    string xml,
    object message
);
```

C#

```
public abstract void WarnXml(
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function WarnXml(
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerBase.WarnXml Method (LogContext, Exception, string, object)

Enums.LogLevel.WarnレベルでXML文字列をログに記録します。

C++

```
void WarnXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

C#

```
public abstract void WarnXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function WarnXml(
    context As LogContext,
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerBase.WarnXml Method (LogContext, string, object)

LogLevel.Warnレベルで XMLメッセージをログに記録します。

C++

```
void WarnXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public abstract void WarnXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function WarnXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerBase.WarnXml Method (string, object)

LogLevel.WarnレベルでXML文字列をログに記録します。

C++

```
void WarnXml(
    string xml,
    object message
);
```

C#

```
public abstract void WarnXml(
    string xml,
    object message
);
```

Visual Basic

```
Public abstract Function WarnXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

現在位置: Hbm.Api.Logging Namespace -> Hbm.Api.Logging.Logger Namespace -> LoggerBase Class -> LoggerBase Methods -> WarnXml Method

LoggerBase Properties

LoggerBaseクラスのプロパティの一覧です。

Public Properties

	名前	説明
	IsActive	ロガーが有効かを示すフラグ。
	LoggerBase.IsActive Property	ログレベルのデバッグが有効かどうかを示すフラグ。
	IsErrorEnabled	ログレベルエラーが有効かどうかを示すフラグ。
	IsInfoEnabled	ログレベル情報が有効かどうかを示すフラグ。
	IsTraceEnabled	ログレベルトレースが有効かどうかを示すフラグ。
	IsWarnEnabled	ログレベルの警告が有効かどうかを示すフラグ。
	Name	名前を取得します。

LoggerBase.IsActive Property

ロガーが有効かどうかを示すフラグ。

C++

```
__property bool IsActive;
```

C#

```
public bool IsActive;
```

Visual Basic

```
Public Property IsActive() As bool
```

LoggerBase.IsDebugEnabled Property

ログレベルのデバッグが有効かどうかを示すフラグ。

C++

```
__property bool IsDebugEnabled;
```

C#

```
public bool IsDebugEnabled;
```

Visual Basic

```
Public Property IsDebugEnabled() As bool
```

LoggerBase.IsErrorEnabled Property

ログレベルエラーが有効かどうかを示すフラグ。

C++

```
__property bool IsErrorEnabled;
```

C#

```
public bool IsErrorEnabled;
```

Visual Basic

```
Public Property IsErrorEnabled() As bool
```

LoggerBase.IsInfoEnabled Property

ログレベル情報が有効かどうかを示すフラグ。

C++

```
__property bool IsInfoEnabled;
```

C#

```
public bool IsInfoEnabled;
```

Visual Basic

```
Public Property IsInfoEnabled() As bool
```

LoggerBase.IsTraceEnabled Property

ログレベルトレースが有効かどうかを示すフラグ。

C++

```
__property bool IsTraceEnabled;
```

C#

```
public bool IsTraceEnabled;
```

Visual Basic

```
Public Property IsTraceEnabled() As bool
```

LoggerBase.IsWarnEnabled Property

ログレベルの警告が有効かどうかを示すフラグ。

C++

```
__property bool IsWarnEnabled;
```

C#

```
public bool IsWarnEnabled;
```

Visual Basic

```
Public Property IsWarnEnabled() As bool
```

LoggerBase.Name Property

名前を取得します。

C++

```
__property bool Name;
```

C#

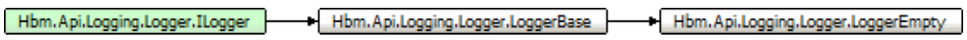
```
public bool Name;
```

Visual Basic

```
Public Property Name() As bool
```

LoggerEmpty Class

何も記録していないデフォルトのロガーオブジェクト。何らかのログレベルが有効になっているかどうかを決定するすべてのプロパティはfalseです。



C++

```
class LoggerEmpty : public LoggerBase;
```

C#

```
internal class LoggerEmpty : LoggerBase;
```

Visual Basic

```
Friend Class LoggerEmpty
Inherits LoggerBase
```

ファイル

LoggerEmpty.cs

LoggerEmpty Members

次の表は、LoggerEmptyによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Debug	これは、Debugメソッドの多重定義の概要です。
	Debug	これは、Debugメソッドの多重定義の概要です。
	Debug	これは、Debugメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	UpdateLoglevels	現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。
	UpdateLoglevels	現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。
	Warn	これは、Warnメソッドの多重定義の概要です。
	Warn	これは、Warnメソッドの多重定義の概要です。
	Warn	これは、Warnメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。

Public Properties

	名前	説明
	IsActive	ロガーが有効かを示すフラグ。
	IsActive	ロガーが有効かを示すフラグ。
	IsDebugEnabled	ログレベルのデバッグが有効かどうかを示すフラグ。
	IsDebugEnabled	ログレベルのデバッグが有効かどうかを示すフラグ。
	IsErrorEnabled	ログレベルエラーが有効かどうかを示すフラグ。
	IsErrorEnabled	ログレベルエラーが有効かどうかを示すフラグ。
	IsInfoEnabled	ログレベル情報が有効かどうかを示すフラグ。
	IsInfoEnabled	ログレベル情報が有効かどうかを示すフラグ。
	IsTraceEnabled	ログレベルトレースが有効かどうかを示すフラグ。
	IsTraceEnabled	ログレベルトレースが有効かどうかを示すフラグ。
	IsWarnEnabled	ログレベルの警告が有効かどうかを示すフラグ。
	IsWarnEnabled	ログレベルの警告が有効かどうかを示すフラグ。
	Name	名前を取得します。
	Name	名前を取得します。

LoggerEmpty Methods

LoggerEmptyクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Debug	これは、Debugメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	Warn	これは、Warnメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。

Debug Method

これは、Debugメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.Debug (Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerEmpty.Debug (LogContext, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerEmpty.Debug (LogContext, object)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerEmpty.Debug (object)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

LoggerEmpty.Debug Method (Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Debug(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Debug(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Debug Method (LogContext, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Debug(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Debug(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Debug Method (LogContext, object)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    LogContext context,
    object message
);
```

C#

```
public override void Debug(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Debug(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerEmpty.Debug Method (object)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    object message
);
```

C#

```
public override void Debug(
    object message
);
```

Visual Basic

```
Public override Function Debug(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

DebugFormat Method

これは、DebugFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.DebugFormat (LogContext, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerEmpty.DebugFormat (string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

LoggerEmpty.DebugFormat Method (LogContext, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void DebugFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function DebugFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerEmpty.DebugFormat Method (string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugFormat(
    string format,
    params object[] args
);
```

C#

```
public override void DebugFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function DebugFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

DebugXml Method

これは、DebugXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.DebugXml (LogContext, string, object)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルで XMLメッセージをログに記録します。
	LoggerEmpty.DebugXml (string, object)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルで XML文字列をログに記録します。

LoggerEmpty.DebugXml Method (LogContext, string, object)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルで XMLメッセージをログに記録します。

C++

```
void DebugXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void DebugXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function DebugXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerEmpty.DebugXml Method (string, object)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルで XML 文字列をログに記録します。

C++

```
void DebugXml(
    string xml,
    object message
);
```

C#

```
public override void DebugXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function DebugXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Error Method

これは、Errorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.Error (Exception, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerEmpty.Error (Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerEmpty.Error (Func<string>, Exception)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerEmpty.Error (LogContext, Exception, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerEmpty.Error (LogContext, Exception, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerEmpty.Error (LogContext, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerEmpty.Error (LogContext, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerEmpty.Error (object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

LoggerEmpty.Error Method (Exception, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Exception ex,
    object message
);
```

C#

```
public override void Error(
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Error(
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerEmpty.Error Method (Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Error(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Error(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Error Method (Func<string>, Exception)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

C#

```
public override void Error(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

Visual Basic

```
Public override Function Error(
    messageBuilderCallback As Func<string>,
    ex As Exception
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。
ex	ログの例外。

LoggerEmpty.Error Method (LogContext, Exception, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Error(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    ex As Exception,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Error Method (LogContext, Exception, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Exception ex,
    object message
);
```

C#

```
public override void Error(
    LogContext context,
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerEmpty.

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Error(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Error Method (LogContext, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    object message
);
```

C#

```
public override void Error(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerEmpty.Error Method (object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    object message
);
```

C#

```
public override void Error(
    object message
);
```

Visual Basic

```
Public override Function Error(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

ErrorFormat Method

これは、ErrorFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.ErrorFormat (Exception, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerEmpty.ErrorFormat (LogContext, Exception, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerEmpty.ErrorFormat (LogContext, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerEmpty.ErrorFormat (string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

LoggerEmpty.ErrorFormat Method (Exception, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerEmpty.ErrorFormat Method (LogContext, Exception, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    LogContext context,
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    LogContext context,
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    context As LogContext,
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerEmpty.ErrorFormat Method (LogContext, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerEmpty.ErrorFormat Method (string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ErrorXml Method

これは、ErrorXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.ErrorXml (Exception, string, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。
	LoggerEmpty.ErrorXml (LogContext, Exception, string, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。
	LoggerEmpty.ErrorXml (LogContext, string, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。
	LoggerEmpty.ErrorXml (string, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

LoggerEmpty.ErrorXml Method (Exception, string, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerEmpty.ErrorXml Method (LogContext, Exception, string, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    context As LogContext,
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerEmpty.ErrorXml Method (LogContext, string, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerEmpty.ErrorXml Method (string, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Info Method

これは、Infoメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.Info (Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerEmpty.Info (LogContext, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerEmpty.Info (LogContext, object)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerEmpty.Info (object)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

LoggerEmpty.Info Method (Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Info(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Info(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Info Method (LogContext, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Info(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Info(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Info Method (LogContext, object)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    LogContext context,
    object message
);
```

C#

```
public override void Info(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Info(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerEmpty.Info Method (object)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    object message
);
```

C#

```
public override void Info(
    object message
);
```

Visual Basic

```
Public override Function Info(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

InfoFormat Method

これは、InfoFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.InfoFormat (LogContext, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerEmpty.InfoFormat (string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

LoggerEmpty.InfoFormat Method (LogContext, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void InfoFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void InfoFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function InfoFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerEmpty.InfoFormat Method (string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void InfoFormat(
    string format,
    params object[] args
);
```

C#

```
public override void InfoFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function InfoFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

InfoXml Method

これは、InfoXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.InfoXml (LogContext, string, object)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルで XMLメッセージをログに記録します。
	LoggerEmpty.InfoXml (string, object)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルで XML文字列をログに記録します。

LoggerEmpty.InfoXml Method (LogContext, string, object)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルで XMLメッセージをログに記録します。

C++

```
void InfoXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void InfoXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function InfoXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerEmpty.InfoXml Method (string, object)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルで XML文字列をログに記録します。

C++

```
void InfoXml(
    string xml,
    object message
);
```

C#

```
public override void InfoXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function InfoXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Trace Method

これは、Traceメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.Trace (Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerEmpty.Trace (LogContext, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerEmpty.Trace (LogContext, object)	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerEmpty.Trace (object)	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

LoggerEmpty.Trace Method (Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Trace(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Trace(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Trace Method (LogContext, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Trace(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Trace(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Trace Method (LogContext, object)

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    LogContext context,
    object  message
);
```

C#

```
public override void Trace(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Trace(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerEmpty.Trace Method (object)

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    object  message
);
```

C#

```
public override void Trace(
    object message
);
```

Visual Basic

```
Public override Function Trace(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

TraceFormat Method

これは、TraceFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.TraceFormat (LogContext, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerEmpty.TraceFormat (string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

LoggerEmpty.TraceFormat Method (LogContext, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void TraceFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void TraceFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function TraceFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerEmpty.TraceFormat Method (string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void TraceFormat(
    string format,
    params object[] args
);
```

C#

```
public override void TraceFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function TraceFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

Warn Method

これは、Warnメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.Warn (Exception, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerEmpty.Warn (Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerEmpty.Warn (Func<string>, Exception)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerEmpty.Warn (LogContext, Exception, Func<string>)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerEmpty.Warn (LogContext, Exception, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録しま
	LoggerEmpty.Warn (LogContext, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerEmpty.Warn (LogContext, object)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerEmpty.Warn (object)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

LoggerEmpty.Warn Method (Exception, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Exception ex,
    object message
);
```

C#

```
public override void Warn(
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Warn(
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerEmpty.Warn Method (Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Warn(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Warn(
    messageBuilderCallback As Func<string>,
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Warn Method (Func<string>, Exception)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

C#

```
public override void Warn(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

Visual Basic

```
Public override Function Warn(
    messageBuilderCallback As Func<string>,
    ex As Exception
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。
ex	ログの例外。

LoggerEmpty.Warn Method (LogContext, Exception, Func<string>)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Warn(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    ex As Exception,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Warn Method (LogContext, Exception, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Exception ex,
    object message
);
```

C#

```
public override void Warn(
    LogContext context,
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerEmpty.Warn Method (LogContext, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Warn(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerEmpty.Warn Method (LogContext, object)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    object message
);
```

C#

```
public override void Warn(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerEmpty.Warn Method (object)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    object message
);
```

C#

```
public override void Warn(
    object message
);
```

Visual Basic

```
Public override Function Warn(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

WarnFormat Method

これは、WarnFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.WarnFormat (Exception, string, params object[])	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerEmpty.WarnFormat (LogContext, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerEmpty.WarnFormat (string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

LoggerEmpty.WarnFormat Method (Exception, string, params object[])

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public override void WarnFormat(
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function WarnFormat(
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerEmpty.WarnFormat Method (LogContext, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void WarnFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function WarnFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerEmpty.WarnFormat Method (string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    string format,
    params object[] args
);
```

C#

```
public override void WarnFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function WarnFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

WarnXml Method

これは、WarnXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerEmpty.WarnXml (Exception, string, object)	Enums.LogLevel.Warnレベルで XML文字列をログに記録します。
	LoggerEmpty.WarnXml (LogContext, Exception, string, object)	Enums.LogLevel.Warnレベルで XMLメッセージをログに記録します。
	LoggerEmpty.WarnXml (LogContext, string, object)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルで XMLメッセージをログに記録します。
	LoggerEmpty.WarnXml (string, object)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

LoggerEmpty.WarnXml Method (Exception, string, object)

Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

C++

```
void WarnXml(
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerEmpty.WarnXml Method (LogContext, Exception, string, object)

Enums.LogLevel.Warnレベルで XMLメッセージをログに記録します。

C++

```
void WarnXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    context As LogContext,
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerEmpty.WarnXml Method (LogContext, string, object)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルで XMLメッセージをログに記録します。

C++

```
void WarnXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerEmpty.WarnXml Method (string, object)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

C++

```
void WarnXml(
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    string xml,
    object message
);
```

Visual Basic

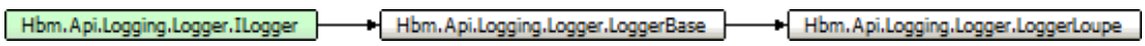
```
Public override Function WarnXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerLoupe Class

Loupeロガーをラップします。



C++

```
class LoggerLoupe  : public LoggerBase;
```

C#

```
internal class LoggerLoupe  : LoggerBase;
```

Visual Basic

```
Friend Class LoggerLoupe
Inherits LoggerBase
```

ファイル

LoggerLoupe.cs

現在位置: Hbm.Api.Logging Namespace -> Hbm.Api.Logging.Logger Namespace

LoggerLoupe Members

次の表は、LoggerLoupeによって公開されるメンバーの一覧です。

クラス

名前	説明
LoggerLoupeMessageSourceProvider	このクラスは、Gibraltarのソース プロパティを提供します。これにより、たとえばLoupe Agentでクラス名別にグループ化できるようになります。

Public Methods

	名前	説明
	Debug	これは、Debugメソッドの多重定義の概要です。
	Debug	これは、Debugメソッドの多重定義の概要です。
	Debug	これは、Debugメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	LoggerLoupe	新しいインスタンスを作成し、データを初期化します。
	Trace	これは、Traceメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	UpdateLoglevels	現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。
	UpdateLoglevels	現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。
	Warn	これは、Warnメソッドの多重定義の概要です。
	Warn	これは、Warnメソッドの多重定義の概要です。
	Warn	これは、Warnメソッドの多重定義の概要です。

	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。

Public Properties

	名前	説明
	IsActive	ロガーが有効かを示すフラグ。
	IsActive	ロガーが有効かを示すフラグ。
	IsDebugEnabled	ログレベルのデバッグが有効かどうかを示すフラグ。
	IsDebugEnabled	ログレベルのデバッグが有効かどうかを示すフラグ。
	IsErrorEnabled	ログレベルエラーが有効かどうかを示すフラグ。
	IsErrorEnabled	ログレベルエラーが有効かどうかを示すフラグ。
	IsInfoEnabled	ログレベル情報が有効かどうかを示すフラグ。
	IsInfoEnabled	ログレベル情報が有効かどうかを示すフラグ。
	IsTraceEnabled	ログレベルトレースが有効かどうかを示すフラグ。
	IsTraceEnabled	ログレベルトレースが有効かどうかを示すフラグ。
	IsWarnEnabled	ログレベルの警告が有効かどうかを示すフラグ。
	IsWarnEnabled	ログレベルの警告が有効かどうかを示すフラグ。
	Name	名前を取得します。
	Name	名前を取得します。

LoggerLoupe.LoggerLoupe Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
LoggerLoupe(
    string name
);
```

C#

```
public LoggerLoupe(
    string name
);
```

Visual Basic

```
Public Sub New(
    name As string
)
```

引数

引数	説明
name	アクティブなLoupeロガーの名前。

LoggerLoupe Classes

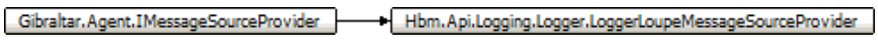
LoggerLoupeクラスのクラスの一覧です。

クラス

名前	説明
LoggerLoupeMessageSourceProvider	このクラスは、Gibraltarのソース プロパティを提供します。これにより、たとえばLoupe Agentでクラス名別にグループ化できるようになります。

LoggerLoupe.LoggerLoupeMessageSourceProvider Class

このクラスは、Gibraltarのソース プロパティを提供します。これにより、たとえばLoupe Agentでクラス名別にグループ化できるようになります。



C++

```
class LoggerLoupeMessageSourceProvider : public Gibraltar.Agent.IMessageSourceProvider;
```

C#

```
internal class LoggerLoupeMessageSourceProvider : Gibraltar.Agent.IMessageSourceProvider;
```

Visual Basic

```
Friend Class LoggerLoupeMessageSourceProvider
Inherits Gibraltar.Agent.IMessageSourceProvider
```

ファイル

LoggerLoupe.cs

関連するトピック

System.EventArgs

LoggerLoupeMessageSourceProvider Members

次の表は、LoggerLoupeMessageSourceProviderによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	LoggerLoupeMessageSourceProvider	新しいインスタンスを作成し、データを初期化します。

Public Properties

	名前	説明
	ClassName	ロガーのクラス名。
R	FileName	常にnullです。現時点では必要ありません。
R	LineNumber	常に-1です。現時点では必要ありません。
R	MethodName	常にnullです。現時点では必要ありません。

LoggerLoupe.LoggerLoupeMessageSourceProvider.LoggerLoupeMessageSourceProvider Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
LoggerLoupeMessageSourceProvider(
    string className
);
```

C#

```
public LoggerLoupeMessageSourceProvider(
    string className
);
```

Visual Basic

```
Public Sub New(
    className As string
)
```

引数

引数	説明
className	ロガーのクラス名。

LoggerLoupeMessageSourceProvider Properties

LoggerLoupeMessageSourceProviderクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ClassName	ロガーのクラス名。
R	FileName	常にnullです。現時点では必要ありません。
R	LineNumber	常に-1です。現時点では必要ありません。
R	MethodName	常にnullです。現時点では必要ありません。

LoggerLoupe.LoggerLoupeMessageSourceProvider.ClassName Property

ロガーのクラス名。

C++

```
__property string ClassName;
```

C#

```
public string ClassName;
```

Visual Basic

```
Public Property ClassName() As string
```

LoggerLoupe.LoggerLoupeMessageSourceProvider.FileName Property

常にnullです。現時点では必要ありません。

C++

```
__property string FileName;
```

C#

```
public string FileName;
```

Visual Basic

```
Public Property FileName() As string
```

LoggerLoupe.LoggerLoupeMessageSourceProvider.LineNumber Property

常に-1です。現時点では必要ありません。

C++

```
__property int LineNumber;
```

C#

```
public int LineNumber;
```

Visual Basic

```
Public Property LineNumber() As Integer
```

LoggerLoupe.LoggerLoupeMessageSourceProvider.MethodName Property

常にnullです。現時点では必要ありません。

C++

```
__property string MethodName;
```

C#

```
public string MethodName;
```

Visual Basic

```
Public Property MethodName() As string
```

LoggerLoupe Methods

LoggerLoupeクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Debug	これは、Debugメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	Warn	これは、Warnメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。

Debug Method

これは、Debugメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.Debug (Func<string>)	Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerLoupe.Debug (LogContext, Func<string>)	Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerLoupe.Debug (LogContext, object)	Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerLoupe.Debug (object)	Enums.LogLevel.Debugレベルでメッセージをログに記録します。

LoggerLoupe.Debug Method (Func<string>)

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Debug(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Debug(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Debug Method (LogContext, Func<string>)

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Debug(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Debug(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Debug Method (LogContext, object)

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    LogContext context,
    object message
);
```

C#

```
public override void Debug(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Debug(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerLoupe.Debug Method (object)

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    object message
);
```

C#

```
public override void Debug(
    object message
);
```

Visual Basic

```
Public override Function Debug(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

DebugFormat Method

これは、DebugFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.DebugFormat (LogContext, string, params object[])	Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerLoupe.DebugFormat (string, params object[])	Enums.LogLevel.Debugレベルでメッセージをログに記録します。

LoggerLoupe.DebugFormat Method (LogContext, string, params object[])

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void DebugFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function DebugFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerLoupe.DebugFormat Method (string, params object[])

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugFormat(
    string format,
    params object[] args
);
```

C#

```
public override void DebugFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function DebugFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

DebugXml Method

これは、DebugXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.DebugXml (LogContext, string, object)	Enums.LogLevel.DebugレベルでXMLメッセージをログに記録します。
	LoggerLoupe.DebugXml (string, object)	Enums.LogLevel.DebugレベルでXMLメッセージをログに記録します。

LoggerLoupe.DebugXml Method (LogContext, string, object)

Enums.LogLevel.DebugレベルでXMLメッセージをログに記録します。

C++

```
void DebugXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void DebugXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function DebugXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerLoupe.DebugXml Method (string, object)

Enums.LogLevel.Debugレベルで XMLメッセージをログに記録します。

C++

```
void DebugXml(
    string xml,
    object message
);
```

C#

```
public override void DebugXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function DebugXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Error Method

これは、Errorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.Error (Exception, object)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerLoupe.Error (Func<string>)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerLoupe.Error (Func<string>, Exception)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerLoupe.Error (LogContext, Exception, Func<string>)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerLoupe.Error (LogContext, Exception, object)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerLoupe.Error (LogContext, Func<string>)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerLoupe.Error (LogContext, object)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerLoupe.Error (object)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。

LoggerLoupe.Error Method (Exception, object)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Exception ex,
    object message
);
```

C#

```
public override void Error(
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Error(
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerLoupe.Error Method (Func<string>)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Error(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Error(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Error Method (Func<string>, Exception)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

C#

```
public override void Error(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

Visual Basic

```
Public override Function Error(
    messageBuilderCallback As Func<string>,
    ex As Exception
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。
ex	ログの例外。

LoggerLoupe.Error Method (LogContext, Exception, Func<string>)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Error(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    ex As Exception,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Error Method (LogContext, Exception, object)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Exception ex,
    object message
);
```

C#

```
public override void Error(
    LogContext context,
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerLoupe.Error Method (LogContext, Func<string>)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Error(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Error Method (LogContext, object)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    object message
);
```

C#

```
public override void Error(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerLoupe.Error Method (object)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    object message
);
```

C#

```
public override void Error(
    object message
);
```

Visual Basic

```
Public override Function Error(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

ErrorFormat Method

これは、ErrorFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.ErrorFormat (Exception, string, params object[])	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerLoupe.ErrorFormat (LogContext, Exception, string, params object[])	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerLoupe.ErrorFormat (LogContext, string, params object[])	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerLoupe.ErrorFormat (string, params object[])	Enums.LogLevel.Errorレベルでメッセージをログに記録します。

LoggerLoupe.ErrorFormat Method (Exception, string, params object[])

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerLoupe.ErrorFormat Method (LogContext, Exception, string, params object[])

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    LogContext context,
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    LogContext context,
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    context As LogContext,
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerLoupe.ErrorFormat Method (LogContext, string, params object[])

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerLoupe.ErrorFormat Method (string, params object[])

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ErrorXml Method

これは、ErrorXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.ErrorXml (Exception, string, object)	Enums.LogLevel.Errorレベルで XML文字列をログに記録します。
	LoggerLoupe.ErrorXml (LogContext, Exception, string, object)	Enums.LogLevel.Errorレベルで XML文字列をログに記録します。
	LoggerLoupe.ErrorXml (LogContext, string, object)	Enums.LogLevel.Errorレベルで XML文字列をログに記録します。
	LoggerLoupe.ErrorXml (string, object)	Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

LoggerLoupe.ErrorXml Method (Exception, string, object)

Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerLoupe.ErrorXml Method (LogContext, Exception, string, object)

Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    context As LogContext,
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerLoupe.ErrorXml Method (LogContext, string, object)

Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerLoupe.ErrorXml Method (string, object)

Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Info Method

これは、Infoメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.Info (Func<string>)	Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerLoupe.Info (LogContext, Func<string>)	Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerLoupe.Info (LogContext, object)	Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerLoupe.Info (object)	Enums.LogLevel.Infoレベルでメッセージをログに記録します。

LoggerLoupe.Info Method (Func<string>)

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Info(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Info(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Info Method (LogContext, Func<string>)

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Info(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Info(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Info Method (LogContext, object)

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    LogContext context,
    object message
);
```

C#

```
public override void Info(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Info(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerLoupe.Info Method (object)

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    object message
);
```

C#

```
public override void Info(
    object message
);
```

Visual Basic

```
Public override Function Info(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

InfoFormat Method

これは、InfoFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.InfoFormat (LogContext, string, params object[])	Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerLoupe.InfoFormat (string, params object[])	Enums.LogLevel.Infoレベルでメッセージをログに記録します。

LoggerLoupe.InfoFormat Method (LogContext, string, params object[])

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void InfoFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void InfoFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function InfoFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerLoupe.InfoFormat Method (string, params object[])

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void InfoFormat(
    string format,
    params object[] args
);
```

C#

```
public override void InfoFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function InfoFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

InfoXml Method

これは、InfoXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.InfoXml (LogContext, string, object)	Enums.LogLevel.Infoレベルで XMLメッセージをログに記録します。
	LoggerLoupe.InfoXml (string, object)	Enums.LogLevel.Infoレベルで XML文字列をログに記録します。

LoggerLoupe.InfoXml Method (LogContext, string, object)

Enums.LogLevel.Infoレベルで XMLメッセージをログに記録します。

C++

```
void InfoXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void InfoXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function InfoXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerLoupe.InfoXml Method (string, object)

Enums.LogLevel.Infoレベルで XML 文字列をログに記録します。

C++

```
void InfoXml(
    string xml,
    object message
);
```

C#

```
public override void InfoXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function InfoXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Trace Method

これは、Traceメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.Trace (Func<string>)	Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerLoupe.Trace (LogContext, Func<string>)	Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerLoupe.Trace (LogContext, object)	Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerLoupe.Trace (object)	LogLevel.Trace レベルでメッセージをログに記録します。

LoggerLoupe.Trace Method (Func<string>)

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Trace(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Trace(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Trace Method (LogContext, Func<string>)

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Trace(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Trace(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Trace Method (LogContext, object)

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    LogContext context,
    object  message
);
```

C#

```
public override void Trace(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Trace(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerLoupe.Trace Method (object)

LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    object  message
);
```

C#

```
public override void Trace(
    object message
);
```

Visual Basic

```
Public override Function Trace(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

TraceFormat Method

これは、TraceFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.TraceFormat (LogContext, string, params object[])	Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerLoupe.TraceFormat (string, params object[])	Enums.LogLevel.Traceレベルでメッセージをログに記録します。

LoggerLoupe.TraceFormat Method (LogContext, string, params object[])

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void TraceFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void TraceFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function TraceFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerLoupe.TraceFormat Method (string, params object[])

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void TraceFormat(
    string format,
    params object[] args
);
```

C#

```
public override void TraceFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function TraceFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

Warn Method

これは、Warnメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.Warn (Exception, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerLoupe.Warn (Func<string>)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerLoupe.Warn (Func<string>, Exception)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerLoupe.Warn (LogContext, Exception, Func<string>)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerLoupe.Warn (LogContext, Exception, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerLoupe.Warn (LogContext, Func<string>)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerLoupe.Warn (LogContext, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerLoupe.Warn (object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。

LoggerLoupe.Warn Method (Exception, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Exception ex,
    object message
);
```

C#

```
public override void Warn(
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Warn(
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerLoupe.Warn Method (Func<string>)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Warn(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Warn(
    messageBuilderCallback As Func<string>,
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Warn Method (Func<string>, Exception)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

C#

```
public override void Warn(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

Visual Basic

```
Public override Function Warn(
    messageBuilderCallback As Func<string>,
    ex As Exception
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。
ex	ログの例外。

LoggerLoupe.Warn Method (LogContext, Exception, Func<string>)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Warn(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    ex As Exception,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Warn Method (LogContext, Exception, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Exception ex,
    object message
);
```

C#

```
public override void Warn(
    LogContext context,
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerLoupe.Warn Method (LogContext, Func<string>)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Warn(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerLoupe.Warn Method (LogContext, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    object message
);
```

C#

```
public override void Warn(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerLoupe.Warn Method (object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    object message
);
```

C#

```
public override void Warn(
    object message
);
```

Visual Basic

```
Public override Function Warn(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

WarnFormat Method

これは、WarnFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.WarnFormat (Exception, string, params object[])	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerLoupe.WarnFormat (LogContext, string, params object[])	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerLoupe.WarnFormat (string, params object[])	Enums.LogLevel.Warnレベルでメッセージをログに記録します。

LoggerLoupe.WarnFormat Method (Exception, string, params object[])

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public override void WarnFormat(
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function WarnFormat(
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerLoupe.WarnFormat Method (LogContext, string, params object[])

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void WarnFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function WarnFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerLoupe.WarnFormat Method (string, params object[])

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    string format,
    params object[] args
);
```

C#

```
public override void WarnFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function WarnFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

WarnXml Method

これは、WarnXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerLoupe.WarnXml (Exception, string, object)	Enums.LogLevel.Warnレベルで XML文字列をログに記録します。
	LoggerLoupe.WarnXml (LogContext, Exception, string, object)	Enums.LogLevel.Warnレベルで XML文字列をログに記録します。
	LoggerLoupe.WarnXml (LogContext, string, object)	Enums.LogLevel.Warnレベルで XML文字列をログに記録します。
	LoggerLoupe.WarnXml (string, object)	Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

LoggerLoupe.WarnXml Method (Exception, string, object)

Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

C++

```
void WarnXml(
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerLoupe.WarnXml Method (LogContext, Exception, string, object)

Enums.LogLevel.WarnレベルでXML文字列をログに記録します。

C++

```
void WarnXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    context As LogContext,
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerLoupe.WarnXml Method (LogContext, string, object)

Enums.LogLevel.WarnレベルでXML文字列をログに記録します。

C++

```
void WarnXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerLoupe.WarnXml Method (string, object)

Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

C++

```
void WarnXml(
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerNlog Class

Nlogロガーをラップします。



C++

```
class LoggerNlog : public LoggerBase;
```

C#

```
internal sealed class LoggerNlog : LoggerBase;
```

Visual Basic

```
Friend sealed Class LoggerNlog
Inherits LoggerBase
```

ファイル

LoggerNlog.cs

LoggerNlog Members

次の表は、LoggerNlogによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Debug	これは、Debugメソッドの多重定義の概要です。
	Debug	これは、Debugメソッドの多重定義の概要です。
	Debug	これは、Debugメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	LoggerNlog	新しいインスタンスを作成し、データを初期化します。
	Trace	これは、Traceメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	UpdateLoglevels	現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。
	UpdateLoglevels	現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。
	Warn	これは、Warnメソッドの多重定義の概要です。
	Warn	これは、Warnメソッドの多重定義の概要です。
	Warn	これは、Warnメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。

Public Properties

	名前	説明
	IsActive	ロガーが有効かを示すフラグ。
	IsActive	ロガーが有効かを示すフラグ。
	IsDebugEnabled	ログレベルのデバッグが有効かどうかを示すフラグ。
	IsDebugEnabled	ログレベルのデバッグが有効かどうかを示すフラグ。
	IsErrorEnabled	ログレベルエラーが有効かどうかを示すフラグ。
	IsErrorEnabled	ログレベルエラーが有効かどうかを示すフラグ。
	IsInfoEnabled	ログレベル情報が有効かどうかを示すフラグ。
	IsInfoEnabled	ログレベル情報が有効かどうかを示すフラグ。
	IsTraceEnabled	ログレベルトレースが有効かどうかを示すフラグ。
	IsTraceEnabled	ログレベルトレースが有効かどうかを示すフラグ。
	IsWarnEnabled	ログレベルの警告が有効かどうかを示すフラグ。
	IsWarnEnabled	ログレベルの警告が有効かどうかを示すフラグ。
	Name	名前を取得します。
	Name	名前を取得します。

LoggerNlog.LoggerNlog Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
LoggerEventArgs(
    NLog.Logger loggerNlog
);
```

C#

```
public LoggerEventArgs(
    NLog.Logger loggerNlog
);
```

Visual Basic

```
Public Sub New(
    loggerNlog As NLog.Logger
)
```

引数

引数	説明
loggerNlog	NLogロガー。

LoggerNlog Methods

LoggerNlogクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Debug	これは、Debugメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	Warn	これは、Warnメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。

Debug Method

これは、Debugメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.Debug (Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerNlog.Debug (LogContext, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerNlog.Debug (LogContext, object)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerNlog.Debug (object)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

LoggerNlog.Debug Method (Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Debug(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Debug(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Debug Method (LogContext, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Debug(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Debug(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Debug Method (LogContext, object)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    LogContext context,
    object message
);
```

C#

```
public override void Debug(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Debug(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerNlog.Debug Method (object)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    object message
);
```

C#

```
public override void Debug(
    object message
);
```

Visual Basic

```
Public override Function Debug(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

DebugFormat Method

これは、DebugFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.DebugFormat (LogContext, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerNlog.DebugFormat (string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

LoggerNlog.DebugFormat Method (LogContext, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void DebugFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function DebugFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerNlog.DebugFormat Method (string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugFormat(
    string format,
    params object[] args
);
```

C#

```
public override void DebugFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function DebugFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

DebugXml Method

これは、DebugXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.DebugXml (LogContext, string, object)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	LoggerNlog.DebugXml (string, object)	Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

LoggerNlog.DebugXml Method (LogContext, string, object)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void DebugXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function DebugXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerNlog.DebugXml Method (string, object)

Hbm.Api.Logging.Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugXml(
    string xml,
    object message
);
```

C#

```
public override void DebugXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function DebugXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Error Method

これは、Errorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.Error (Exception, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerNlog.Error (Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerNlog.Error (Func<string>, Exception)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerNlog.Error (LogContext, Exception, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerNlog.Error (LogContext, Exception, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerNlog.Error (LogContext, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerNlog.Error (LogContext, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerNlog.Error (object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

LoggerNlog.Error Method (Exception, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Exception ex,
    object message
);
```

C#

```
public override void Error(
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Error(
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerNlog.Error Method (Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Error(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Error(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Error Method (Func<string>, Exception)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

C#

```
public override void Error(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

Visual Basic

```
Public override Function Error(
    messageBuilderCallback As Func<string>,
    ex As Exception
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。
ex	ログの例外。

LoggerNlog.Error Method (LogContext, Exception, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Error(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    ex As Exception,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Error Method (LogContext, Exception, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Exception ex,
    object message
);
```

C#

```
public override void Error(
    LogContext context,
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerNlog.Error Method (LogContext, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Error(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Error Method (LogContext, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    object message
);
```

C#

```
public override void Error(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Error(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerNlog.Error Method (object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    object message
);
```

C#

```
public override void Error(
    object message
);
```

Visual Basic

```
Public override Function Error(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

ErrorFormat Method

これは、ErrorFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.ErrorFormat (Exception, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerNlog.ErrorFormat (LogContext, Exception, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerNlog.ErrorFormat (LogContext, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	LoggerNlog.ErrorFormat (string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

LoggerNlog.ErrorFormat Method (Exception, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerNlog.ErrorFormat Method (LogContext, Exception, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    LogContext context,
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    LogContext context,
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    context As LogContext,
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerNlog.ErrorFormat Method (LogContext, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerNlog.ErrorFormat Method (string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    string format,
    params object[] args
);
```

C#

```
public override void ErrorFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function ErrorFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ErrorXml Method

これは、ErrorXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.ErrorXml (Exception, string, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。
	LoggerNlog.ErrorXml (LogContext, Exception, string, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XMLメッセージをログに記録します。
	LoggerNlog.ErrorXml (LogContext, string, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XMLメッセージをログに記録します。
	LoggerNlog.ErrorXml (string, object)	Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

LoggerNlog.ErrorXml Method (Exception, string, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerNlog.ErrorXml Method (LogContext, Exception, string, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XMLメッセージをログに記録します。

C++

```
void ErrorXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    context As LogContext,
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerNlog.ErrorXml Method (LogContext, string, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XMLメッセージをログに記録します。

C++

```
void ErrorXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerNlog.ErrorXml Method (string, object)

Hbm.Api.Logging.Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    string xml,
    object message
);
```

C#

```
public override void ErrorXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function ErrorXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Info Method

これは、Infoメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.Info (Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerNlog.Info (LogContext, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerNlog.Info (LogContext, object)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerNlog.Info (object)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

LoggerNlog.Info Method (Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Info(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Info(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Info Method (LogContext, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Info(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Info(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Info Method (LogContext, object)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    LogContext context,
    object message
);
```

C#

```
public override void Info(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Info(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerNlog.Info Method (object)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    object message
);
```

C#

```
public override void Info(
    object message
);
```

Visual Basic

```
Public override Function Info(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

InfoFormat Method

これは、InfoFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.InfoFormat (LogContext, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	LoggerNlog.InfoFormat (string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

LoggerNlog.InfoFormat Method (LogContext, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void InfoFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void InfoFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function InfoFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerNlog.InfoFormat Method (string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void InfoFormat(
    string format,
    params object[] args
);
```

C#

```
public override void InfoFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function InfoFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

InfoXml Method

これは、InfoXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.InfoXml (LogContext, string, object)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルで XMLメッセージをログに記録します。
	LoggerNlog.InfoXml (string, object)	Hbm.Api.Logging.Enums.LogLevel.Infoレベルで XML文字列をログに記録します。

LoggerNlog.InfoXml Method (LogContext, string, object)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルで XMLメッセージをログに記録します。

C++

```
void InfoXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void InfoXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function InfoXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerNlog.InfoXml Method (string, object)

Hbm.Api.Logging.Enums.LogLevel.Infoレベルで XML 文字列をログに記録します。

C++

```
void InfoXml(
    string xml,
    object message
);
```

C#

```
public override void InfoXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function InfoXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Trace Method

これは、Traceメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.Trace (Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerNlog.Trace (LogContext, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerNlog.Trace (LogContext, object)	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerNlog.Trace (object)	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

LoggerNlog.Trace Method (Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Trace(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Trace(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Trace Method (LogContext, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Trace(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Trace(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Trace Method (LogContext, object)

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    LogContext context,
    object  message
);
```

C#

```
public override void Trace(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Trace(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerNlog.Trace Method (object)

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    object  message
);
```

C#

```
public override void Trace(
    object message
);
```

Visual Basic

```
Public override Function Trace(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

TraceFormat Method

これは、TraceFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.TraceFormat (LogContext, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	LoggerNlog.TraceFormat (string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

LoggerNlog.TraceFormat Method (LogContext, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void TraceFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void TraceFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function TraceFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerNlog.TraceFormat Method (string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void TraceFormat(
    string format,
    params object[] args
);
```

C#

```
public override void TraceFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function TraceFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

Warn Method

これは、Warnメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.Warn (Exception, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerNlog.Warn (Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerNlog.Warn (Func<string>, Exception)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerNlog.Warn (LogContext, Exception, Func<string>)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerNlog.Warn (LogContext, Exception, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録しま
	LoggerNlog.Warn (LogContext, Func<string>)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerNlog.Warn (LogContext, object)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerNlog.Warn (object)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

LoggerNlog.Warn Method (Exception, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Exception ex,
    object message
);
```

C#

```
public override void Warn(
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Warn(
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerNlog.Warn Method (Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Warn(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Warn(
    messageBuilderCallback As Func<string>,
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Warn Method (Func<string>, Exception)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

C#

```
public override void Warn(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

Visual Basic

```
Public override Function Warn(
    messageBuilderCallback As Func<string>,
    ex As Exception
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。
ex	ログの例外。

LoggerNlog.Warn Method (LogContext, Exception, Func<string>)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Warn(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    ex As Exception,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Warn Method (LogContext, Exception, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Exception ex,
    object message
);
```

C#

```
public override void Warn(
    LogContext context,
    Exception ex,
    object message
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
message	ログのメッセージオブジェクト。

LoggerNlog.Warn Method (LogContext, Func<string>)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
public override void Warn(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

LoggerNlog.Warn Method (LogContext, object)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    object message
);
```

C#

```
public override void Warn(
    LogContext context,
    object message
);
```

Visual Basic

```
Public override Function Warn(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

LoggerNlog.Warn Method (object)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    object message
);
```

C#

```
public override void Warn(
    object message
);
```

Visual Basic

```
Public override Function Warn(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

WarnFormat Method

これは、WarnFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.WarnFormat (Exception, string, params object[])	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerNlog.WarnFormat (LogContext, string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	LoggerNlog.WarnFormat (string, params object[])	Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

LoggerNlog.WarnFormat Method (Exception, string, params object[])

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
public override void WarnFormat(
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function WarnFormat(
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerNlog.WarnFormat Method (LogContext, string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
public override void WarnFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function WarnFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

LoggerNlog.WarnFormat Method (string, params object[])

Hbm.Api.Logging.Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    string format,
    params object[] args
);
```

C#

```
public override void WarnFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Public override Function WarnFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

WarnXml Method

これは、WarnXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	LoggerNlog.WarnXml (Exception, string, object)	Enums.LogLevel.Warnレベルで XML文字列をログに記録します。
	LoggerNlog.WarnXml (LogContext, Exception, string, object)	Enums.LogLevel.Warnレベルで XMLメッセージをログに記録します。
	LoggerNlog.WarnXml (LogContext, string, object)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルで XMLメッセージをログに記録します。
	LoggerNlog.WarnXml (string, object)	Hbm.Api.Logging.Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

LoggerNlog.WarnXml Method (Exception, string, object)

Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

C++

```
void WarnXml(
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerNlog.WarnXml Method (LogContext, Exception, string, object)

Enums.LogLevel.WarnレベルでXML文字列をログに記録します。

C++

```
void WarnXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    context As LogContext,
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerNlog.WarnXml Method (LogContext, string, object)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルで XMLメッセージをログに記録します。

C++

```
void WarnXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

LoggerNlog.WarnXml Method (string, object)

Hbm.Api.Logging.Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

C++

```
void WarnXml(
    string xml,
    object message
);
```

C#

```
public override void WarnXml(
    string xml,
    object message
);
```

Visual Basic

```
Public override Function WarnXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

ILogger Interface

特定のログフレームワーク(NLog、Loupeなど)をカプセル化するロガーオブジェクトのインタフェース。



C++

```
__interface ILogger;
```

C#

```
public interface ILogger;
```

Visual Basic

```
Public Interface ILogger
```

ファイル

ILogger.cs

ILogger Members

次の表は、ILoggerによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Debug	これは、Debugメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	UpdateLoglevels	現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。
	Warn	これは、Warnメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。

Public Properties

	名前	説明
	IsActive	ロガーが有効かを示すフラグ。
	IsDebugEnabled	ログレベルのデバッグが有効かどうかを示すフラグ。
	IsErrorEnabled	ログレベルエラーが有効かどうかを示すフラグ。
	IsInfoEnabled	ログレベル情報が有効かどうかを示すフラグ。
	IsTraceEnabled	ログレベルトレースが有効かどうかを示すフラグ。
	IsWarnEnabled	ログレベルの警告が有効かどうかを示すフラグ。
	Name	名前を取得します。

ILogger Methods

ILoggerクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Debug	これは、Debugメソッドの多重定義の概要です。
	DebugFormat	これは、DebugFormatメソッドの多重定義の概要です。
	DebugXml	これは、DebugXmlメソッドの多重定義の概要です。
	Error	これは、Errorメソッドの多重定義の概要です。
	ErrorFormat	これは、ErrorFormatメソッドの多重定義の概要です。
	ErrorXml	これは、ErrorXmlメソッドの多重定義の概要です。
	Info	これは、Infoメソッドの多重定義の概要です。
	InfoFormat	これは、InfoFormatメソッドの多重定義の概要です。
	InfoXml	これは、InfoXmlメソッドの多重定義の概要です。
	Trace	これは、Traceメソッドの多重定義の概要です。
	TraceFormat	これは、TraceFormatメソッドの多重定義の概要です。
	UpdateLoglevels	現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。
	Warn	これは、Warnメソッドの多重定義の概要です。
	WarnFormat	これは、WarnFormatメソッドの多重定義の概要です。
	WarnXml	これは、WarnXmlメソッドの多重定義の概要です。

Debug Method

これは、Debugメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.Debug (Func<string>)	Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	ILogger.Debug (LogContext, Func<string>)	Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	ILogger.Debug (LogContext, object)	Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	ILogger.Debug (object)	Enums.LogLevel.Debugレベルでメッセージをログに記録します。

ILogger.Debug Method (Func<string>)

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    Func<string> messageBuilderCallback
);
```

C#

```
void Debug(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Debug(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Debug Method (LogContext, Func<string>)

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
void Debug(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Debug(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Debug Method (LogContext, object)

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    LogContext context,
    object message
);
```

C#

```
void Debug(
    LogContext context,
    object message
);
```

Visual Basic

```
Function Debug(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

ILogger.Debug Method (object)

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void Debug(
    object message
);
```

C#

```
void Debug(
    object message
);
```

Visual Basic

```
Function Debug(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

DebugFormat Method

これは、DebugFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.DebugFormat (LogContext, string, params object[])	Enums.LogLevel.Debugレベルでメッセージをログに記録します。
	ILogger.DebugFormat (string, params object[])	Enums.LogLevel.Debugレベルでメッセージをログに記録します。

ILogger.DebugFormat Method (LogContext, string, params object[])

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
void DebugFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Function DebugFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ILogger.DebugFormat Method (string, params object[])

Enums.LogLevel.Debugレベルでメッセージをログに記録します。

C++

```
void DebugFormat(
    string format,
    params object[] args
);
```

C#

```
void DebugFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Function DebugFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

DebugXml Method

これは、DebugXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.DebugXml (LogContext, string, object)	Enums.LogLevel.DebugレベルでXMLメッセージをログに記録します。
	ILogger.DebugXml (string, object)	Enums.LogLevel.DebugレベルでXML文字列をログに記録します。

ILogger.DebugXml Method (LogContext, string, object)

Enums.LogLevel.Debugレベルで XMLメッセージをログに記録します。

C++

```
void DebugXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
void DebugXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Function DebugXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

ILogger.DebugXml Method (string, object)

Enums.LogLevel.DebugレベルでXML文字列をログに記録します。

C++

```
void DebugXml(
    string xml,
    object message
);
```

C#

```
void DebugXml(
    string xml,
    object message
);
```

Visual Basic

```
Function DebugXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Error Method

これは、Errorメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.Error (Exception, object)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	ILogger.Error (Func<string>)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	ILogger.Error (Func<string>, Exception)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	ILogger.Error (LogContext, Exception, Func<string>)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	ILogger.Error (LogContext, Exception, object)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	ILogger.Error (LogContext, Func<string>)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	ILogger.Error (LogContext, object)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	ILogger.Error (object)	Enums.LogLevel.Errorレベルでメッセージをログに記録します。

ILogger.Error Method (Exception, object)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Exception ex,
    object message
);
```

C#

```
void Error(
    Exception ex,
    object message
);
```

Visual Basic

```
Function Error(
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
message	ログのメッセージオブジェクト。

ILogger.Error Method (Func<string>)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Func<string> messageBuilderCallback
);
```

C#

```
void Error(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Error(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Error Method (Func<string>, Exception)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

C#

```
void Error(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

Visual Basic

```
Function Error(
    messageBuilderCallback As Func<string>,
    ex As Exception
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。
ex	ログの例外。

ILogger.Error Method (LogContext, Exception, Func<string>)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

C#

```
void Error(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Error(
    context As LogContext,
    ex As Exception,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Error Method (LogContext, Exception, object)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Exception ex,
    object message
);
```

C#

```
void Error(
    LogContext context,
    Exception ex,
    object message
);
```

Visual Basic

```
Function Error(
    context As LogContext,
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
message	ログのメッセージオブジェクト。

ILogger.Error Method (LogContext, Func<string>)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
void Error(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Error(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Error Method (LogContext, object)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    LogContext context,
    object message
);
```

C#

```
void Error(
    LogContext context,
    object message
);
```

Visual Basic

```
Function Error(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

ILogger.Error Method (object)

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void Error(
    object message
);
```

C#

```
void Error(
    object message
);
```

Visual Basic

```
Function Error(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

ErrorFormat Method

これは、ErrorFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.ErrorFormat (Exception, string, params object[])	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	ILogger.ErrorFormat (LogContext, Exception, string, params object[])	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	ILogger.ErrorFormat (LogContext, string, params object[])	Enums.LogLevel.Errorレベルでメッセージをログに記録します。
	ILogger.ErrorFormat (string, params object[])	Enums.LogLevel.Errorレベルでメッセージをログに記録します。

ILogger.ErrorFormat Method (Exception, string, params object[])

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
void ErrorFormat(
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Function ErrorFormat(
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ILogger.ErrorFormat Method (LogContext, Exception, string, params object[])

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    LogContext context,
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
void ErrorFormat(
    LogContext context,
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Function ErrorFormat(
    context As LogContext,
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ILogger.ErrorFormat Method (LogContext, string, params object[])

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
void ErrorFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Function ErrorFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ILogger.ErrorFormat Method (string, params object[])

Enums.LogLevel.Errorレベルでメッセージをログに記録します。

C++

```
void ErrorFormat(
    string format,
    params object[] args
);
```

C#

```
void ErrorFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Function ErrorFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ErrorXml Method

これは、ErrorXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.ErrorXml (Exception, string, object)	Enums.LogLevel.Errorレベルで XML文字列をログに記録します。
	ILogger.ErrorXml (LogContext, Exception, string, object)	Enums.LogLevel.Errorレベルで XMLメッセージをログに記録します。
	ILogger.ErrorXml (LogContext, string, object)	Enums.LogLevel.Errorレベルで XMLメッセージをログに記録します。
	ILogger.ErrorXml (string, object)	Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

ILogger.ErrorXml Method (Exception, string, object)

Enums.LogLevel.ErrorレベルでXML文字列をログに記録します。

C++

```
void ErrorXml(
    Exception ex,
    string xml,
    object message
);
```

C#

```
void ErrorXml(
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Function ErrorXml(
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

ILogger.ErrorXml Method (LogContext, Exception, string, object)

Enums.LogLevel.ErrorレベルでXMLメッセージをログに記録します。

C++

```
void ErrorXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

C#

```
void ErrorXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Function ErrorXml(
    context As LogContext,
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

ILogger.ErrorXml Method (LogContext, string, object)

C++

```
void ErrorXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
void ErrorXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Function ErrorXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

ILogger.ErrorXml Method (string, object)

Enums.LogLevel.Errorレベルで XML文字列をログに記録します。

C++

```
void ErrorXml(
    string xml,
    object message
);
```

C#

```
void ErrorXml(
    string xml,
    object message
);
```

Visual Basic

```
Function ErrorXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Info Method

これは、Infoメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.Info (Func<string>)	Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	ILogger.Info (LogContext, Func<string>)	Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	ILogger.Info (LogContext, object)	Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	ILogger.Info (object)	Enums.LogLevel.Infoレベルでメッセージをログに記録します。

ILogger.Info Method (Func<string>)

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    Func<string> messageBuilderCallback
);
```

C#

```
void Info(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Info(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Info Method (LogContext, Func<string>)

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
void Info(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Info(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Info Method (LogContext, object)

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    LogContext context,
    object message
);
```

C#

```
void Info(
    LogContext context,
    object message
);
```

Visual Basic

```
Function Info(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

ILogger.Info Method (object)

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void Info(
    object message
);
```

C#

```
void Info(
    object message
);
```

Visual Basic

```
Function Info(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

InfoFormat Method

これは、InfoFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.InfoFormat (LogContext, string, params object[])	Enums.LogLevel.Infoレベルでメッセージをログに記録します。
	ILogger.InfoFormat (string, params object[])	Enums.LogLevel.Infoレベルでメッセージをログに記録します。

ILogger.InfoFormat Method (LogContext, string, params object[])

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void InfoFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
void InfoFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Function InfoFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ILogger.InfoFormat Method (string, params object[])

Enums.LogLevel.Infoレベルでメッセージをログに記録します。

C++

```
void InfoFormat(
    string format,
    params object[] args
);
```

C#

```
void InfoFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Function InfoFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

InfoXml Method

これは、InfoXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.InfoXml (LogContext, string, object)	Enums.LogLevel.Infoレベルで XMLメッセージをログに記録します。
	ILogger.InfoXml (string, object)	Enums.LogLevel.Infoレベルで XML文字列をログに記録します。

ILogger.InfoXml Method (LogContext, string, object)

Enums.LogLevel.Infoレベルで XMLメッセージをログに記録します。

C++

```
void InfoXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
void InfoXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Function InfoXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

ILogger.InfoXml Method (string, object)

Enums.LogLevel.Infoレベルで XML 文字列をログに記録します。

C++

```
void InfoXml(
    string xml,
    object message
);
```

C#

```
void InfoXml(
    string xml,
    object message
);
```

Visual Basic

```
Function InfoXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

Trace Method

これは、Traceメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.Trace (Func<string>)	Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	ILogger.Trace (LogContext, Func<string>)	Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	ILogger.Trace (LogContext, object)	Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	ILogger.Trace (object)	Enums.LogLevel.Traceレベルでメッセージをログに記録します。

ILogger.Trace Method (Func<string>)

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    Func<string> messageBuilderCallback
);
```

C#

```
void Trace(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Trace(
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Trace Method (LogContext, Func<string>)

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
void Trace(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Trace(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Trace Method (LogContext, object)

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    LogContext context,
    object  message
);
```

C#

```
void Trace(
    LogContext context,
    object message
);
```

Visual Basic

```
Function Trace(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

ILogger.Trace Method (object)

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void Trace(
    object  message
);
```

C#

```
void Trace(
    object message
);
```

Visual Basic

```
Function Trace(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

TraceFormat Method

これは、TraceFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.TraceFormat (LogContext, string, params object[])	Enums.LogLevel.Traceレベルでメッセージをログに記録します。
	ILogger.TraceFormat (string, params object[])	Enums.LogLevel.Traceレベルでメッセージをログに記録します。

ILogger.TraceFormat Method (LogContext, string, params object[])

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void TraceFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
void TraceFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Function TraceFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ILogger.TraceFormat Method (string, params object[])

Enums.LogLevel.Traceレベルでメッセージをログに記録します。

C++

```
void TraceFormat(
    string format,
    params object[] args
);
```

C#

```
void TraceFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Function TraceFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ILogger.UpdateLoglevels Method

現在のHbm.Api.Logging.LogManager.LogLevelsActiveプロパティに基づいてログレベルを更新します。

C++

```
void UpdateLoglevels();
```

C#

```
void UpdateLoglevels();
```

Visual Basic

```
Function UpdateLoglevels() As void
```

Warn Method

これは、Warnメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.Warn (Exception, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	ILogger.Warn (Func<string>)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	ILogger.Warn (Func<string>, Exception)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	ILogger.Warn (LogContext, Exception, Func<string>)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	ILogger.Warn (LogContext, Exception, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	ILogger.Warn (LogContext, Func<string>)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	ILogger.Warn (LogContext, object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	ILogger.Warn (object)	Enums.LogLevel.Warnレベルでメッセージをログに記録します。

ILogger.Warn Method (Exception, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Exception ex,
    object message
);
```

C#

```
void Warn(
    Exception ex,
    object message
);
```

Visual Basic

```
Function Warn(
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
message	ログのメッセージオブジェクト。

ILogger.Warn Method (Func<string>)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Func<string> messageBuilderCallback
);
```

C#

```
void Warn(
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Warn(
    messageBuilderCallback As Func<string>,
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Warn Method (Func<string>, Exception)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

C#

```
void Warn(
    Func<string> messageBuilderCallback,
    Exception ex
);
```

Visual Basic

```
Function Warn(
    messageBuilderCallback As Func<string>,
    ex As Exception
) As void
```

引数

引数	説明
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。
ex	ログの例外。

ILogger.Warn Method (LogContext, Exception, Func<string>)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

C#

```
void Warn(
    LogContext context,
    Exception ex,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Warn(
    context As LogContext,
    ex As Exception,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Warn Method (LogContext, Exception, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Exception ex,
    object message
);
```

C#

```
void Warn(
    LogContext context,
    Exception ex,
    object message
);
```

Visual Basic

```
Function Warn(
    context As LogContext,
    ex As Exception,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
message	ログのメッセージオブジェクト。

ILogger.Warn Method (LogContext, Func<string>)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

C#

```
void Warn(
    LogContext context,
    Func<string> messageBuilderCallback
);
```

Visual Basic

```
Function Warn(
    context As LogContext,
    messageBuilderCallback As Func<string>
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
messageBuilderCallback	ログメッセージ文字列を返すコールバックメソッド。

ILogger.Warn Method (LogContext, object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    LogContext context,
    object message
);
```

C#

```
void Warn(
    LogContext context,
    object message
);
```

Visual Basic

```
Function Warn(
    context As LogContext,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
message	ログのメッセージオブジェクト。

ILogger.Warn Method (object)

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void Warn(
    object message
);
```

C#

```
void Warn(
    object message
);
```

Visual Basic

```
Function Warn(
    message As object
) As void
```

引数

引数	説明
message	ログのメッセージオブジェクト。

WarnFormat Method

これは、WarnFormatメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.WarnFormat (Exception, string, params object[])	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	ILogger.WarnFormat (LogContext, string, params object[])	Enums.LogLevel.Warnレベルでメッセージをログに記録します。
	ILogger.WarnFormat (string, params object[])	Enums.LogLevel.Warnレベルでメッセージをログに記録します。

ILogger.WarnFormat Method (Exception, string, params object[])

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    Exception ex,
    string format,
    params object[] args
);
```

C#

```
void WarnFormat(
    Exception ex,
    string format,
    params object[] args
);
```

Visual Basic

```
Function WarnFormat(
    ex As Exception,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
ex	ログの例外。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ILogger.WarnFormat Method (LogContext, string, params object[])

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    LogContext context,
    string format,
    params object[] args
);
```

C#

```
void WarnFormat(
    LogContext context,
    string format,
    params object[] args
);
```

Visual Basic

```
Function WarnFormat(
    context As LogContext,
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

ILogger.WarnFormat Method (string, params object[])

Enums.LogLevel.Warnレベルでメッセージをログに記録します。

C++

```
void WarnFormat(
    string format,
    params object[] args
);
```

C#

```
void WarnFormat(
    string format,
    params object[] args
);
```

Visual Basic

```
Function WarnFormat(
    format As string,
    args As params object[]
) As void
```

引数

引数	説明
format	ログに記録するメッセージオブジェクトの形式。String.Format(string, object[])を参照。
args	引数の形式。

WarnXml Method

これは、WarnXmlメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	ILogger.WarnXml (Exception, string, object)	Enums.LogLevel.Warnレベルで XML 文字列をログに記録します。
	ILogger.WarnXml (LogContext, Exception, string, object)	Enums.LogLevel.Warnレベルで XML メッセージをログに記録します。
	ILogger.WarnXml (LogContext, string, object)	Enums.LogLevel.Warnレベルで XML メッセージをログに記録します。
	ILogger.WarnXml (string, object)	Enums.LogLevel.Warnレベルで XML 文字列をログに記録します。

ILogger.WarnXml Method (Exception, string, object)

Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

C++

```
void WarnXml(
    Exception ex,
    string xml,
    object message
);
```

C#

```
void WarnXml(
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Function WarnXml(
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

ILogger.WarnXml Method (LogContext, Exception, string, object)

Enums.LogLevel.Warnレベルで XMLメッセージをログに記録します。

C++

```
void WarnXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

C#

```
void WarnXml(
    LogContext context,
    Exception ex,
    string xml,
    object message
);
```

Visual Basic

```
Function WarnXml(
    context As LogContext,
    ex As Exception,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
ex	ログの例外。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

ILogger.WarnXml Method (LogContext, string, object)

Enums.LogLevel.WarnレベルでXMLメッセージをログに記録します。

C++

```
void WarnXml(
    LogContext context,
    string xml,
    object message
);
```

C#

```
void WarnXml(
    LogContext context,
    string xml,
    object message
);
```

Visual Basic

```
Function WarnXml(
    context As LogContext,
    xml As string,
    message As object
) As void
```

引数

引数	説明
context	ログメッセージのコンテキストの詳細。LogContextは '.'で区切られます。
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

ILogger.WarnXml Method (string, object)

Enums.LogLevel.Warnレベルで XML文字列をログに記録します。

C++

```
void WarnXml(
    string xml,
    object message
);
```

C#

```
void WarnXml(
    string xml,
    object message
);
```

Visual Basic

```
Function WarnXml(
    xml As string,
    message As object
) As void
```

引数

引数	説明
xml	ログのXML文字列。
message	ログのメッセージオブジェクト。

ILogger Properties

ILoggerクラスのプロパティの一覧です。

Public Properties

	名前	説明
	IsActive	ロガーが有効かを示すフラグ。
	IsDebugEnabled	ログレベルのデバッグが有効かどうかを示すフラグ。
	IsErrorEnabled	ログレベルエラーが有効かどうかを示すフラグ。
	IsInfoEnabled	ログレベル情報が有効かどうかを示すフラグ。
	IsTraceEnabled	ログレベルトレースが有効かどうかを示すフラグ。
	IsWarnEnabled	ログレベルの警告が有効かどうかを示すフラグ。
	Name	名前を取得します。

ILogger.IsActive Property

ロガーが有効かを示すフラグ。

C++

```
__property bool IsActive;
```

C#

```
bool IsActive;
```

Visual Basic

```
Property IsActive() As bool
```

ILogger.IsDebugEnabled Property

ログレベルのデバッグが有効かどうかを示すフラグ。

C++

```
__property bool IsDebugEnabled;
```

C#

```
bool IsDebugEnabled;
```

Visual Basic

```
Property IsDebugEnabled() As bool
```

ILogger.IsErrorEnabled Property

ログレベルエラーが有効かどうかを示すフラグ。

C++

```
__property bool IsErrorEnabled;
```

C#

```
bool IsErrorEnabled;
```

Visual Basic

```
Property IsErrorEnabled() As bool
```

ILogger.IsInfoEnabled Property

ログレベル情報が有効かどうかを示すフラグ。

C++

```
__property bool IsInfoEnabled;
```

C#

```
bool IsInfoEnabled;
```

Visual Basic

```
Property IsInfoEnabled() As bool
```

ILogger.IsTraceEnabled Property

ログレベルトレースが有効かどうかを示すフラグ。

C++

```
__property bool IsTraceEnabled;
```

C#

```
bool IsTraceEnabled;
```

Visual Basic

```
Property IsTraceEnabled() As bool
```

ILogger.IsWarnEnabled Property

ログレベルの警告が有効かどうかを示すフラグ。

C++

```
__property bool IsWarnEnabled;
```

C#

```
bool IsWarnEnabled;
```

Visual Basic

```
Property IsWarnEnabled() As bool
```

ILogger.Name Property

名前を取得します。

C++

```
__property string Name;
```

C#

```
string Name;
```

Visual Basic

```
Property Name() As string
```

Supported devices

目次

Supported devices ・・・

QuantumXDevice.DisableRouting Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2615
QuantumXDevice.EnableRouting Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2616
FlashLed Method ・・ 2617
QuantumXDevice.FlashLed Method (Connector, LedFlashMode, out List<Problem>) ・・・・・・・・ 2618
QuantumXDevice.FlashLed Method (LedFlashMode, out List<Problem>) ・・・・・・・・・・・・・ 2619
QuantumXDevice.FlashLed Method (List<Connector>, LedFlashMode, out List<Problem>) ・・・・・ 2620
QuantumXDevice.GetAllStandardConnectors Method ・・・・・・・・・・・・・・・・・・・・・・ 2621
QuantumXDevice.GetAmplifierSerialNumber Method ・・・・・・・・・・・・・・・・・・・・・・ 2622
QuantumXDevice.GetAmplifierType Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2623
QuantumXDevice.GetCalibrationCertificateFileInfos Method ・・・・・・・・・・・・・・・・・・ 2624
QuantumXDevice.GetChannelHardwareInfo Method ・・・・・・・・・・・・・・・・・・・・・・・ 2625
GetDeviceErrorStatusList Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2626
QuantumXDevice.GetDeviceErrorStatusList Method (Channel, out List<Problem>) ・・・・・・・・ 2627
QuantumXDevice.GetDeviceErrorStatusList Method (Connector, out List<Problem>) ・・・・・・・・ 2628
QuantumXDevice.GetDeviceErrorStatusList Method (out List<Problem>) ・・・・・・・・・・・・・ 2629
GetDeviceStatusList Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2630
QuantumXDevice.GetDeviceStatusList Method (Channel, out List<Problem>) ・・・・・・・・・・・ 2631
QuantumXDevice.GetDeviceStatusList Method (Connector, out List<Problem>) ・・・・・・・・・・ 2632
QuantumXDevice.GetDeviceStatusList Method (out List<Problem>, StatusType) ・・・・・・・・・・ 2633
QuantumXDevice.GetOverflowValue Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2634
QuantumXDevice.GetPossibleDataAcquisitionModes Method ・・・・・・・・・・・・・・・・・・・ 2635
GetShunt Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2636
QuantumXDevice.GetShunt Method (Connector) ・・・・・・・・・・・・・・・・・・・・・・・・ 2637
QuantumXDevice.GetShunt Method (List<Connector>) ・・・・・・・・・・・・・・・・・・・・・ 2638
QuantumXDevice.GetSupportedCanBitRates Method ・・・・・・・・・・・・・・・・・・・・・・ 2639
QuantumXDevice.GetSupportedFilterFrequencies Method ・・・・・・・・・・・・・・・・・・・・ 2640
QuantumXDevice.GetSupportedFilters Method ・・・・・・・・・・・・・・・・・・・・・・・・・ 2641
QuantumXDevice.GetSupportedJackType Method ・・・・・・・・・・・・・・・・・・・・・・・・ 2642
QuantumXDevice.GetSupportedSampleRates Method ・・・・・・・・・・・・・・・・・・・・・・ 2643
QuantumXDevice.GetSupportedScalingTypes Method ・・・・・・・・・・・・・・・・・・・・・・ 2644
QuantumXDevice.GetSupportedSensorTypes Method ・・・・・・・・・・・・・・・・・・・・・・ 2645
QuantumXDevice.GetTimeSourceQuality Method ・・・・・・・・・・・・・・・・・・・・・・・・ 2646
QuantumXDevice.HandleChannelsChangesTriggeredByTedsActivation Method ・・・・・・・・・・・ 2647
QuantumXDevice.IsFirmwareVersionSupported Method ・・・・・・・・・・・・・・・・・・・・・ 2648
QuantumXDevice.MapConnectionInfo Method ・・・・・・・・・・・・・・・・・・・・・・・・・ 2649
QuantumXDevice.PrepareDaq Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2650
QuantumXDevice.ReadFbgSpectrum Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2651
QuantumXDevice.ReadSingleElectricalValue Method ・・・・・・・・・・・・・・・・・・・・・・ 2652
QuantumXDevice.ReadSingleMeasurementValue Method ・・・・・・・・・・・・・・・・・・・・・ 2653
QuantumXDevice.ReadSingleMeasurementValueOfAllSignals Method ・・・・・・・・・・・・・・・ 2654
QuantumXDevice.ReadTeds Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2655
QuantumXDevice.Reconnect Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2656
QuantumXDevice.RefreshDeviceInfo Method ・・・・・・・・・・・・・・・・・・・・・・・・・ 2657
QuantumXDevice.RemoveInputSignalReference Method ・・・・・・・・・・・・・・・・・・・・・ 2658
QuantumXDevice.Reset Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2659
QuantumXDevice.ResetCanController Method ・・・・・・・・・・・・・・・・・・・・・・・・・ 2660
QuantumXDevice.SetAnalogOut Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2661
QuantumXDevice.SetCanBusParameters Method ・・・・・・・・・・・・・・・・・・・・・・・・ 2662
QuantumXDevice.SetCanChannelActivation Method ・・・・・・・・・・・・・・・・・・・・・・ 2663
QuantumXDevice.SetChannelName Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2664
QuantumXDevice.SetDataRateDomain Method ・・・・・・・・・・・・・・・・・・・・・・・・・ 2665
QuantumXDevice.SetDeviceName Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2666
QuantumXDevice.SetDigitalIO Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2667

QuantumXDevice.SetFbgBandsDetectionResult Method ・・・・・・・・・・・・・・・・・・・・・・・・ 2668
QuantumXDevice.SetInputSignalReference Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2669
SetShunt Method ・・ 2670
QuantumXDevice.SetShunt Method (Connector, ShuntMode, out List<Problem>) ・・・・・・・・・・ 2671
QuantumXDevice.SetShunt Method (List<Connector>, ShuntMode, out List<Problem>) ・・・・・・・ 2672
QuantumXDevice.SetSignalName Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2673
QuantumXDevice.SetTedsUsageMode Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2674
QuantumXDevice.SetUnit Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2675
QuantumXDevice.SupportsExtendedSampleRateMode Method ・・・・・・・・・・・・・・・・・・・・ 2676
QuantumXDevice.SupportsPacketRateChange Method ・・・・・・・・・・・・・・・・・・・・・・・・ 2677
QuantumXDevice.UpdateCanConnectorsSettings Method ・・・・・・・・・・・・・・・・・・・・・・ 2678
QuantumXDevice.UpdateConnectorSettings Method ・・・・・・・・・・・・・・・・・・・・・・・・ 2679
QuantumXDevice.WriteTeds Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2680
QuantumXDevice Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2681
QuantumXDevice.AdditionalFeatures Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2682
QuantumXDevice.Extension Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2683
QuantumXDevice.FamilyName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2684
QuantumXDevice.FpgaVersion Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2685
QuantumXDevice.HardwareControllerDate Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2686
QuantumXDevice.HardwareControllerProgramDate Property ・・・・・・・・・・・・・・・・・・・・ 2687
QuantumXDevice.HardwareControllerProgramVersion Property ・・・・・・・・・・・・・・・・・・ 2688
QuantumXDevice.HardwareControllerVersion Property ・・・・・・・・・・・・・・・・・・・・・・ 2689
QuantumXDevice.HardwareIdentification Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2690
QuantumXDevice.HasAdditionalTedsFieldsSupport Property ・・・・・・・・・・・・・・・・・・・・ 2691
QuantumXDevice.IsAutomaticTedsMonitoringSupported Property ・・・・・・・・・・・・・・・・・ 2692
QuantumXDevice.IsElectricalSnapshotSupported Property ・・・・・・・・・・・・・・・・・・・・ 2693
QuantumXDevice.IsFirmwareUpdateSupported Property ・・・・・・・・・・・・・・・・・・・・・・ 2694
QuantumXDevice.IsReadingFbgSpectrumSupported Property ・・・・・・・・・・・・・・・・・・・・ 2695
QuantumXDevice.ModuleId Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2696
QuantumXDevice.ModuleLabel Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2697
QuantumXDevice.Revision Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2698
QuantumXDevice.Selection Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2699
QuantumXDevice.SpecialFirmwareVersion Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2700
QuantumXDevice.SpecialVersion Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2701
QuantumXDevice.SupportsCanBusTermination Property ・・・・・・・・・・・・・・・・・・・・・・ 2702
QuantumXDevice.SupportsChangingConnectorType Property ・・・・・・・・・・・・・・・・・・・・ 2703
QuantumXDeviceFamily Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2704
QuantumXDeviceFamily Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2705
QuantumXDeviceFamily.QuantumXDeviceFamily Constructor ・・・・・・・・・・・・・・・・・・・・ 2706
QuantumXDeviceFamily Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2707
QuantumXDeviceFamily.Find Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2708
QuantumXDeviceFamily.GetPossibleDataAcquisitionModes Method ・・・・・・・・・・・・・・・・・ 2709
QuantumXDeviceFamily.PrepareDataAcquisition Method ・・・・・・・・・・・・・・・・・・・・・・ 2710
QuantumXDeviceFamily.Scan Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2711
QuantumXDeviceFamily Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2712
QuantumXDeviceFamily.FamilyName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2713
QuantumXDeviceFamily.IsScanSupported Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 2714
QuantumXDeviceFamily.SkillReader Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2715
Hbm.Api.QuantumX.Channels Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2716
QuantumXAnalogInChannel Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2717
QuantumXAnalogInChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2718
QuantumXAnalogInChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2719
QuantumXAnalogInChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2720

QuantumXAnalogInChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2721
QuantumXAnalogInChannel.ConnectorPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2722
QuantumXAnalogInChannel.OriginOfName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2723
QuantumXAnalogInChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2724
QuantumXAnalogInChannel.TedsUsageMode Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 2725
QuantumXAnalogInChannel.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・・ 2726
QuantumXAnalogOutChannel Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2727
QuantumXAnalogOutChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2728
QuantumXAnalogOutChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2729
QuantumXAnalogOutChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2730
QuantumXAnalogOutChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2731
QuantumXAnalogOutChannel.CalibratedBy Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 2732
QuantumXAnalogOutChannel.CalibrationDate Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2733
QuantumXAnalogOutChannel.CalibrationExpirationDate Property ・・・・・・・・・・・・・・・・・・・ 2734
QuantumXAnalogOutChannel.ConnectorPosition Property ・・・・・・・・・・・・・・・・・・・・・・・ 2735
QuantumXAnalogOutChannel.Indentifier Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2736
QuantumXAnalogOutChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2737
QuantumXAnalogOutChannel.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・・ 2738
QuantumXCanInChannel Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2739
QuantumXCanInChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2740
QuantumXCanInChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2741
QuantumXCanInChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2742
QuantumXCanInChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2743
QuantumXCanInChannel.ConnectorPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 2744
QuantumXCanInChannel.OriginOfName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2745
QuantumXCanInChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2746
QuantumXCanInChannel.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2747
QuantumXCanRawChannel Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2748
QuantumXCanRawChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2749
QuantumXCanRawChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2750
QuantumXCanRawChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2751
QuantumXCanRawChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2752
QuantumXCanRawChannel.ConnectorPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2753
QuantumXCanRawChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2754
QuantumXCanRawChannel.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・・ 2755
QuantumXDigitalChannel Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2756
QuantumXDigitalChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2757
QuantumXDigitalChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2758
QuantumXDigitalChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2759
QuantumXDigitalChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2760
QuantumXDigitalChannel.ConnectorPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2761
QuantumXDigitalChannel.DefaultValue Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2762
QuantumXDigitalChannel.InputModuleReference Property ・・・・・・・・・・・・・・・・・・・・・・ 2763
QuantumXDigitalChannel.InputSignalReference Property ・・・・・・・・・・・・・・・・・・・・・・ 2764
QuantumXDigitalChannel.IsInverted Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2765
QuantumXDigitalChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2766
QuantumXDigitalChannel.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・・ 2767
QuantumXVirtualChannel Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2768
QuantumXVirtualChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2769
QuantumXVirtualChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2770
QuantumXVirtualChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2771
QuantumXVirtualChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2772
QuantumXVirtualChannel.ConnectorPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2773

QuantumXVirtualChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2774
QuantumXVirtualChannel.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2775
IQuantumXChannel Interface ・・ 2776
IQuantumXChannel Members ・・ 2777
IQuantumXChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2778
IQuantumXChannel.ConnectorPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2779
IQuantumXChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2780
IQuantumXChannel.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2781
Hbm.Api.QuantumX.Connectors Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2782
QuantumXAnalogInConnector Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2783
QuantumXAnalogInConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2784
QuantumXAnalogInConnector Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2785
QuantumXAnalogInConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2786
QuantumXAnalogInConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2787
QuantumXAnalogInConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2788
QuantumXAnalogInConnector.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・ 2789
QuantumXAnalogInConnector.XmlView Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2790
QuantumXAnalogOutConnector Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2791
QuantumXAnalogOutConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2792
QuantumXAnalogOutConnector Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2793
QuantumXAnalogOutConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・ 2794
QuantumXAnalogOutConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2795
QuantumXAnalogOutConnector.OutputType Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 2796
QuantumXAnalogOutConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2797
QuantumXAnalogOutConnector.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・ 2798
QuantumXAnalogOutConnector.XmlView Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2799
QuantumXCanConnector Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2800
QuantumXCanConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2801
QuantumXCanConnector.QuantumXCanConnector Constructor ・・・・・・・・・・・・・・・・・・・・ 2802
QuantumXCanConnector Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2803
QuantumXCanConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2804
QuantumXCanConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2805
QuantumXCanConnector.Ecus Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2806
QuantumXCanConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2807
QuantumXCanConnector.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2808
QuantumXCanConnector.XmlView Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2809
QuantumXDigitalConnector Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2810
QuantumXDigitalConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2811
QuantumXDigitalConnector Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2812
QuantumXDigitalConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2813
QuantumXDigitalConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2814
QuantumXDigitalConnector.Pin Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2815
QuantumXDigitalConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2816
QuantumXDigitalConnector.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・ 2817
QuantumXDigitalConnector.XmlView Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2818
QuantumXOffConnector Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2819
QuantumXOffConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2820
QuantumXOffConnector Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2821
QuantumXOffConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2822
QuantumXOffConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2823
QuantumXOffConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2824
QuantumXOffConnector.XmlAddressPosition Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2825
QuantumXOffConnector.XmlView Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2826

IQuantumXConnector Interface ・・・

QuantumXAnalogInSignal.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2968
QuantumXAnalogInSignal Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2969
QuantumXAnalogInSignal.Description Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2970
QuantumXAnalogInSignal.IsDaqAvailable Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2971
QuantumXAnalogInSignal.IsIsochronous Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2972
QuantumXAnalogInSignal.OriginOfName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2973
QuantumXAnalogInSignal.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2974
QuantumXAnalogInSignal.SignalReference Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2975
QuantumXAnalogOutSignal Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2976
QuantumXAnalogOutSignal Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2977
QuantumXAnalogOutSignal.QuantumXAnalogOutSignal Constructor ・・・・・・・・・・・・・・・・・・ 2978
QuantumXAnalogOutSignal Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2979
QuantumXAnalogOutSignal.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2980
QuantumXAnalogOutSignal Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2981
QuantumXAnalogOutSignal.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2982
QuantumXAnalogOutSignal.SignalReference Property ・・・・・・・・・・・・・・・・・・・・・・・・ 2983
QuantumXCanInSignal Class ・・ 2984
QuantumXCanInSignal Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2985
QuantumXCanInSignal Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2986
QuantumXCanInSignal.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2987
QuantumXCanInSignal Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2988
QuantumXCanInSignal.Description Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2989
QuantumXCanInSignal.IsDaqAvailable Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2990
QuantumXCanInSignal.IsIsochronous Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2991
QuantumXCanInSignal.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2992
QuantumXCanInSignal.SignalReference Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2993
QuantumXCanRawSignal Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2994
QuantumXCanRawSignal Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2995
QuantumXCanRawSignal Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2996
QuantumXCanRawSignal.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2997
QuantumXCanRawSignal Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2998
QuantumXCanRawSignal.IsDaqAvailable Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 2999
QuantumXCanRawSignal.IsIsochronous Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3000
QuantumXCanRawSignal.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3001
QuantumXCanRawSignal.SignalReference Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 3002
QuantumXDigitalSignal Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3003
QuantumXDigitalSignal Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3004
QuantumXDigitalSignal Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3005
QuantumXDigitalSignal.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3006
QuantumXDigitalSignal Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3007
QuantumXDigitalSignal.Description Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3008
QuantumXDigitalSignal.IsDaqAvailable Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3009
QuantumXDigitalSignal.IsIsochronous Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3010
QuantumXDigitalSignal.OriginOfName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3011
QuantumXDigitalSignal.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3012
QuantumXDigitalSignal.SignalReference Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 3013
QuantumXSyncSignal Class ・・ 3014
QuantumXSyncSignal Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3015
QuantumXSyncSignal Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3016
QuantumXSyncSignal.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3017
QuantumXSyncSignal Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3018
QuantumXSyncSignal.Description Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3019
QuantumXSyncSignal.IsDaqAvailable Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3020

QuantumXSyncSignal.IsIsochronous Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3021
QuantumXSyncSignal.OriginOfName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3022
QuantumXSyncSignal.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3023
QuantumXSyncSignal.SignalReference Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3024
IQuantumXIsochronousSignal Interface ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3025
IQuantumXIsochronousSignal Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3026
IQuantumXIsochronousSignal Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3027
IQuantumXIsochronousSignal.IsIsochronous Property ・・・・・・・・・・・・・・・・・・・・・・・ 3028
IQuantumXSignal Interface ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3029
IQuantumXSignal Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3030
IQuantumXSignal Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3031
IQuantumXSignal.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3032
IQuantumXSignal.SignalReference Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3033
Hbm.Api.QuantumX.Statuses Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3034
QuantumXChannelTestSignalActiveStatus Class ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3035
QuantumXChannelTestSignalActiveStatus Members ・・・・・・・・・・・・・・・・・・・・・・・・ 3036
QuantumXChannelTestSignalActiveStatus.QuantumXChannelTestSignalActiveStatus Constructor ・・・ 3037
QuantumXChannelTestSignalActiveStatus Methods ・・・・・・・・・・・・・・・・・・・・・・・・ 3038
QuantumXChannelTestSignalActiveStatus.IsEqual Method ・・・・・・・・・・・・・・・・・・・・ 3039
QuantumXChannelTestSignalActiveStatus Properties ・・・・・・・・・・・・・・・・・・・・・・・ 3040
QuantumXChannelTestSignalActiveStatus.Value Property ・・・・・・・・・・・・・・・・・・・・ 3041
QuantumXConnectorShuntStatus Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3042
QuantumXConnectorShuntStatus Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3043
QuantumXConnectorShuntStatus.QuantumXConnectorShuntStatus Constructor ・・・・・・・・・・・ 3044
QuantumXConnectorShuntStatus Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3045
QuantumXConnectorShuntStatus.IsEqual Method ・・・・・・・・・・・・・・・・・・・・・・・・ 3046
QuantumXConnectorShuntStatus Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3047
QuantumXConnectorShuntStatus.Value Property ・・・・・・・・・・・・・・・・・・・・・・・・ 3048
Hbm.Api.Pmx Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3049
サポートされていないPMXの機能 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3050
PMXの追加機能 ・・・ 3051
PmxAdditionalFeatures Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3052
PmxAdditionalFeatures Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3053
PmxAdditionalFeatures.PmxAdditionalFeatures Constructor ・・・・・・・・・・・・・・・・・・・ 3054
PmxAdditionalFeatures Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3055
PmxAdditionalFeatures.ActivateTEDS Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3056
PmxAdditionalFeatures.AssignLimitSwitch Method ・・・・・・・・・・・・・・・・・・・・・・・ 3057
PmxAdditionalFeatures.ClearPeakValues Method ・・・・・・・・・・・・・・・・・・・・・・・・ 3058
PmxAdditionalFeatures.GetCurrentParameterSetNumber Method ・・・・・・・・・・・・・・・・・ 3059
PmxAdditionalFeatures.GetDeviceStatus Method ・・・・・・・・・・・・・・・・・・・・・・・・ 3060
PmxAdditionalFeatures.GetDigitalOutBitMask Method ・・・・・・・・・・・・・・・・・・・・・ 3061
PmxAdditionalFeatures.GetEngineeringUnit Method ・・・・・・・・・・・・・・・・・・・・・・ 3062
PmxAdditionalFeatures.GetLimitSwitch Method ・・・・・・・・・・・・・・・・・・・・・・・・ 3063
PmxAdditionalFeatures.GetMaxValue Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3064
PmxAdditionalFeatures.GetMinValue Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3065
PmxAdditionalFeatures.GetPeakToPeakValue Method ・・・・・・・・・・・・・・・・・・・・・・ 3066
PmxAdditionalFeatures.HoldPeakValues Method ・・・・・・・・・・・・・・・・・・・・・・・・ 3067
PmxAdditionalFeatures.LoadParameterSet Method ・・・・・・・・・・・・・・・・・・・・・・・ 3068
PmxAdditionalFeatures.ReceiveReply Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3069
PmxAdditionalFeatures.SendCommand Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3070
PmxAdditionalFeatures.SendCommandWithoutReply Method ・・・・・・・・・・・・・・・・・・・ 3071
Hbm.Api.Pmx Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3072
LimitSwitch Class ・・ 3073

LimitSwitch Members ・・・

PmxDevice.GetSupportedJackType Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3131
PmxDevice.GetSupportedSampleRates Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3132
PmxDevice.GetSupportedScalingTypes Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3133
PmxDevice.GetSupportedSensorTypes Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3134
PmxDevice.GetTimeSourceQuality Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3135
PmxDevice.HandleChannelsChangesTriggeredByTedsActivation Method ・・・・・・・・・・・・・・ 3136
PmxDevice.IsFirmwareVersionSupported Method ・・・・・・・・・・・・・・・・・・・・・・・・・ 3137
PmxDevice.MapConnectionInfo Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3138
PmxDevice.ReadSingleMeasurementValue Method ・・・・・・・・・・・・・・・・・・・・・・・・ 3139
PmxDevice.ReadSingleMeasurementValueOfAllSignals Method ・・・・・・・・・・・・・・・・・・ 3140
PmxDevice.ReadTeds Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3141
PmxDevice.Reconnect Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3142
PmxDevice.RefreshDeviceInfo Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3143
PmxDevice.RemoveInputSignalReference Method ・・・・・・・・・・・・・・・・・・・・・・・・ 3144
PmxDevice.Reset Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3145
PmxDevice.SetAnalogOut Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3146
PmxDevice.SetCanBusParameters Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3147
PmxDevice.SetCanChannelActivation Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3148
PmxDevice.SetChannelName Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3149
PmxDevice.SetDataRateDomain Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3150
PmxDevice.SetDeviceName Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3151
PmxDevice.SetDigitalIO Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3152
PmxDevice.SetInputSignalReference Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3153
SetShunt Method ・・ 3154
PmxDevice.SetShunt Method (Connector, ShuntMode, out List<Problem>) ・・・・・・・・・・・・ 3155
PmxDevice.SetShunt Method (List<Connector>, ShuntMode, out List<Problem>) ・・・・・・・・・ 3156
PmxDevice.SetSignalName Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3157
PmxDevice.SetTedsUsageMode Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3158
PmxDevice.SetUnit Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3159
PmxDevice.WriteTeds Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3160
PmxDevice Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3161
PmxDevice.AdditionalFeatures Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3162
PmxDevice.FamilyName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3163
PmxDevice.LogContextDevPmxStatus Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 3164
PmxDeviceFamily Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3165
PmxDeviceFamily Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3166
PmxDeviceFamily.PmxDeviceFamily Constructor ・・・・・・・・・・・・・・・・・・・・・・・・ 3167
PmxDeviceFamily Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3168
PmxDeviceFamily.Find Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3169
PmxDeviceFamily.GetPossibleDataAcquisitionModes Method ・・・・・・・・・・・・・・・・・・ 3170
PmxDeviceFamily.PrepareDataAcquisition Method ・・・・・・・・・・・・・・・・・・・・・・・ 3171
PmxDeviceFamily.Scan Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3172
PmxDeviceFamily Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3173
PmxDeviceFamily.FamilyName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3174
PmxDeviceFamily.IsScanSupported Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3175
PmxDeviceFamily.SkillReader Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3176
Hbm.Api.Pmx.Channels Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3177
PmxAnalogInChannel Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3178
PmxAnalogInChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3179
PmxAnalogInChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3180
PmxAnalogInChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3181
PmxAnalogInChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3182
PmxAnalogInChannel.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・ 3183

PmxAnalogInChannel.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3184
PmxAnalogInChannel.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3185
PmxAnalogInChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3186
PmxAnalogInChannel.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3187
PmxAnalogInChannel.TedsUsageMode Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3188
PmxAnalogOutChannel Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3189
PmxAnalogOutChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3190
PmxAnalogOutChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3191
PmxAnalogOutChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3192
PmxAnalogOutChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3193
PmxAnalogOutChannel.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・ 3194
PmxAnalogOutChannel.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3195
PmxAnalogOutChannel.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・ 3196
PmxAnalogOutChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3197
PmxAnalogOutChannel.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3198
PmxDigitalChannel Class ・・ 3199
PmxDigitalChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3200
PmxDigitalChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3201
PmxDigitalChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3202
PmxDigitalChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3203
PmxDigitalChannel.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・・ 3204
PmxDigitalChannel.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3205
PmxDigitalChannel.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・・ 3206
PmxDigitalChannel.Pin Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3207
PmxDigitalChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3208
PmxDigitalChannel.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3209
PmxVirtualChannel Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3210
PmxVirtualChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3211
PmxVirtualChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3212
PmxVirtualChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3213
PmxVirtualChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3214
PmxVirtualChannel.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・ 3215
PmxVirtualChannel.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3216
PmxVirtualChannel.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・ 3217
PmxVirtualChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3218
PmxVirtualChannel.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3219
IPmxChannel Interface ・・ 3220
IPmxChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3221
IPmxChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3222
IPmxChannel.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3223
Hbm.Api.Pmx.Connectors Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3224
PmxAnalogInConnector Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3225
PmxAnalogInConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3226
PmxAnalogInConnector Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3227
PmxAnalogInConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3228
PmxAnalogInConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3229
PmxAnalogInConnector.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・ 3230
PmxAnalogInConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・ 3231
PmxAnalogInConnector.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・ 3232
PmxAnalogInConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3233
PmxAnalogInConnector.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3234
PmxAnalogOutConnector Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3235
PmxAnalogOutConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3236

PmxAnalogOutConnector Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3237
PmxAnalogOutConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3238
PmxAnalogOutConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3239
PmxAnalogOutConnector.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・ 3240
PmxAnalogOutConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・ 3241
PmxAnalogOutConnector.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・ 3242
PmxAnalogOutConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3243
PmxAnalogOutConnector.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3244
PmxDigitalConnector Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3245
PmxDigitalConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3246
PmxDigitalConnector Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3247
PmxDigitalConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3248
PmxDigitalConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3249
PmxDigitalConnector.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・ 3250
PmxDigitalConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3251
PmxDigitalConnector.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・・ 3252
PmxDigitalConnector.Pin Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3253
PmxDigitalConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3254
PmxDigitalConnector.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3255
PmxVirtualConnector Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3256
PmxVirtualConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3257
PmxVirtualConnector Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3258
PmxVirtualConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3259
PmxVirtualConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3260
PmxVirtualConnector.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・ 3261
PmxVirtualConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3262
PmxVirtualConnector.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・・ 3263
PmxVirtualConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3264
PmxVirtualConnector.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3265
IPmxConnector Interface ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3266
IPmxConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3267
IPmxConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3268
IPmxConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3269
Hbm.Api.Pmx.Enums Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3270
Hbm.Api.Pmx.Enums.AmplifierType Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・ 3271
Hbm.Api.Pmx.Enums.EngineeringUnit Enumeration ・・・・・・・・・・・・・・・・・・・・・・・ 3273
Hbm.Api.Pmx.Enums.ExcitationAmplitude Enumeration ・・・・・・・・・・・・・・・・・・・・・ 3281
Hbm.Api.Pmx.Enums.FilterCharacteristic Enumeration ・・・・・・・・・・・・・・・・・・・・・ 3282
Hbm.Api.Pmx.Enums.FilterFrequency Enumeration ・・・・・・・・・・・・・・・・・・・・・・・ 3283
Hbm.Api.Pmx.Enums.InputType Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3285
Hbm.Api.Pmx.Enums.LimitSwitchOperatingDirection Enumeration ・・・・・・・・・・・・・・・・ 3286
Hbm.Api.Pmx.Enums.PmxDeviceStatus Enumeration ・・・・・・・・・・・・・・・・・・・・・・・ 3287
Hbm.Api.Pmx.Enums.PmxSensorType Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・ 3289
Hbm.Api.Pmx.Enums.SampleRate Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3291
Hbm.Api.Pmx.Enums.SsiBaudRate Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3293
Hbm.Api.Pmx.Enums.TransducerSensitivity Enumeration ・・・・・・・・・・・・・・・・・・・・・ 3294
Hbm.Api.Pmx.Signals Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3295
PmxAnalogInSignal Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3296
PmxAnalogInSignal Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3297
PmxAnalogInSignal Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3298
PmxAnalogInSignal.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3299
PmxAnalogInSignal Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3300
PmxAnalogInSignal.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3301

PmxAnalogInSignal.Slot Property ・・・

MgcDevice.AssignConnector Method ・・・

MgcDevice.SetShunt Method (Connector, ShuntMode, out List<Problem>) ・・・・・・・・・・・・ 3412
MgcDevice.SetShunt Method (List<Connector>, ShuntMode, out List<Problem>) ・・・・・・・・・・ 3413
MgcDevice.SetSignalName Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3414
MgcDevice.SetTedsUsageMode Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3415
MgcDevice.SetUnit Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3416
MgcDevice.WriteTeds Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3417
MgcDevice Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3418
MgcDevice.AdditionalFeatures Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3419
MgcDevice.FamilyName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3420
MgcDevice.MeasurementSignalType Propertys ・・・・・・・・・・・・・・・・・・・・・・・・・ 3421
MgcDeviceFamily Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3422
MgcDeviceFamily Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3423
MgcDeviceFamily.MgcDeviceFamily Constructor ・・・・・・・・・・・・・・・・・・・・・・・・ 3424
MgcDeviceFamily Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3425
MgcDeviceFamily.Find Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3426
MgcDeviceFamily.GetPossibleDataAcquisitionModes Method ・・・・・・・・・・・・・・・・・・ 3427
MgcDeviceFamily.PrepareDataAcquisition Method ・・・・・・・・・・・・・・・・・・・・・・・ 3428
MgcDeviceFamily.Scan Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3429
MgcDeviceFamily Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3430
MgcDeviceFamily.FamilyName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3431
MgcDeviceFamily.IsScanSupported Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3432
MgcDeviceFamily.SkillReader Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3433
MgcIndicationScalingValues Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3434
MgcIndicationScalingValues Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3435
MgcIndicationScalingValues Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3436
MgcIndicationScalingValues.InputRange Property ・・・・・・・・・・・・・・・・・・・・・・・ 3437
MgcIndicationScalingValues.ZeroPoint Property ・・・・・・・・・・・・・・・・・・・・・・・・ 3438
MgcLimitSwitch Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3439
MgcLimitSwitch Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3440
MgcLimitSwitch Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3441
MgcLimitSwitch.Enabled Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3442
MgcLimitSwitch.Hysteresis Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3443
MgcLimitSwitch.IgnoreMeasurementValueStatus Property ・・・・・・・・・・・・・・・・・・・・ 3444
MgcLimitSwitch.InputSignal Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3445
MgcLimitSwitch.InvertResetBehaviour Property ・・・・・・・・・・・・・・・・・・・・・・・・ 3446
MgcLimitSwitch.Limit Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3447
MgcLimitSwitch.LimitSwitchNumber Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 3448
MgcLimitSwitch.OperatingDirection Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 3449
MgcLimitSwitch.ResetBehaviorMask Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 3450
Hbm.Api.Mgc.Channels Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3451
MgcAnalogInChannel Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3452
MgcAnalogInChannel Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3453
MgcAnalogInChannel Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3454
MgcAnalogInChannel.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3455
MgcAnalogInChannel Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3456
MgcAnalogInChannel.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・ 3457
MgcAnalogInChannel.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 3458
MgcAnalogInChannel.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・・ 3459
MgcAnalogInChannel.ConnectionBoardSerialNumber Property ・・・・・・・・・・・・・・・・・・ 3460
MgcAnalogInChannel.ConnectionBoardType Property ・・・・・・・・・・・・・・・・・・・・・・ 3461
MgcAnalogInChannel.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3462
MgcAnalogInChannel.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3463
MgcAnalogInChannel.TedsUsageMode Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 3464

MgcAnalogOutChannel Class ・・・

IMgcChannel.ConnectionBoardType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3519
Hbm.Api.Mgc.Connectors Namespace ・・ 3520
MgcAnalogInConnector Class ・・ 3521
MgcAnalogInConnector Members ・・・ 3522
MgcAnalogInConnector Methods ・・・ 3523
MgcAnalogInConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3524
MgcAnalogInConnector Properties ・・ 3525
MgcAnalogInConnector.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3526
MgcAnalogInConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3527
MgcAnalogInConnector.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3528
MgcAnalogInConnector.ConnectionBoardSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・・・・ 3529
MgcAnalogInConnector.ConnectionBoardType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3530
MgcAnalogInConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3531
MgcAnalogInConnector.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3532
MgcAnalogOutConnector Class ・・ 3533
MgcAnalogOutConnector Members ・・ 3534
MgcAnalogOutConnector Methods ・・ 3535
MgcAnalogOutConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3537
MgcAnalogOutConnector.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3538
MgcAnalogOutConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3539
MgcAnalogOutConnector.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3540
MgcAnalogOutConnector.ConnectionBoardSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・・・ 3541
MgcAnalogOutConnector.ConnectionBoardType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3542
MgcAnalogOutConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3543
MgcAnalogOutConnector.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3544
MgcCanConnector Class ・・ 3545
MgcCanConnector Members ・・・ 3546
MgcCanConnector Methods ・・・ 3547
MgcCanConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3548
MgcCanConnector Properties ・・ 3549
MgcCanConnector.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3550
MgcCanConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3551
MgcCanConnector.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3552
MgcCanConnector.ConnectionBoardSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3553
MgcCanConnector.ConnectionBoardType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3554
MgcCanConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3555
MgcCanConnector.Slot Property ・・ 3556
MgcDigitalConnector Class ・・・ 3557
MgcDigitalConnector Members ・・・ 3558
MgcDigitalConnector Methods ・・・ 3559
MgcDigitalConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3560
MgcDigitalConnector Properties ・・ 3561
MgcDigitalConnector.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3562
MgcDigitalConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3563
MgcDigitalConnector.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3564
MgcDigitalConnector.ConnectionBoardSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・・・・ 3565
MgcDigitalConnector.ConnectionBoardType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3566
MgcDigitalConnector.Pin Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3567
MgcDigitalConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3568
MgcDigitalConnector.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3569
MgcVirtualConnector Class ・・・ 3570
MgcVirtualConnector Members ・・・ 3571
MgcVirtualConnector Methods ・・・ 3572

MgcVirtualConnector.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3573
MgcVirtualConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3574
MgcVirtualConnector.AmplifierSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・・・ 3575
MgcVirtualConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3576
MgcVirtualConnector.AmplifierTypeName Property ・・・・・・・・・・・・・・・・・・・・・・・・ 3577
MgcVirtualConnector.ConnectionBoardSerialNumber Property ・・・・・・・・・・・・・・・・・・・ 3578
MgcVirtualConnector.ConnectionBoardType Property ・・・・・・・・・・・・・・・・・・・・・・・ 3579
MgcVirtualConnector.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3580
MgcVirtualConnector.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3581
IMgcConnector Interface ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3582
IMgcConnector Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3583
IMgcConnector Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3584
IMgcConnector.AmplifierType Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3585
IMgcConnector.ConnectionBoardSerialNumber Property ・・・・・・・・・・・・・・・・・・・・・ 3586
IMgcConnector.ConnectionBoardType Property ・・・・・・・・・・・・・・・・・・・・・・・・・ 3587
Hbm.Api.Mgc.Enums Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3588
Hbm.Api.Mgc.Enums.AmplifierType Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・ 3589
Hbm.Api.Mgc.Enums.AnalogOutBesselFrequency Enumeration ・・・・・・・・・・・・・・・・・・・ 3591
Hbm.Api.Mgc.Enums.AnalogOutButterworthFrequency Enumeration ・・・・・・・・・・・・・・・・ 3593
Hbm.Api.Mgc.Enums.AnalogOutMode Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・ 3594
Hbm.Api.Mgc.Enums.BaudRate Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3596
Hbm.Api.Mgc.Enums.ByteOrder Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3597
Hbm.Api.Mgc.Enums.ConnectionBoardType Enumeration ・・・・・・・・・・・・・・・・・・・・・・ 3598
Hbm.Api.Mgc.Enums.DataFormat Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3602
Hbm.Api.Mgc.Enums.DeviceSynchronization Enumeration ・・・・・・・・・・・・・・・・・・・・・ 3603
Hbm.Api.Mgc.Enums.DigitalOutMode Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・ 3604
Hbm.Api.Mgc.Enums.DischargeTimeType Enumeration ・・・・・・・・・・・・・・・・・・・・・・・ 3605
Hbm.Api.Mgc.Enums.ExcitationAmplitude Enumeration ・・・・・・・・・・・・・・・・・・・・・・ 3606
Hbm.Api.Mgc.Enums.FilterCharacteristic Enumeration ・・・・・・・・・・・・・・・・・・・・・・ 3608
Hbm.Api.Mgc.Enums.FilterFrequency Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・ 3609
Hbm.Api.Mgc.Enums.FrameFormat Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3615
Hbm.Api.Mgc.Enums.InputType Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3616
Hbm.Api.Mgc.Enums.LimitSwitchOperatingDirection Enumeration ・・・・・・・・・・・・・・・・・ 3617
Hbm.Api.Mgc.Enums.MeasurementSignalType Enumeration ・・・・・・・・・・・・・・・・・・・・ 3618
Hbm.Api.Mgc.Enums.MgcSensorType Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・ 3620
Hbm.Api.Mgc.Enums.ModeSignalByteOrder Enumeration ・・・・・・・・・・・・・・・・・・・・・・ 3625
Hbm.Api.Mgc.Enums.SampleRate Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3626
Hbm.Api.Mgc.Enums.SignalType Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3628
Hbm.Api.Mgc.Enums.TimeChannelType Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・ 3629
Hbm.Api.Mgc.Enums.TransducerSensitivity Enumeration ・・・・・・・・・・・・・・・・・・・・・・ 3630
Hbm.Api.Mgc.Enums.TriggerOperationMode Enumeration ・・・・・・・・・・・・・・・・・・・・・・ 3631
Hbm.Api.Mgc.Enums.Wiring Enumeration ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3632
Hbm.Api.Mgc.Signals Namespace ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3633
MgcAnalogInSignal Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3634
MgcAnalogInSignal Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3635
MgcAnalogInSignal Methods ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3636
MgcAnalogInSignal.GetUniqueID Method ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3637
MgcAnalogInSignal Properties ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3638
MgcAnalogInSignal.IndicationScaling Property ・・・・・・・・・・・・・・・・・・・・・・・・・・ 3639
MgcAnalogInSignal.Position Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3640
MgcAnalogInSignal.Slot Property ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3641
MgcAnalogOutSignal Class ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3642
MgcAnalogOutSignal Members ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 3643

MgcAnalogOutSignal Methods ・・・

トピック

名前	説明
Hbm.Api.QuantumX Namespace	名前空間Hbm.Api.QuantumXには、主にQuantumXデバイスを使用して測定を設定および実行するための関数が含まれています。 したがって、抽象クラスHbm.Api.Common.Device(および抽象クラスHbm.Api.Common.Entities.DeviceFamily)を実装します。
Hbm.Api.Pmx Namespace	名前空間Hbm.Api.Pmxには、主にPMXデバイスを使用して測定を設定および実行するための関数が含まれています。 したがって、抽象クラスHbm.Api.Common.Device(および抽象クラスHbm.Api.Common.Entities.DeviceFamily)を実装します。
Hbm.Api.Mgc Namespace	名前空間Hbm.Api.Mgcには、主にMGCデバイスを使用して測定を設定および実行する関数が含まれています。 したがって、抽象クラスHbm.Api.Common.Device(および抽象クラスHbm.Api.Common.Entities.DeviceFamily)を実装します。

Hbm.Api.QuantumX Namespace

名前空間Hbm.Api.QuantumXには、主にQuantumXデバイスを使用して測定を設定および実行するための関数が含まれています。

したがって、抽象クラスHbm.Api.Common.Device(および抽象クラスHbm.Api.Common.Entities.DeviceFamily)を実装します。

クラス

名前	説明
Hbm.Api.QuantumX	これは名前空間Hbm.Api.QuantumXです。
Hbm.Api.QuantumX.Channels	これは名前空間Hbm.Api.QuantumX.Channelsです。
Hbm.Api.QuantumX.Connectors	これは名前空間Hbm.Api.QuantumX.Connectorsです。
Hbm.Api.QuantumX.Enums	これは名前空間Hbm.Api.QuantumX.Enumsです。
Hbm.Api.QuantumX.Exceptions	これは名前空間Hbm.Api.QuantumX.Exceptionsです。
Hbm.Api.QuantumX.Signals	これは名前空間Hbm.Api.QuantumX.Signalsです。
Hbm.Api.QuantumX.Statuses	これは名前空間Hbm.Api.QuantumX.Statusesです。

トピック

名前	説明
サポートされていないQuantumXの機能	次の表に、APIで現在サポートされていないいくつかの機能を示します。
QuantumXの追加機能	これらはQuantumXの追加機能です。また、デバイス経由でもアクセス可能です。これらにアクセスするには、共通デバイス(デバイススキャンで配信されるなど)をQuantumXDeviceにキャストする必要があります。QuantumXのすべての追加機能は、AdditionalFeaturesの下にグループ化されています。

サポートされていないQuantumXの機能

次の表に、APIで現在サポートされていないいくつかの機能を示します。

機能	説明
IRIGタイムサービスの構成	サポートされていません。
テスト信号	サポートされていません。
仮想チャネル	演算チャネルなどの仮想チャネルはサポートされていません。
ファームウェア	4.2.56以降でサポートされています。4.12.20でテスト済みです。

サポートされていないモジュール: MX590

QuantumXの追加機能

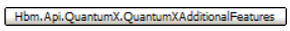
これらはQuantumXの追加機能です。また、デバイス経由でもアクセス可能です。これらにアクセスするには、共通デバイス(デバイススキャンで配信されるなど)をQuantumXDeviceにキャストする必要があります。QuantumXのすべての追加機能は、AdditionalFeaturesの下にグループ化されています。

クラス

名前	説明
QuantumXAdditionalFeatures	QuantumXDeviceのすべての追加機能を保持するクラス。追加機能は、Deviceクラスの抽象関数によって事前に定義されていないすべてのパブリック関数をカバーします。

QuantumXAdditionalFeatures Class

QuantumXDeviceのすべての追加機能を保持するクラス。追加機能は、Deviceクラスの抽象関数によって事前に定義されていないすべてのパブリック関数をカバーします。



C++

```
class QuantumXAdditionalFeatures;
```

C#

```
public class QuantumXAdditionalFeatures;
```

Visual Basic

```
Public Class QuantumXAdditionalFeatures
```

ファイル

QuantumXAdditionalFeatures.cs

備考

注意: 同じデバイスへのマルチスレッド アクセスを防ぐために、すべてのパブリックメソッドを同期する必要があります。

QuantumXAdditionalFeatures Members

次の表は、QuantumXAdditionalFeaturesによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	AssignXmlView	xml Connectorビューを現在のデバイスに書き込みます。
	DisableConnector	デバイス上で指定したコネクタconnectorToDisableを無効にします。新しいHbm.Api.QuantumX.Connectors.QuantumXOffConnectorインスタンスが生成され、デバイスに割り当てられます。connectorToDisableコネクタは、デバイスのコネクタ リスト内の新しいコネクタdisabledConnectorに自動的に置き換えられます。
	DisableExtendedSampleRateMode	拡張サンプルレートモードを無効にし、現在のモジュールのすべてのコネクタに対して通常モードを再度有効にします。
	DisableIsochronousDataTransfer	信号のアイソクロナスデータ転送を無効にします。
	DisableLedBlinking	これは、DisableLedBlinkingメソッドの多重定義の概要です。
	EnableConnector	デバイス上で指定したoffConnectorToEnableコネクタを有効にします。新しいコネクタインスタンスがoffConnectorToEnableの位置に生成され、デバイスに割り当てられます。offConnectorToEnableコネクタは、デバイスのコネクタ リスト内の新しいコネクタEnabledConnectorに自動的に置き換えられます。
	EnableExtendedSampleRateMode	現在のモジュールの指定したコネクタに対して拡張サンプルレートモードを有効にします。
	EnableIsochronousDataTransfer	信号のアイソクロナスデータ転送を有効にします。
	EnableLedBlinking	これは、EnableLEDBlinkingメソッドの多重定義の概要です。
	GetDigitalIOInputThreshold	デジタルI/Oの入力しきい値を取得します。
	GetXmlView	指定したXMLビューのコンテンツをデバイスから読み取ります。
	QuantumXAdditionalFeatures	新しいインスタンスを作成し、データを初期化します。
	Reboot	デバイスを再起動します。
	SetDataRateDomain	現在のモジュールに指定したdataRateDomainを設定します。重要: 実行の成功後、ユーザは手動またはRebootを呼び出してモジュールを再起動する必要があります。
	SetDigitalIOInputThreshold	デジタル I/Oの入力しきい値を設定します。

QuantumXAdditionalFeatures.QuantumXAdditionalFeatures Constructor

新しいインスタンスを作成し、データを初期化します。

C++

```
QuantumXAdditionalFeatures(
    QuantumXDevice quantumXDevice
);
```

C#

```
public QuantumXAdditionalFeatures(
    QuantumXDevice quantumXDevice
);
```

Visual Basic

```
Public Sub New(
    quantumXDevice As QuantumXDevice
)
```

引数

引数	説明
quantumXDevice	追加機能をサポートするQuantumXDevice。

QuantumXAdditionalFeatures Methods

QuantumXAdditionalFeaturesクラスのメソッドの一覧です。

Public Methods

	名前	説明
	AssignXmlView	xml Connectorビューを現在のデバイスに書き込みます。
	DisableConnector	デバイス上で指定したコネクタconnectorToDisableを無効にします。新しいHbm.Api.QuantumX.Connectors.QuantumXOffConnectorインスタンスが生成され、デバイスに割り当てられます。connectorToDisableコネクタは、デバイスのコネクタ リスト内の新しいコネクタdisabledConnectorに自動的に置き換えられます。
	DisableExtendedSampleRateMode	拡張サンプルレートモードを無効にし、現在のモジュールのすべてのコネクタに対して通常モードを再度有効にします。
	DisableIsochronousDataTransfer	信号のアイソクロナスデータ転送を無効にします。
	DisableLedBlinking	これは、DisableLedBlinkingメソッドの多重定義の概要です。
	EnableConnector	デバイス上で指定したoffConnectorToEnableコネクタを有効にします。新しいコネクタインスタンスがoffConnectorToEnableの位置に生成され、デバイスに割り当てられます。offConnectorToEnableコネクタは、デバイスのコネクタ リスト内の新しいコネクタEnabledConnectorに自動的に置き換えられます。
	EnableExtendedSampleRateMode	現在のモジュールの指定したコネクタに対して拡張サンプルレートモードを有効にします。
	EnableIsochronousDataTransfer	信号のアイソクロナスデータ転送を有効にします。
	EnableLedBlinking	これは、EnableLEDBlinkingメソッドの多重定義の概要です。
	GetDigitalIOInputThreshold	デジタルI/Oの入力しきい値を取得します。
	GetXmlView	指定したXMLビューのコンテンツをデバイスから読み取ります。
	Reboot	デバイスを再起動します。
	SetDataRateDomain	現在のモジュールに指定したdataRateDomainを設定します。重要: 実行の成功後、ユーザは手動またはRebootを呼び出してモジュールを再起動する必要があります。
	SetDigitalIOInputThreshold	デジタル I/Oの入力しきい値を設定します。

QuantumXAdditionalFeatures.AssignXmlView Method

xml Connectorビューを現在のデバイスに書き込みます。

C++

```
bool AssignXmlView(
    string xmlViewContent,
    XmlViewType view,
    out List<Problem> problems
);
```

C#

```
public bool AssignXmlView(
    string xmlViewContent,
    XmlViewType view,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function AssignXmlView(
    xmlViewContent As string,
    view As XmlViewType,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
xmlViewContent	割り当てられる完全なXMLビューコンテンツを含むString。
view	xmlViewContentが属するビューの型。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合はtrue、それ以外の場合はfalse。

現在位置: Supported devices -> Hbm.Api.QuantumX Namespace ->
サポートされていないQuantumXの機能 -> QuantumXAdditionalFeatures Class ->
QuantumXAdditionalFeatures.QuantumXAdditionalFeatures Constructor

QuantumXAdditionalFeatures.DisableConnector Method

デバイス上で指定したコネクタconnectorToDisableを無効にします。新しいHbm.Api.QuantumX.Connectors.QuantumXOffConnectorインスタンスが生成され、デバイスに割り当てられます。connectorToDisableコネクタは、デバイスのコネクタ リスト内の新しいコネクタdisabledConnectorに自動的に置き換えられます。

C++

```
bool DisableConnector(
    IQuantumXConnector connectorToDisable,
    out QuantumXOffConnector disabledConnector,
    out List<Problem> problems
);
```

C#

```
public bool DisableConnector(
    IQuantumXConnector connectorToDisable,
    out QuantumXOffConnector disabledConnector,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function DisableConnector(
    connectorToDisable As IQuantumXConnector,
    disabledConnector As out QuantumXOffConnector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connectorToDisable	モジュールで無効にするコネクタ。
disabledConnector	コネクタへの詳細な参照としての無効なコネクタの新しいインスタンス。
problems	コネクタを無効にする際に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXAdditionalFeatures.DisableExtendedSampleRateMode Method

拡張サンプルレートモードを無効にし、現在のモジュールのすべてのコネクタに対して通常モードを再度有効にします。

C++

```
bool DisableExtendedSampleRateMode(
    out List<Problem> problems
);
```

C#

```
public bool DisableExtendedSampleRateMode(
    out List<Problem> problems
);
```

Visual Basic

```
Public Function DisableExtendedSampleRateMode(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	操作中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXAdditionalFeatures.DisableIsochronousDataTransfer Method

信号のアイソクロナスデータ転送を無効にします。

C++

```
bool DisableIsochronousDataTransfer(
    IQuantumXIsochronousSignal signal,
    out List<Problem> problems
);
```

C#

```
public bool DisableIsochronousDataTransfer(
    IQuantumXIsochronousSignal signal,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function DisableIsochronousDataTransfer(
    signal As IQuantumXIsochronousSignal,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
signal	フラグを設定する信号。
problems	データレートドメインの設定中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

DisableLedBlinking Method

これは、DisableLedBlinkingメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	QuantumXAdditionalFeatures.DisableLedBlinking ()	(廃止)現在のモジュールのLEDの点滅を無効にします。
	QuantumXAdditionalFeatures.DisableLedBlinking (Connector)	(廃止)指定したコネクタのLED点滅を無効にします。

QuantumXAdditionalFeatures.DisableLedBlinking Method ()

(廃止)現在のモジュールのLEDの点滅を無効にします。

C++

```
[Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)]
void DisableLedBlinking();
```

C#

```
[Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)]
public void DisableLedBlinking();
```

Visual Basic

```
<Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)>
Public Function DisableLedBlinking() As void
```

備考

廃止: 代わりにCommonメソッドHbm.Api.Common.Entities.Device.FlashLed(...)を使用してください。

例外

例外	説明
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.CommunicationFailedException	デバイスとの通信エラーが発生した場合にスローされます。

QuantumXAdditionalFeatures.DisableLedBlinking Method (Connector)

(廃止)指定したコネクタのLED点滅を無効にします。

C++

```
[Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)]
void DisableLedBlinking(
Connector connector
);
```

C#

```
[Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)]
public void DisableLedBlinking(
Connector connector
);
```

Visual Basic

```
<Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)>
Public Function DisableLedBlinking(
   connector As Connector
) As void
```

備考

廃止: 代わりにCommonメソッドHbm.Api.Common.Entities.Device.FlashLed(...)を使用してください。

例外

例外	説明
Hbm.Api.Common.Exceptions.UnknownConnectorException	コネクタがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.CommunicationFailedException	デバイスとの通信エラーが発生した場合にスローされます。

QuantumXAdditionalFeatures.EnableConnector Method

デバイス上で指定したoffConnectorToEnableコネクタを有効にします。新しいコネクタインスタンスがoffConnectorToEnableの位置に生成され、デバイスに割り当てられます。offConnectorToEnableコネクタは、デバイスのコネクタ リスト内の新しいコネクタEnabledConnectorに自動的に置き換えられます。

C++

```
bool EnableConnector(
    QuantumXOffConnector offConnectorToEnable,
    out Connector enabledConnector,
    out List<Problem> problems,
    QuantumXEnableConnectorType connectorType = QuantumXEnableConnectorType.Default
);
```

C#

```
public bool EnableConnector(
    QuantumXOffConnector offConnectorToEnable,
    out Connector enabledConnector,
    out List<Problem> problems,
    QuantumXEnableConnectorType connectorType
);
```

Visual Basic

```
Public Function EnableConnector(
    offConnectorToEnable As QuantumXOffConnector,
    enabledConnector As out QuantumXOffConnector,
    problems As out List<Problem>,
    connectorType As QuantumXEnableConnectorType = QuantumXEnableConnectorType.Default
) As bool
```

引数

引数	説明
offConnectorToEnable	モジュールで有効にするコネクタ。
enabledConnector	コネクタへの詳細な参照として、Hbm.Api.QuantumX.Connectors.QuantumXAnalogInConnectorまたはHbm.Api.QuantumX.Connectors.QuantumXAnalogOutConnectorタイプの有効なコネクタ(出力パラメータ)。
problems	コネクタを有効にする際に発生した警告とエラー。
connectorType	有効にした場合のコネクタの型(オプション)。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXAdditionalFeatures.EnableExtendedSampleRateMode Method

現在のモジュールの指定したコネクタに対して拡張サンプルレート モードを有効にします。

C++

```
bool EnableExtendedSampleRateMode(
    IQuantumXConnector connectorsForExtendedSampleRate,
    out List<Problem> problems
);
```

C#

```
public bool EnableExtendedSampleRateMode(
    IQuantumXConnector connectorsForExtendedSampleRate,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function EnableExtendedSampleRateMode(
    connectorsForExtendedSampleRate As IQuantumXConnector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connectorsForExtendedSampleRate	拡張サンプルレート モード用のコネクタ。
problems	拡張サンプルレート モードの設定中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

現在、MX403、MX410、MX411のみが拡張サンプルレートモードに対応しています。最大2つのコネクタのみを有効にでき、その他はすべて無効(オフ)になります。

現在位置: Supported devices -> Hbm.Api.QuantumX Namespace ->
サポートされていないQuantumXの機能 -> QuantumXAdditionalFeatures Class ->
QuantumXAdditionalFeatures.QuantumXAdditionalFeatures Constructor

QuantumXAdditionalFeatures.EnableIsochronousDataTransfer Method

信号のアイソクロナスデータ転送を有効にします。

C++

```
bool EnableIsochronousDataTransfer(
    IQuantumXIsochronousSignal signal,
    out List<Problem> problems
);
```

C#

```
public bool EnableIsochronousDataTransfer(
    IQuantumXIsochronousSignal signal,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function EnableIsochronousDataTransfer(
    signal As IQuantumXIsochronousSignal,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
signal	フラグを設定する信号。
problems	データレートドメインの設定中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

EnableLedBlinking Method

これは、EnableLEDBlinkingメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	QuantumXAdditionalFeatures.EnableLedBlinking (Connector, LedBlinking)	(廃止)指定したコネクタのLED点滅を有効にします。
	QuantumXAdditionalFeatures.EnableLedBlinking (LedBlinking)	(廃止)現在のモジュールのLEDの点滅を有効にします。

現在位置: Supported devices -> Hbm.Api.QuantumX Namespace ->
サポートされていないQuantumXの機能 -> QuantumXAdditionalFeatures Class ->
QuantumXAdditionalFeatures.QuantumXAdditionalFeatures Constructor

QuantumXAdditionalFeatures.EnableLedBlinking Method (Connector, LedBlinking)

(廃止)指定したコネクタのLED点滅を有効にします。

C++

```
[Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)]
void EnableLedBlinking(
    Connector connector,
    LedBlinking ledBlinking = LedBlinking.Orange
);
```

C#

```
[Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)]
public void EnableLedBlinking(
    Connector connector,
    LedBlinking ledBlinking
);
```

Visual Basic

```
<Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)>
Public Function EnableLedBlinking(
   connector As Connector,
   ledBlinking As LedBlinking = LedBlinking.Orange
) As void
```

引数

引数	説明
connector	LEDの点滅を有効にするコネクタ。
ledBlinking	LED点滅の色(オプション)、デフォルトは橙色点滅です。

備考

廃止: 代わりにCommonメソッドHbm.Api.Common.Entities.Device.FlashLed(...)を使用してください。

例外

例外	説明
Hbm.Api.Common.Exceptions.UnknownConnectorException	コネクタがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.CommunicationFailedException	デバイスとの通信エラーが発生した場合にスローされます。

QuantumXAdditionalFeatures.EnableLedBlinking Method (LedBlinking)

(廃止)現在のモジュールのLEDの点滅を有効にします。

C++

```
[Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)]
void EnableLedBlinking(
    LedBlinking ledBlinking = LedBlinking.Orange
);
```

C#

```
[Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)]
public void EnableLedBlinking(
    LedBlinking ledBlinking
);
```

Visual Basic

```
<Obsolete("Please use common method Hbm.Api.Common.Entities.Device.FlashLed(...) instead.",
false)>
Public Function EnableLedBlinking(
   ledBlinking As LedBlinking = LedBlinking.Orange
) As void
```

引数

引数	説明
ledBlinking	LED点滅の色(オプション)、デフォルトは橙色点滅です。

備考

廃止: 代わりにCommonメソッドHbm.Api.Common.Entities.Device.FlashLed(...)を使用してください。

例外

例外	説明
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.CommunicationFailedException	デバイスとの通信エラーが発生した場合にスローされます。

QuantumXAdditionalFeatures.GetDigitalIOInputThreshold Method

デジタルI/Oの入力しきい値を取得します。

C++

```
DigitalIoInputThreshold GetDigitalIOInputThreshold();
```

C#

```
public DigitalIoInputThreshold GetDigitalIOInputThreshold();
```

Visual Basic

```
Public Function GetDigitalIOInputThreshold() As DigitalIoInputThreshold
```

戻り値

入力しきい値。

備考

ReSharperはInconsistentNamingを無効にします。

QuantumXAdditionalFeatures.GetXmlView Method

指定したXMLビューのコンテンツをデバイスから読み取ります。

C++

```
string GetXmlView(
    XmlViewType view
);
```

C#

```
public string GetXmlView(
    XmlViewType view
);
```

Visual Basic

```
Public Function GetXmlView(
    view As XmlViewType
) As string
```

引数

引数	説明
view	xmlとして取得するビュー。

戻り値

ビューのXMLコンテンツは`string`として返し、デバイスがビューをサポートしていない場合、`String.Empty`になります。

例外

例外	説明
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.CommunicationFailedException	デバイスとの通信エラーが発生した場合にスローされます。

QuantumXAdditionalFeatures.Reboot Method

デバイスを再起動します。

C++

```
void Reboot();
```

C#

```
public void Reboot();
```

Visual Basic

```
Public Function Reboot() As void
```

例外

例外	説明
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.CommunicationFailedException	デバイスとの通信エラーが発生した場合にスローされます。

QuantumXAdditionalFeatures.SetDataRateDomain Method

現在のモジュールに指定したdataRateDomainを設定します。重要: 実行の成功後、ユーザは手動またはRebootを呼び出してモジュールを再起動する必要があります。

C++

```
bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

C#

```
public bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function SetDataRateDomain(
    dataRateDomain As DataRateDomainType,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
dataRateDomain	モジュールに設定するデータレートドメインタイプ。
problems	データレートドメインの設定中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

現在位置: Supported devices -> Hbm.Api.QuantumX Namespace ->
サポートされていないQuantumXの機能 -> QuantumXAdditionalFeatures Class ->
QuantumXAdditionalFeatures.QuantumXAdditionalFeatures Constructor

QuantumXAdditionalFeatures.SetDigitalIOInputThreshold Method

デジタル I/Oの入力しきい値を設定します。

C++

```
bool SetDigitalIOInputThreshold(
    DigitalIoInputThreshold inputThreshold,
    out List<Problem> problems
);
```

C#

```
public bool SetDigitalIOInputThreshold(
    DigitalIoInputThreshold inputThreshold,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function SetDigitalIOInputThreshold(
    inputThreshold As DigitalIoInputThreshold,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
inputThreshold	入力しきい値。
problems	データレートドメインの設定中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

ReSharperはInconsistentNamingを無効にします。

Hbm.Api.QuantumX Namespace

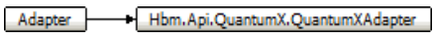
これは名前空間Hbm.Api.QuantumXです。

クラス

名前	説明
QuantumXAdapter	QuantumXデバイスのネットワークアダプタを表します。
QuantumXDevice	QuantumXデバイスを表します。
QuantumXDeviceFamily	QuantumXデバイスの具体的なデバイスファミリの実装。

QuantumXAdapter Class

QuantumXデバイスのネットワークアダプタを表します。



C++

```
class QuantumXAdapter : public Adapter;
```

C#

```
internal class QuantumXAdapter : Adapter;
```

Visual Basic

```
Friend Class QuantumXAdapter
Inherits Adapter
```

ファイル

QuantumXAdapter.cs

QuantumXAdapter Members

次の表は、QuantumXAdapterによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	RouterUUID	ルータのUUID。

QuantumXAdapter Properties

QuantumXAdapterクラスのプロパティの一覧です。

Internal Properties

	名前	説明
	RouterUUID	ルータのUUID。

QuantumXAdapter.RouterUUID Property

ルータのUUID。

C++

```
__property string RouterUUID;
```

C#

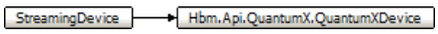
```
internal string RouterUUID;
```

Visual Basic

```
Friend Property RouterUUID() As string
```

QuantumXDevice Class

QuantumXデバイスを表します。



C++

```
class QuantumXDevice : public StreamingDevice;
```

C#

```
public sealed class QuantumXDevice : StreamingDevice;
```

Visual Basic

```
Public sealed Class QuantumXDevice
Inherits StreamingDevice
```

ファイル

QuantumXDevice.cs

備考

注意: 同一デバイスのマルチスレッドアクセスを防ぐために、すべてのパブリックメソッドはHbm.Api.Common.Entities.Device.SyncLockを使用して同期する必要があります。

QuantumXDevice Members

次の表は、QuantumXDeviceによって公開されるメンバーの一覧です。

Internal Methods

	名前	説明
	GetAllStandardConnectors	オブジェクトモデルからすべての標準コネクタを取得します。

Protected Methods

	名前	説明
	GetOverflowValue	QuantumXデバイスのオーバーフロー値。この値以上のすべての測定値は、Hbm.Api.Common.Enums.MeasurementValueState.Overflowとしてマークされます。
	HandleChannelsChangesTriggeredByTedsActivation	TEDSのアクティブ化によってトリガされるチャネルの変更を処理します。
	MapConnectionInfo	指定したConnectionInfoをデバイスの可能なConnectionInfoにマップします。これが不可能な場合、関数はnullを返します。
	PrepareDaq	連続測定セッションを準備します。
	Reconnect	基盤となる物理デバイスに再接続します。
	RefreshDeviceInfo	Name、SerialNoなどの関連するデバイスプロパティを更新します。この関数は、すべてのコネクタを読み取らずにデバイス自体に関する情報を取得する場合に使用します。
	SupportsPacketRateChange	QuantumXはパケットレートの変更をサポートしています。

Public Methods

	名前	説明
	ActivateTeds	指定したコネクタでTEDS設定をアクティブにし、新しいチャネルパラメータを読み込みます。
	AssignAutoCalibrationMode	指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当て処理中の警告とエラーは、problemsの一覧にまとめられます。
	AssignChannel	指定したチャネル(センサ、信号などを含む)の設定をデバイスの物理チャネルに割り当てます。割り当て処理中の警告とエラーは、problemsの一覧にまとめられます。
	AssignConnector	指定したコネクタ(チャネル、センサ、信号などを含む)の設定をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。
	AssignSensor	指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。
	AssignSignal	指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。
	AssignTimeSource	TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーはproblemsの一覧に収集されます。
	AssignZero	指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。
	AutoDetectFbgBands	
	ChangeIp	

	名前	説明
	ChangingConnectorType	コネクタの型を変更します。変更後、デバイスのオブジェクトモデルを更新するために、デバイスを切断して再度接続する必要があります。
	DisableRouting	ルーティング機能を無効にします。
	EnableRouting	ルーティング機能を有効にします。
	FlashLed	これはFlashLedメソッドの多重定義の概要です。
	GetAmplifierSerialNumber	チャネルのアンプのシリアル番号を取得します。
	GetAmplifierType	チャネルのアンプの種類を取得します。
	GetCalibrationCertificateFileInfos	要求したリリースタイプに応じて、キャリブレーション証明ファイル情報の一覧を返します。キャリブレーション証明は、指定した出力ディレクトリに作成/ダウンロードされます。
	GetChannelHardwareInfo	チャネルのハードウェア情報(モデルやModuleLabelなど)を取得します。
	GetDeviceErrorStatusList	これはGetDeviceErrorStatusListメソッドの多重定義の概要です。
	GetDeviceStatusList	これはGetDeviceStatusListメソッドの多重定義の概要です。
	GetPossibleDataAcquisitionModes	デバイスの現在の同期設定に応じて可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブルを介して確立される)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedが含まれます。デバイスが時間ベースの同期(PTPやNTPなど)を使用する場合、返されるリストにはモードDataAcquisitionMode.TimestampSynchronizedが含まれます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはモードDataAcquisitionMode.Unsynchronizedが含まれる可能性が高くなります。デバイスが接続されていない場合、可能なデータ収集モードのリストは空になります。
	GetShunt	これは GetShuntメソッドの多重定義の概要です。
	GetSupportedCanBitRates	サポートされているCANビットレートを取得します。
	GetSupportedFilterFrequencies	指定したフィルタでサポートされているフィルタ周波数のリストを取得します。
	GetSupportedFilters	指定した信号に対してサポートされているフィルタのリストを取得します。
	GetSupportedJackType	指定した信号に対してサポートされているジャックの種類を取得します。
	GetSupportedSampleRates	サポートされているサンプルレートのリストを取得します。
	GetSupportedScalingTypes	指定したチャネルでサポートされているスケーリングタイプのリストを取得します。
	GetSupportedSensorTypes	指定したチャネルでサポートされているセンサタイプのリストを取得します
	GetTimeSourceQuality	オフセット(同期タイムソースとデバイスの時刻の間のタイムスパン)をミリ秒単位で返し、TimeSource依存文字列と同期の品質に関する詳細情報を返します。NtpTimeSourceの場合、この文字列には標準パラメータ(カンマ区切り)を含める必要があります: "remote, refid, st, t, when, poll, reach, delay, offset, jitter"。
	IsFirmwareVersionSupported	指定したファームウェアバージョンがサポートされているかどうかを判断します。
	QuantumXDevice	これは、QuantumXDeviceコンストラクタの多重定義の概要です。
	ReadFbgSpectrum	
	ReadSingleElectricalValue	

	名前	説明
	ReadSingleMeasurementValue	DAQセッションの初期化を完了せずに、指定した各信号の単一の測定値を取得します。フィルタ設定は無視されます。測定値は信号に内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことでそこから取得できます。
	ReadSingleMeasurementValueOfAllSignals	DAQセッションの初期化を完了せずに、デバイスのすべての信号の単一の測定値を取得します。フィルタ設定は無視されます。測定値は信号に内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことでそこから取得できます。
	ReadTeds	
	RemoveInputSignalReference	
	Reset	デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数が呼び出されると、デバイスは切断されます。これは、デバイスのパラメータ化が工場出荷時の既定値を読み込むことによって変更され、デバイスの関連付けられたオブジェクトモデルと一致しなくなるために使用です(この関数を実装する場合は、必ずデバイスを切断し、Hbm.Api.Common.Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知してください)。
	ResetCanController	これは、QuantumXDeviceクラスのメンバーであるResetCanControllerです。
	SetAnalogOut	アナログ出力信号を指定した値に設定します。
	SetCanBusParameters	指定したCanConnectorのすべてのCAN busパラメータ(CanConnector.BitRate、CanConnector.IsTerminated、CanConnector.SupplyVoltageなど)を割り当てますが、チャネルは割り当てません。
	SetCanChannelActivation	指定したチャネルのリストをアクティブ化または非アクティブ化します。アクティブ化または非アクティブ化できるのは、CanInChannelとCanOutChannelのみです。
	SetChannelName	
	SetDataRateDomain	データレートドメインを設定します。
	SetDeviceName	物理デバイスのモジュール/デバイス名を設定し、成功した場合はオブジェクトモデルも更新します(Device.Name)。
	SetDigitalIO	指定した値に従ってデジタル出力信号を設定します。デジタル入力信号はハードウェアでは設定できません。
	SetFbgBandsDetectionResult	
	SetInputSignalReference	入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。
	SetShunt	これは、SetShuntメソッドの多重定義の概要です。
	SetSignalName	指定した信号に信号名を割り当てます。
	SetTedsUsageMode	TEDS使用モードを設定し、コネクタ設定を再読み取りします。
	SetUnit	指定したチャネルに単位を割り当てます。
	SupportsExtendedSampleRateMode	デバイスが拡張サンプルレートモードをサポートしているかどうかを決定します。
	UpdateCanConnectorsSettings	デバイスからすべてのCAN コネクタを読み取り、その設定を更新します。
	UpdateConnectorSettings	デバイスから提供されたIQuantumXConnector設定を更新します。
	WriteTeds	

Public Properties

	名前	説明
	AdditionalFeatures	QuantumXデバイスの追加機能。
	Extension	システムの基本モジュールタイプの拡張。
R	FamilyName	このデバイスが属するファミリの名前。
	FpgaVersion	プラットフォームのFPGAバージョン。
	HardwareControllerDate	プラットフォームのハードウェアコントローラの日付。
	HardwareControllerProgramDate	プラットフォームのハードウェアコントローラプログラムの日付。
	HardwareControllerProgramVersion	プラットフォームのハードウェアコントローラプログラム バージョン。
	HardwareControllerVersion	プラットフォームのハードウェアコントローラバージョン。
	HardwareIdentification	プラットフォームのハードウェアIDバージョン。
R	HasAdditionalTedsFieldsSupport	デバイス ファミリが校正とシリアル番号のTEDSフィールドをサポートしているかどうかを示す値を取得します。
R	IsAutomaticTedsMonitoringSupported	このインスタンスが自動TEDSモニタリングをサポートしているかどうかを示す値を取得します(QuantumXの場合は常に`true`)。
R	IsElectricalSnapshotSupported	このインスタンスが電気値の要求をサポートしているかどうかを示す値を取得します。
	IsFirmwareUpdateSupported	このインスタンスがファームウェアの更新をサポートしているかどうかを示す値を取得または設定します。
R	IsReadingFbgSpectrumSupported	このインスタンスがスペクトルの読み取りをサポートしているかどうかを示す値を取得します。
	ModuleId	システムのモジュールID。
	ModuleLabel	システムのモジュール ラベル。
	Revision	システムのリビジョン。
	Selection	システムの基本モジュールタイプの仕様。
	SpecialFirmwareVersion	システムの特別なファームウェアバージョン。
	SpecialVersion	システムの特別バージョン。
	SupportsCanBusTermination	デバイスがCAN busターミネーションをサポートしているかどうかを示す値を取得します。
R	SupportsChangingConnectorType	デバイスがCAN bus間での切り替えをサポートしているかどうかを示す値を取得します。

QuantumXDevice Constructor

これは、QuantumXDeviceコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	QuantumXDevice.QuantumXDevice ()	クラスの新しいインスタンスを作成し、通信マネージャ、追加機能などを初期化します。デバイスがスキャンメカニズムで検出できる場合は、このコンストラクタを使用します。
	QuantumXDevice.QuantumXDevice (string, int, int, int)	クラスの新しいインスタンスを作成し、通信マネージャ、追加機能などを初期化します。スキャンメカニズムを使用せずにデバイスを接続する必要がある場合、または既知のIPアドレスを持つデバイスに接続する場合は、このコンストラクタを使用します。

QuantumXDevice.QuantumXDevice Constructor ()

クラスの新しいインスタンスを作成し、通信マネージャ、追加機能などを初期化します。デバイスがスキャンメカニズムで検出できる場合は、このコンストラクタを使用します。

C++

```
QuantumXDevice();
```

C#

```
public QuantumXDevice();
```

Visual Basic

```
Public Sub New()
```

QuantumXDevice.QuantumXDevice Constructor (string, int, int, int)

クラスの新しいインスタンスを作成し、通信マネージャ、追加機能などを初期化します。スキャンメカニズムを使用せずにデバイスを接続する必要がある場合、または既知のIPアドレスを持つデバイスに接続する場合は、このコンストラクタを使用します。

C++

```
QuantumXDevice(
    string ipAddress,
    int port = CONNECTION_DEFAULT_PORT,
    int streamingPort = EthernetConnectionInfo.INVALID_PORT,
    int httpPort = EthernetConnectionInfo.INVALID_PORT
);
```

C#

```
public QuantumXDevice(
    string ipAddress,
    int port,
    int streamingPort,
    int httpPort
);
```

Visual Basic

```
Public Sub New(
    ipAddress As string,
    port As Integer = CONNECTION_DEFAULT_PORT,
    streamingPort As Integer = EthernetConnectionInfo.INVALID_PORT,
    httpPort As Integer = EthernetConnectionInfo.INVALID_PORT
)
```

引数

引数	説明
ipAddress	デバイスのIPアドレス。
port	デバイスを接続するためのポート(オプション)。デバイスがCX27に接続されている場合、このパラメータはオプションではありません。このパラメータが設定されていない場合は、デフォルトの接続ポートを使用します。
streamingPort	ストリーミングポート(スキャンが利用可能な場合はオプション)。このパラメータが設定されていない場合は、デバイス接続時に設定されます(スキャン結果によって取得します)。スキャンメカニズムが利用できない場合は、このパラメータはオプションではありません(不明な場合は、デフォルトのHbm.Api.Common.Entities.ConnectionInfos.StreamingConnectionInfo.STREAMING_DEFAULT_PORTを使用するようにしてください)。
httpPort	ストリーミングに必要なHTTP ポート(スキャンが利用可能な場合はオプション)。このパラメータが設定されていない場合は、デバイス接続時に設定されます(スキャン結果によって取得します)。スキャンメカニズムが利用できない場合は、このパラメータはオプションではありません(不明な場合は、デフォルトのHbm.Api.Common.Entities.ConnectionInfos.StreamingConnectionInfo.HTTP_DEFAULT_PORTを使用するようにしてください)。

QuantumXDevice Methods

次の表は、QuantumXDeviceクラスのメソッドの一覧です。

Internal Methods

	名前	説明
	GetAllStandardConnectors	オブジェクトモデルからすべての標準コネクタを取得します。

Protected Methods

	名前	説明
	GetOverflowValue	QuantumXデバイスのオーバーフロー値。この値以上のすべての測定値は、Hbm.Api.Common.Enums.MeasurementValueState.Overflowとしてマークされます。
	HandleChannelsChangesTriggeredByTedsActivation	TEDSのアクティブ化によってトリガされるチャネルの変更を処理します。
	MapConnectionInfo	指定したConnectionInfoをデバイスの可能なConnectionInfoにマップします。これが不可能な場合、関数はnullを返します。
	PrepareDaq	連続測定セッションを準備します。
	Reconnect	基盤となる物理デバイスに再接続します。
	RefreshDeviceInfo	Name、SerialNoなどの関連するデバイスプロパティを更新します。この関数は、すべてのコネクタを読み取らずにデバイス自体に関する情報を取得する場合に使用します。
	SupportsPacketRateChange	QuantumXはパケットレートの変更をサポートしています。

Public Methods

	名前	説明
	ActivateTeds	指定したコネクタでTEDS設定をアクティブにし、新しいチャネルパラメータを読み込みます。
	AssignAutoCalibrationMode	指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当て処理中の警告とエラーは、problemsの一覧にまとめられます。
	AssignChannel	指定したチャネル(センサ、信号などを含む)の設定をデバイスの物理チャネルに割り当てます。割り当て処理中の警告とエラーは、problemsの一覧にまとめられます。
	AssignConnector	指定したコネクタ(チャネル、センサ、信号などを含む)の設定をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。
	AssignSensor	指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。
	AssignSignal	指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。
	AssignTimeSource	TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーはproblemsの一覧に収集されます。
	AssignZero	指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。
	AutoDetectFbgBands	
	ChangeIp	

	名前	説明
	ChangingConnectorType	コネクタの型を変更します。変更後、デバイスのオブジェクトモデルを更新するために、デバイスを切断して再度接続する必要があります。
	DisableRouting	ルーティング機能を無効にします。
	EnableRouting	ルーティング機能を有効にします。
	FlashLed	これはFlashLedメソッドの多重定義の概要です。
	GetAmplifierSerialNumber	チャネルのアンプのシリアル番号を取得します。
	GetAmplifierType	チャネルのアンプの種類を取得します。
	GetCalibrationCertificateFileInfos	要求したリリースタイプに応じて、キャリブレーション証明ファイル情報の一覧を返します。キャリブレーション証明は、指定した出力ディレクトリに作成/ダウンロードされます。
	GetChannelHardwareInfo	チャネルのハードウェア情報(モデルやModuleLabelなど)を取得します。
	GetDeviceErrorStatusList	これはGetDeviceErrorStatusListメソッドの多重定義の概要です。
	GetDeviceStatusList	これはGetDeviceStatusListメソッドの多重定義の概要です。
	GetPossibleDataAcquisitionModes	デバイスの現在の同期設定に応じて可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブルを介して確立される)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedが含まれます。デバイスが時間ベースの同期(PTPやNTPなど)を使用する場合、返されるリストにはモードDataAcquisitionMode.TimestampSynchronizedが含まれます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはモードDataAcquisitionMode.Unsynchronizedが含まれる可能性が高くなります。デバイスが接続されていない場合、可能なデータ収集モードのリストは空になります。
	GetShunt	これは GetShuntメソッドの多重定義の概要です。
	GetSupportedCanBitRates	サポートされているCANビットレートを取得します。
	GetSupportedFilterFrequencies	指定したフィルタでサポートされているフィルタ周波数のリストを取得します。
	GetSupportedFilters	指定した信号に対してサポートされているフィルタのリストを取得します。
	GetSupportedJackType	指定した信号に対してサポートされているジャックの種類を取得します。
	GetSupportedSampleRates	サポートされているサンプルレートのリストを取得します。
	GetSupportedScalingTypes	指定したチャネルでサポートされているスケーリングタイプのリストを取得します。
	GetSupportedSensorTypes	指定したチャネルでサポートされているセンサタイプのリストを取得します
	GetTimeSourceQuality	オフセット(同期タイムソースとデバイスの時刻の間のタイムスパン)をミリ秒単位で返し、TimeSource依存文字列と同期の品質に関する詳細情報を返します。NtpTimeSourceの場合、この文字列には標準パラメータ(カンマ区切り)を含める必要があります: "remote, refid, st, t, when, poll, reach, delay, offset, jitter"。
	IsFirmwareVersionSupported	指定したファームウェアバージョンがサポートされているかどうかを判断します。
	ReadFbgSpectrum	
	ReadSingleElectricalValue	

	名前	説明
	ReadSingleMeasurementValue	DAQセッションの初期化を完了せずに、指定した各信号の単一の測定値を取得します。フィルタ設定は無視されます。測定値は信号に内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことでそこから取得できます。
	ReadSingleMeasurementValueOfAllSignals	DAQセッションの初期化を完了せずに、デバイスのすべての信号の単一の測定値を取得します。フィルタ設定は無視されます。測定値は信号に内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことでそこから取得できます。
	ReadTeds	
	RemoveInputSignalReference	
	Reset	デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数が呼び出されると、デバイスは切断されます。これは、デバイスのパラメータ化が工場出荷時の既定値を読み込むことによって変更され、デバイスの関連付けられたオブジェクトモデルと一致しなくなるために使用です(この関数を実装する場合は、必ずデバイスを切断し、Hbm.Api.Common.Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知してください)。
	ResetCanController	これは、QuantumXDeviceクラスのメンバーであるResetCanControllerです。
	SetAnalogOut	アナログ出力信号を指定した値に設定します。
	SetCanBusParameters	指定したCanConnectorのすべてのCAN busパラメータ(CanConnector.BitRate、CanConnector.IsTerminated、CanConnector.SupplyVoltageなど)を割り当てますが、チャネルは割り当てません。
	SetCanChannelActivation	指定したチャネルのリストをアクティブ化または非アクティブ化します。アクティブ化または非アクティブ化できるのは、CanInChannelとCanOutChannelのみです。
	SetChannelName	
	SetDataRateDomain	データレートドメインを設定します。
	SetDeviceName	物理デバイスのモジュール/デバイス名を設定し、成功した場合はオブジェクトモデルも更新します(Device.Name)。
	SetDigitalIO	指定した値に従ってデジタル出力信号を設定します。デジタル入力信号はハードウェアでは設定できません。
	SetFbgBandsDetectionResult	
	SetInputSignalReference	入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。
	SetShunt	これは、SetShuntメソッドの多重定義の概要です。
	SetSignalName	指定した信号に信号名を割り当てます。
	SetTedsUsageMode	TEDS使用モードを設定し、コネクタ設定を再読み取りします。
	SetUnit	指定したチャネルに単位を割り当てます。
	SupportsExtendedSampleRateMode	デバイスが拡張サンプルレートモードをサポートしているかどうかを決定します。
	UpdateCanConnectorsSettings	デバイスからすべてのCAN コネクタを読み取り、その設定を更新します。
	UpdateConnectorSettings	デバイスから提供されたIQuantumXConnector設定を更新します。
	WriteTeds	

QuantumXDevice.ActivateTeds Method

指定したコネクタでTEDS設定をアクティブにし、新しいチャネルパラメータを読み込みます。

C++

```
bool ActivateTeds(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override bool ActivateTeds(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function ActivateTeds(
    connector As Connector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	接続されたTEDSがアクティブ化されるコネクタ。
problems	空、またはアクティベーションプロセス中に発生した警告とエラーのリスト。

戻り値

TEDS設定が有効になっている場合は`true`、それ以外の場合は`false`。

備考

成功した場合、指定したコネクタのセンサオブジェクトと一部のチャネルプロパティが、TEDS設定に従って更新されたバージョンに置き換えられます。

QuantumXDevice.AssignAutoCalibrationMode Method

指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当て処理中の警告とエラーは、problemsの一覧にまとめられます。

C++

```
bool AssignAutoCalibrationMode(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignAutoCalibrationMode(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignAutoCalibrationMode(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

QuantumXDevice.AssignChannel Method

指定したチャネル(センサ、信号などを含む)の設定をデバイスの物理チャネルに割り当てます。割り当て処理中の警告とエラーは、problemsの一覧にまとめられます。

C++

```
bool AssignChannel(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignChannel(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignChannel(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	実際の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

QuantumXDevice.AssignConnector Method

指定したコネクタ(チャネル、センサ、信号などを含む)の設定をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。

C++

```
bool AssignConnector(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override bool AssignConnector(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignConnector(
    connector As Connector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	実際の(変更された)コネクタ。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
Hbm.Api.Common.Exceptions.UnknownConnectorException	コネクタがこのデバイスに属していない場合にスローされます。

QuantumXDevice.AssignSensor Method

指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。

C++

```
bool AssignSensor(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignSensor(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignSensor(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	実際の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
T:Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。
T:System.InvalidOperationException	チャネルのコネクタがT:Hbm.Api.QuantumX.Connectors.IQuantumXConnectorを実装していない場合にスローされます。

QuantumXDevice.AssignSignal Method

指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。

C++

```
bool AssignSignal(
    Signal signal,
    out List<Problem> problems
);
```

C#

```
public bool AssignSignal(
    Signal signal,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function AssignSignal(
    signal As Signal,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
signal	実際の(変更された)信号。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

例外

例外	説明
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

QuantumXDevice.AssignTimeSource Method

TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーはproblemsの一覧に収集されます。

C++

```
bool AssignTimeSource(
    out List<Problem> problems
);
```

C#

```
public bool AssignTimeSource(
    out List<Problem> problems
);
```

Visual Basic

```
Public Function AssignTimeSource(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXDevice.AssignZero Method

指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、problemsの一覧に収集されます。

C++

```
bool AssignZero(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignZero(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignZero(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	実際の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。
System.InvalidOperationException	チャネルのコネクタがHbm.Api.QuantumX.Connectors.IQuantumXConnectorを実装していない場合にスローされます。

QuantumXDevice.AutoDetectFbgBands Method

C++

```
FbgBandsDetectionResult AutoDetectFbgBands(
    FbgConnector fbgConnector,
    double threshold,
    double bandWidth
);
```

C#

```
public override FbgBandsDetectionResult AutoDetectFbgBands(
    FbgConnector fbgConnector,
    double threshold,
    double bandWidth
);
```

Visual Basic

```
Public override Function AutoDetectFbgBands(
    fbgConnector As FbgConnector,
    threshold As double,
    bandWidth As double
) As FbgBandsDetectionResult
```

QuantumXDevice.ChangeIp Method

C++

```
bool ChangeIp(
    string ipAddress,
    string subnetMask,
    bool isDhcp,
    string gateWay,
    unsigned tcpPort,
    string interfaceName,
    out List<Problem> problems
);
```

C#

```
public override bool ChangeIp(
    string ipAddress,
    string subnetMask,
    bool isDhcp,
    string gateWay,
    uint tcpPort,
    string interfaceName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function ChangeIp(
    ipAddress As string,
    subnetMask As string,
    isDhcp As bool,
    gateWay As string,
    tcpPort As UInteger,
    interfaceName As string,
    problems As out List<Problem>
) As bool
```

QuantumXDevice.ChangingConnectorType Method

コネクタの型を変更します。変更後、デバイスのオブジェクトモデルを更新するために、デバイスを切断して再度接続する必要があります。

C++

```
bool ChangingConnectorType(
    IQuantumXConnector quantumXConnector,
    ConnectorType connectorType,
    out List<Problem> problems
);
```

C#

```
public bool ChangingConnectorType(
    IQuantumXConnector quantumXConnector,
    ConnectorType connectorType,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function ChangingConnectorType(
    quantumXConnector As IQuantumXConnector,
    connectorType As out ConnectorType,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
quantumXConnector	種類を変更するIQuantumXConnector。
connectorType	変更後のコネクタの種類。
problems	変更する際にデバイスによって報告された警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

注意:この関数の実行後、デバイスを切断して再接続する必要があります。

例外

例外	説明
ArgumentNullException	quantumXConnector。

QuantumXDevice.DisableRouting Method

ルーティング機能を無効にします。

C++

```
bool DisableRouting(
    out List<Problem> problems
);
```

C#

```
public override bool DisableRouting(
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function DisableRouting(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

QuantumXDevice.EnableRouting Method

ルーティング機能を有効にします。

C++

```
bool EnableRouting(
    out List<Problem> problems
);
```

C#

```
public override bool EnableRouting(
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function EnableRouting(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

FlashLed Method

これはFlashLedメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	QuantumXDevice.FlashLed (Connector, LedFlashMode, out List<Problem>)	指定したコネクタのLEDを指定したモード/色で点滅させます。
	QuantumXDevice.FlashLed (LedFlashMode, out List<Problem>)	指定したモード/色でデバイスのLEDを点滅させます。
	QuantumXDevice.FlashLed (List<Connector>, LedFlashMode, out List<Problem>)	指定したコネクタのLEDを指定したモード/色で点滅させます。

QuantumXDevice.FlashLed Method (Connector, LedFlashMode, out List<Problem>)

指定したコネクタのLEDを指定したモード/色で点滅させます。

C++

```
bool FlashLed(
    Connector connector,
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool FlashLed(
    Connector connector,
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function FlashLed(
    connector As Connector,
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	点滅させるコネクタ。
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

QuantumXDevice.FlashLed Method (LedFlashMode, out List<Problem>)

指定したモード/色でデバイスのLEDを点滅させます。

C++

```
bool FlashLed(
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool FlashLed(
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function FlashLed(
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

QuantumXDevice.FlashLed Method (List<Connector>, LedFlashMode, out List<Problem>)

指定したコネクタのLEDを指定したモード/色で点滅させます。

C++

```
bool FlashLed(
    List<Connector> connectors,
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool FlashLed(
    List<Connector> connectors,
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function FlashLed(
    connectors As List<Connector>,
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connectors	点滅させるコネクタ。
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

QuantumXDevice.GetAllStandardConnectors Method

オブジェクトモデルからすべての標準コネクタを取得します。

C++

```
List<IQuantumXConnector> GetAllStandardConnectors();
```

C#

```
internal List<IQuantumXConnector> GetAllStandardConnectors();
```

Visual Basic

```
Friend Function GetAllStandardConnectors() As List<IQuantumXConnector>
```

戻り値

標準コネクタのリスト。

QuantumXDevice.GetAmplifierSerialNumber Method

チャネルのアンプのシリアル番号を取得します。

C++

```
string GetAmplifierSerialNumber(
    Channel channel
);
```

C#

```
public override string GetAmplifierSerialNumber(
    Channel channel
);
```

Visual Basic

```
Public override Function GetAmplifierSerialNumber(
    channel As Channel
) As string
```

引数

引数	説明
channel	アンプシリアル番号を返すチャネル。

戻り値

指定したチャネルのアンプのシリアル番号。

例外

例外	説明
NotImplementedException	

QuantumXDevice.GetAmplifierType Method

チャネルのアンプの種類を取得します。

C++

```
string GetAmplifierType(
    Channel channel
);
```

C#

```
public override string GetAmplifierType(
    Channel channel
);
```

Visual Basic

```
Public override Function GetAmplifierType(
    channel As Channel
) As string
```

引数

引数	説明
channel	アンプの種類を返すチャネル。

戻り値

指定したチャネルのアンプの種類。

QuantumXDevice.GetCalibrationCertificateFileInfos Method

要求したリリースタイプに応じて、キャリブレーション証明ファイル情報の一覧を返します。キャリブレーション証明は、指定した出力ディレクトリに作成/ダウンロードされます。

C++

```
List<CalibrationCertificateFileInfo> GetCalibrationCertificateFileInfos(
    ReleaseType releaseType,
    string outputDirectory,
    out List<Problem> problems
);
```

C#

```
public override List<CalibrationCertificateFileInfo> GetCalibrationCertificateFileInfos(
    ReleaseType releaseType,
    string outputDirectory,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetCalibrationCertificateFileInfos(
    releaseType As ReleaseType,
    outputDirectory As string,
    problems As out List<Problem>
) As List<CalibrationCertificateFileInfo>
```

引数

引数	説明
releaseType	要求された校正証明のリリースタイプ。
outputDirectory	校正証明書を保存/作成するディレクトリ。
problems	校正証明の取得中に発生した警告とエラー。

戻り値

校正証明書ファイル情報のリスト。

備考

一時的に作成された校正証明書(デバイスからダウンロード)は忘れずに削除してください。

QuantumXDevice.GetChannelHardwareInfo Method

チャネルのハードウェア情報(モデルやModuleLabelなど)を取得します。

C++

```
string GetChannelHardwareInfo(
    Channel channel
);
```

C#

```
public override string GetChannelHardwareInfo(
    Channel channel
);
```

Visual Basic

```
Public override Function GetChannelHardwareInfo(
    channel As Channel
) As string
```

引数

引数	説明
channel	ハードウェア情報を返すチャネル。

戻り値

指定したチャネルのハードウェア情報。

GetDeviceErrorStatusList Method

これはGetDeviceErrorStatusListメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	QuantumXDevice.GetDeviceErrorStatusList (Channel, out List<Problem>)	指定したチャネルのデバイスエラーステータスのリストを取得します。
	QuantumXDevice.GetDeviceErrorStatusList (Connector, out List<Problem>)	指定したコネクタのデバイスエラーステータスのリストを取得します。
	QuantumXDevice.GetDeviceErrorStatusList (out List<Problem>)	デバイスエラーステータスのリストを取得します。

QuantumXDevice.GetDeviceErrorStatusList Method (Channel, out List<Problem>)

指定したチャネルのデバイスエラーステータスのリストを取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceErrorStatusList(
    channel As Channel,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
channel	エラーステータスを取得するチャネル。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

指定したチャネルのデバイスエラーステータスのリスト。

QuantumXDevice.GetDeviceErrorStatusList Method (Connector, out List<Problem>)

指定したコネクタのデバイスエラーステータスのリストを取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceErrorStatusList(
    connector As Connector,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
connector	エラーステータスを取得するコネクタ。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

指定したコネクタのデバイスエラーステータスのリスト。

QuantumXDevice.GetDeviceErrorStatusList Method (out List<Problem>)

デバイスエラーステータスのリストを取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    out List<Problem> problems
);
```

C#

```
public override List<DeviceErrorStatus> GetDeviceErrorStatusList(
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceErrorStatusList(
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

デバイスエラーステータスのリスト。

GetDeviceStatusList Method

これはGetDeviceStatusListメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	QuantumXDevice.GetDeviceStatusList (Channel, out List<Problem>)	指定したチャネルのデバイスステータスのリストを取得します。
	QuantumXDevice.GetDeviceStatusList (Connector, out List<Problem>)	指定したコネクタのデバイスステータスのリストを取得します。
	QuantumXDevice.GetDeviceStatusList (out List<Problem>, StatusType)	デバイスステータスのリストを取得します。

QuantumXDevice.GetDeviceStatusList Method (Channel, out List<Problem>)

指定したチャネルのデバイスステータスのリストを取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceStatus> GetDeviceStatusList(
    Channel channel
    out List<Problem> problems,
);
```

Visual Basic

```
Public override Function GetDeviceStatusList(
    channel As Channel,
    problems As out List<Problem>
) As List<DeviceStatus>
```

引数

引数	説明
channel	ステータスを取得するチャネル。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

指定したチャネルのデバイスステータスのリスト。

QuantumXDevice.GetDeviceStatusList Method (Connector, out List<Problem>)

指定したコネクタのデバイスステータスのリストを取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceStatus> GetDeviceStatusList(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceStatusList(
    connector As Connector,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
connector	ステータスを取得するコネクタ。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

指定したコネクタのデバイスステータスのリスト。

QuantumXDevice.GetDeviceStatusList Method (out List<Problem>, StatusType)

デバイスステータスのリストを取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    out List<Problem> problems,
    StatusType statusTypesToGet = StatusType.All
);
```

C#

```
public override List<DeviceStatus> GetDeviceStatusList(
    out List<Problem> problems,
    StatusType statusTypesToGet
);
```

Visual Basic

```
Public override Function GetDeviceStatusList(
    problems As out List<Problem>,
    statusTypesToGet As StatusType = StatusType.All
) As List<DeviceStatus>
```

引数

引数	説明
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。
statusTypesToGet	要求したステータスタイプを含むT:Hbm.Api.Common.Enums.StatusType(デフォルトはすべてのステータス情報)。

戻り値

指定したチャネルのデバイスステータスのリスト。

QuantumXDevice.GetOverflowValue Method

QuantumXデバイスのオーバーフロー値。この値以上のすべての測定値は、Hbm.Api.Common.Enums.MeasurementValueState.Overflowとしてマークされます。

C++

```
double GetOverflowValue();
```

C#

```
protected override double GetOverflowValue();
```

Visual Basic

```
Protected override Function GetOverflowValue() As double
```

戻り値

デバイスのオーバーフロー値。

QuantumXDevice.GetPossibleDataAcquisitionModes Method

デバイスの現在の同期設定に応じて可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブルを介して確立される)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedが含まれます。デバイスが時間ベースの同期(PTPやNTPなど)を使用する場合、返されるリストにはモードDataAcquisitionMode.TimestampSynchronizedが含まれます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはモードDataAcquisitionMode.Unsynchronizedが含まれる可能性が高くなります。デバイスが接続されていない場合、可能なデータ収集モードのリストは空になります。

C++

```
List<DataAcquisitionMode> GetPossibleDataAcquisitionModes();
```

C#

```
public override List<DataAcquisitionMode> GetPossibleDataAcquisitionModes();
```

Visual Basic

```
Public override Function GetPossibleDataAcquisitionModes() As List<DataAcquisitionMode>
```

戻り値

現在の同期設定に応じて、このデバイスに使用できるDataAcquisitionModeのリスト。

GetShunt Method

これは GetShuntメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	QuantumXDevice.GetShunt (Connector)	指定したコネクタのShuntModeを取得します。
	QuantumXDevice.GetShunt (List<Connector>)	指定したコネクタのShuntModeを取得します。

QuantumXDevice.GetShunt Method (Connector)

指定したコネクタのShuntModeを取得します。

C++

```
ShuntMode GetShunt(
    Connector connector
);
```

C#

```
public override ShuntMode GetShunt(
    Connector connector
);
```

Visual Basic

```
Public override Function GetShunt(
   connector As Connector
) As ShuntMode
```

引数

引数	説明
connector	ShuntModeを取得するコネクタ。

戻り値

指定されたコネクタのShuntMode。

QuantumXDevice.GetShunt Method (List<Connector>)

指定したコネクタのShuntModeを取得します。

C++

```
List<ShuntMode> GetShunt(
    List<Connector> connectors
);
```

C#

```
public override List<ShuntMode> GetShunt(
    List<Connector> connectors
);
```

Visual Basic

```
Public override Function GetShunt(
   connectors As List<Connector>
) As List<ShuntMode>
```

引数

引数	説明
connectors	ShuntModeを取得するコネクタのリスト。

戻り値

指定されたコネクタのShuntModeのリスト。

QuantumXDevice.GetSupportedCanBitRates Method

サポートされているCANビットレートを取得します。

C++

```
List<int> GetSupportedCanBitRates();
```

C#

```
public override List<int> GetSupportedCanBitRates();
```

Visual Basic

```
Public override Function GetSupportedCanBitRates() As List<int>
```

戻り値

サポートしている CANビットレートのリスト。

QuantumXDevice.GetSupportedFilterFrequencies Method

指定したフィルタでサポートされているフィルタ周波数のリストを取得します。

C++

```
List<decimal> GetSupportedFilterFrequencies(
    Filter Filter,
    bool isOffineQuery = false
);
```

C#

```
public override List<decimal> GetSupportedFilterFrequencies(
    Filter Filter,
    bool isOffineQuery
);
```

Visual Basic

```
Public override Function GetSupportedFilterFrequencies(
   Filter As Filter
   isOffineQuery As bool = false
) As List<decimal>
```

引数

引数	説明
filter	取得したいフィルタの対象となるフィルタ周波数。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なフィルタ周波数のリスト。

QuantumXDevice.GetSupportedFilters Method

指定した信号に対してサポートされているフィルタのリストを取得します。

C++

```
List<Filter> GetSupportedFilters(
    Signal signal,
    bool isOfflineQuery = false
);
```

C#

```
public override List<Filter> GetSupportedFilters(
    Signal signal,
    bool isOfflineQuery
);
```

Visual Basic

```
Public override Function GetSupportedFilters(
   signal As Signal
   isOfflineQuery As bool = false
) As List<Filter>
```

引数

引数	説明
signal	取得したいフィルタの対象となる信号。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なフィルタのリスト。

QuantumXDevice.GetSupportedJackType Method

指定した信号に対してサポートされているジャックの種類を取得します。

C++

```
string GetSupportedJackType(
    Signal signal,
    bool isOffineQuery = false
);
```

C#

```
public override string GetSupportedJackType(
    Signal signal,
    bool isOffineQuery
);
```

Visual Basic

```
Public override Function GetSupportedJackType(
   signal As Signal
   isOffineQuery As bool = false
) As List<Filter>
```

引数

引数	説明
signal	信号。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

ジャックの種類名。

QuantumXDevice.GetSupportedSampleRates Method

サポートされているサンプルレートのリストを取得します。

C++

```
List<decimal> GetSupportedSampleRates(
    Signal signal,
    bool isOfflineQuery = false
);
```

C#

```
public override List<decimal> GetSupportedSampleRates(
    Signal signal,
    bool isOfflineQuery
);
```

Visual Basic

```
Public override Function GetSupportedSampleRates(
   signal As Signal
   isOfflineQuery As bool = false
) As List<decimal>
```

引数

引数	説明
signal	取得したいサンプルレートの信号。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なサンプルレートのリスト。

QuantumXDevice.GetSupportedScalingTypes Method

指定したチャネルでサポートされているスケーリングタイプのリストを取得します。

C++

```
List<ScalingType> GetSupportedScalingTypes(
    Channel channel,
    bool isOffineQuery = false
);
```

C#

```
public override List<ScalingType> GetSupportedScalingTypes(
    Channel channel,
    bool isOffineQuery
);
```

Visual Basic

```
Public override Function GetSupportedScalingTypes(
    channel As Channel,
   isOffineQuery As bool = false
) As List<ScalingType>
```

引数

引数	説明
channel	取得したいスケーリングタイプのチャネル。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なスケーリングタイプのリスト。

戻り値

可能なスケーリングタイプは、チャネルの現在のセンサによっても異なります。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

QuantumXDevice.GetSupportedSensorTypes Method

指定したチャネルでサポートされているセンサタイプのリストを取得します。

C++

```
List<SensorType> GetSupportedSensorTypes(
    Channel channel,
    bool isOfflineQuery
);
```

C#

```
public override List<SensorType> GetSupportedSensorTypes(
    Channel channel,
    bool isOfflineQuery
);
```

Visual Basic

```
Public override Function GetSupportedSensorTypes(
    channel As Channel,
   isOfflineQuery As bool
) As List<SensorType>
```

引数

引数	説明
channel	取得したいセンサタイプのチャネル。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なセンサタイプのリスト。

QuantumXDevice.GetTimeSourceQuality Method

オフセット(同期タイムソースとデバイスの時刻の間のタイムスパン)をミリ秒単位で返し、TimeSource依存文字列と同期の品質に関する詳細情報を返します。NtpTimeSourceの場合、この文字列には標準パラメータ(カンマ区切り)を含める必要があります: "remote, refid, st, t, when, poll, reach, delay, offset, jitter"。

C++

```
void GetTimeSourceQuality(
    out double offsetInMs,
    out string quality
);
```

C#

```
public override void GetTimeSourceQuality(
    out double offsetInMs,
    out string quality
);
```

Visual Basic

```
Public override Function GetTimeSourceQuality(
    offsetInMs As out double,
    quality As out string
) As void
```

引数

引数	説明
offsetInMs	タイムソースへのオフセット(ミリ秒単位)。オフセットを決定できない場合は double.MaxValue になります。
quality	TimeSourceタイプに依存する同期品質に関する情報。

例外

例外	説明
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.CommunicationFailedException	デバイスとの通信エラーが発生した場合にスローされます。

QuantumXDevice.HandleChannelsChangesTriggeredByTedsActivation Method

TEDSのアクティブ化によってトリガされるチャネルの変更を処理します。

C++

```
void HandleChannelsChangesTriggeredByTedsActivation(
    Dictionary<Connector, Channel> channels,
    List<Problem> problems
);
```

C#

```
public override void HandleChannelsChangesTriggeredByTedsActivation(
    Dictionary<Connector, Channel> channels,
    List<Problem> problems
);
```

Visual Basic

```
Public override Function HandleChannelsChangesTriggeredByTedsActivation(
    channels As Dictionary<Connector, Channel>,
    problems As List<Problem>
) As void
```

引数

引数	説明
channels	チャネル。
problems	警告とエラー。

QuantumXDevice.IsFirmwareVersionSupported Method

指定したファームウェアバージョンがサポートされているかどうかを判断します。

C++

```
bool IsFirmwareVersionSupported(
    string firmwareVersion
);
```

C#

```
public override bool IsFirmwareVersionSupported(
    string firmwareVersion
);
```

Visual Basic

```
Public override Function IsFirmwareVersionSupported(
    firmwareVersion As string
) As bool
```

引数

引数	説明
firmwareVersion	確認するファームウェアのバージョン。

戻り値

指定したファームウェアバージョンがサポートされていない場合はfalseになります。

備考

この関数は、指定したファームウェアバージョンが最新の既知のバージョンよりも高い場合(APIがリリースされた時点)でもtrueを返します。

QuantumXDevice.MapConnectionInfo Method

指定したConnectionInfoをデバイスの可能なConnectionInfoにマップします。これが不可能な場合、関数はnullを返します。

C++

```
ConnectionInfo MapConnectionInfo(
    ConnectionInfo connectionInfo
);
```

C#

```
public override ConnectionInfo MapConnectionInfo(
    ConnectionInfo connectionInfo
);
```

Visual Basic

```
Public override Function MapConnectionInfo(
    connectionInfo As ConnectionInfo
) As ConnectionInfo
```

引数

引数	説明
connectionInfo	サポートされているConnectionInfoにマップするConnectionInfo。

戻り値

デバイスの有効なConnectionInfo、またはマッピングが不可能な場合はnull。

例

ConnectionInfoがEthernetConnectionInfoであるが、デバイスがStreamingConnectionInfoのみをサポートしている場合: 関数はEthernetConnectionInfoをStreamingConnectionInfoにマッピングし、不足しているパラメータ(可能な場合、デフォルトのストリーミングパラメータなど)を入力します。これが不可能な場合、関数はnullを返す必要があります。

QuantumXDevice.PrepareDaq Method

連続測定セッションを準備します。

C++

```
void PrepareDaq(
    List<Signal> measurementSignals
);
```

C#

```
public override void PrepareDaq(
    List<Signal> measurementSignals
);
```

Visual Basic

```
Public override Function PrepareDaq(
    measurementSignals As List<Signal>
) As void
```

引数

引数	説明
measurementSignals	測定に使用する信号。

例外

例外	説明
Hbm.Api.Common.Exceptions.ParameterizationNotCompletedException	measurementSignalsには、内部でパラメータ化がまだ実行中の信号が含まれています。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

QuantumXDevice.ReadFbgSpectrum Method

C++

```
FbgSpectrum ReadFbgSpectrum(
    FbgSpectrumType fbgSpectrumType,
    FbgConnector fbgConnector
);
```

C#

```
public override FbgSpectrum ReadFbgSpectrum(
    FbgSpectrumType fbgSpectrumType,
    FbgConnector fbgConnector
);
```

Visual Basic

```
Public override Function ReadFbgSpectrum(
    fbgSpectrumType As FbgSpectrumType,
    fbgConnector As FbgConnector
) As FbgSpectrum
```

QuantumXDevice.ReadSingleElectricalValue Method

C++

```
double ReadSingleElectricalValue(
    AnalogInSignal signal
);
```

C#

```
public override double ReadSingleElectricalValue(
    AnalogInSignal signal
);
```

Visual Basic

```
Public override Function ReadSingleElectricalValue(
    signal As AnalogInSignal
) As double
```

戻り値

デバイスのオーバーフロー値。

QuantumXDevice.ReadSingleMeasurementValue Method

DAQセッションの初期化を完了せずに、指定した各信号の単一の測定値を取得します。フィルタ設定は無視されます。測定値は信号に内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことでそこから取得できます。

C++

```
void ReadSingleMeasurementValue(
    List<Signal> measurementSignals
);
```

C#

```
public override void ReadSingleMeasurementValue(
    List<Signal> measurementSignals
);
```

Visual Basic

```
Public override Function ReadSingleMeasurementValue(
    measurementSignals As List<Signal>
) As void
```

引数

引数	説明
measurementSignals	測定値を取得する信号。

備考

デバイスが単一の測定値のタイムスタンプをサポートしていない場合、PCの現在の時刻を使用してタイムスタンプを生成します。

例外

例外	説明
DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
CommunicationFailedException	デバイスとの通信エラーが発生した場合にスローされます。

QuantumXDevice.ReadSingleMeasurementValueOfAllSignals Method

DAQセッションの初期化を完了せずに、デバイスのすべての信号の単一の測定値を取得します。フィルタ設定は無視されます。測定値は信号に内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことでそこから取得できます。

C++

```
void ReadSingleMeasurementValueOfAllSignals();
```

C#

```
public override void ReadSingleMeasurementValueOfAllSignals();
```

Visual Basic

```
Public override Function ReadSingleMeasurementValueOfAllSignals() As void
```

備考

デバイスが単一の測定値のタイムスタンプをサポートしていない場合、PCの現在の時刻を使用してタイムスタンプを生成します。

例外

例外	説明
DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

QuantumXDevice.ReadTeds Method

C++

```
Sensor ReadTeds(
    Connector connector,
    out string tid,
    out List<Problem> problems
);
```

C#

```
public override Sensor ReadTeds(
    Connector connector,
    out string tid,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function ReadTeds(
    connector As Connector,
    tid As out string,
    problems As out List<Problem>
) As Sensor
```

QuantumXDevice.Reconnect Method

基盤となる物理デバイスに再接続します。

C++

```
bool Reconnect(
    out List<Problem> problems
);
```

C#

```
protected override bool Reconnect(
    out List<Problem> problems
);
```

Visual Basic

```
Protected override Function Reconnect(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	再接続プロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXDevice.RefreshDeviceInfo Method

Name、SerialNoなどの関連するデバイスプロパティを更新します。この関数は、すべてのコネクタを読み取らずにデバイス自体に関する情報を取得する場合に使用します。

C++

```
bool RefreshDeviceInfo(
    List<Problem> problems
);
```

C#

```
protected override bool RefreshDeviceInfo(
    List<Problem> problems
);
```

Visual Basic

```
Protected override Function RefreshDeviceInfo(
    problems As List<Problem>
) As bool
```

引数

引数	説明
problems	更新プロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXDevice.RemoveInputSignalReference Method

C++

```
bool RemoveInputSignalReference(
    IInputSignalReference channel,
    out List<Problem> problems
);
```

C#

```
public override bool RemoveInputSignalReference(
    IInputSignalReference channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function RemoveInputSignalReference(
    channel As IInputSignalReference
    problems As out List<Problem>
) As bool
```

QuantumXDevice.Reset Method

デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数が呼び出されると、デバイスは切断されます。これは、デバイスのパラメータ化が工場出荷時の既定値を読み込むことによって変更され、デバイスの関連付けられたオブジェクトモデルと一致しなくなるために使用です(この関数を実装する場合は、必ずデバイスを切断し、Hbm.Api.Common.Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知してください)。

C++

```
void Reset(
    out List<Problem> problems
);
```

C#

```
public override void Reset(
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function Reset(
    problems As out List<Problem>
) As void
```

引数

引数	説明
problems	デバイスのリセット中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

この関数の実行後にデバイスが切断され、一定時間アクセス不能になる可能性があることに注意してください。

QuantumXDevice.ResetCanController Method

これは、QuantumXDeviceクラスのメンバーであるResetCanControllerです。

C++

```
bool ResetCanController(
    IQuantumXConnector quantumXConnector,
    out List<Problem> problems
);
```

C#

```
public bool ResetCanController(
    IQuantumXConnector quantumXConnector,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function ResetCanController(
    quantumXConnector As IQuantumXConnector,
    problems As out List<Problem>
) As bool
```

QuantumXDevice.SetAnalogOut Method

アナログ出力信号を指定した値に設定します。

C++

```
void SetAnalogOut(
    AnalogOutSignal analogOutSignal,
    decimal value
);
```

C#

```
public override void SetAnalogOut(
    AnalogOutSignal analogOutSignal,
    decimal value
);
```

Visual Basic

```
Public override Function SetAnalogOut(
    analogOutSignal As AnalogOutSignal,
    value out decimal
) As void
```

引数

引数	説明
analogOutSignal	設定するアナログ出力信号。
value	設定する値。

例外

例外	説明
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。
System.InvalidOperationException	チャネルのコネクタがHbm.Api.QuantumX.Connectors.QuantumXAnalogOutConnectorでない場合にスローされます。
Hbm.Api.Common.Exceptions.CommunicationFailedException	デバイスとの通信エラーが発生した場合にスローされます。

QuantumXDevice.SetCanBusParameters Method

指定したCanConnectorのすべてのCAN busパラメータ(CanConnector.BitRate、CanConnector.IsTerminated、CanConnector.SupplyVoltageなど)を割り当てますが、チャネルは割り当てません。

C++

```
bool SetCanBusParameters(
    CanConnector canConnector,
    out List<Problem> problems
);
```

C#

```
public override bool SetCanBusParameters(
    CanConnector canConnector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetCanBusParameters(
    canConnector As CanConnector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
canConnector	プロパティを割り当てる必要がある既存の(変更された)CanConnector。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXDevice.SetCanChannelActivation Method

指定したチャネルのリストをアクティブ化または非アクティブ化します。アクティブ化または非アクティブ化できるのは、CanInChannelとCanOutChannelのみです。

C++

```
bool SetCanChannelActivation(
    List<Channel> canChannels,
    bool activate,
    out List<Problem> problems
);
```

C#

```
public override bool SetCanChannelActivation(
    List<Channel> canChannels,
    bool activate,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetCanChannelActivation(
    canChannels As List<Channel>,
    activate As bool,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
canChannels	アクティブ化/非アクティブ化する必要があるCANチャネル。
activate	指定したチャネルをアクティブにする場合はtrue、それ以外の場合はfalse。
problems	アクティブ化/非アクティブ化プロセス中に発生した警告とエラーのリスト。

戻り値

TEDS設定が有効になっている場合はtrue、それ以外の場合はfalse。

QuantumXDevice.SetChannelName Method

C++

```
bool SetChannelName(
    Channel channel,
    string channelName,
    out List<Problem> problems
);
```

C#

```
public override bool SetChannelName(
    Channel channel,
    string channelName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetChannelName(
    channel As Channel,
    channelName As string,
    problems As out List<Problem>
) As bool
```

QuantumXDevice.SetDataRateDomain Method

データレートドメインを設定します。

C++

```
bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

C#

```
public override bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetDataRateDomain(
    dataRateDomain As DataRateDomainType,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
dataRateDomain	データレートドメイン。
problems	警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXDevice.SetDeviceName Method

物理デバイスのモジュール/デバイス名を設定し、成功した場合はオブジェクトモデルも更新します(Device.Name)。

C++

```
bool SetDeviceName(
    string newName,
    out List<Problem> problems
);
```

C#

```
public override bool SetDeviceName(
    string newName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetDeviceName(
    newName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
newName	新しい名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXDevice.SetDigitalIO Method

指定した値に従ってデジタル出力信号を設定します。デジタル入力信号はハードウェアでは設定できません。

C++

```
void SetDigitalIO(
    DigitalSignal digitalSignal,
    DigitalValueType value
);
```

C#

```
public override void SetDigitalIO(
    DigitalSignal digitalSignal,
    DigitalValueType value
);
```

Visual Basic

```
Public override Function SetDigitalIO(
    digitalSignal As DigitalSignal,
    value As out DigitalValueType
) As void
```

引数

引数	説明
digitalSignal	調整するデジタル信号。
value	設定する値。

例外

例外	説明
DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。
InvalidOperationException	信号のコネクタがQuantumXDigitalConnectorでない場合にスローされます。 信号のコネクタの方向がDirectionType.Outでない場合にスローされます。
CommunicationFailedException	デバイスとの通信エラーが発生した場合にスローされます。

QuantumXDevice.SetFbgBandsDetectionResult Method

C++

```
bool SetFbgBandsDetectionResult(
    FbgConnector fbgConnector,
    FbgBandsDetectionResult fbgBandsDetectionResult,
    out List<Problem> problems
);
```

C#

```
public override bool SetFbgBandsDetectionResult(
    FbgConnector fbgConnector,
    FbgBandsDetectionResult fbgBandsDetectionResult,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetFbgBandsDetectionResult(
    fbgConnector As FbgConnector,
    fbgBandsDetectionResult As FbgBandsDetectionResult,
    problems As out List<Problem>
) As bool
```

QuantumXDevice.SetInputSignalReference Method

入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。

C++

```
bool SetInputSignalReference(
    IInputSignalReference channel,
    Device inputSignalDevice,
    Signal inputSignal,
    out List<Problem> problems
);
```

C#

```
public override bool SetInputSignalReference(
    IInputSignalReference channel,
    Device inputSignalDevice,
    Signal inputSignal,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetInputSignalReference(
    channel As IInputSignalReference
    inputSignalDevice As Device
    inputSignal As Signal
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	入力信号リファレンスで設定される(出力)チャネル。
inputSignalDevice	入力信号のデバイス。
inputSignal	入力信号
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

SetShunt Method

これは、SetShuntメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	QuantumXDevice.SetShunt (Connector, ShuntMode, out List<Problem>)	指定したコネクタのシャント出力の構成を設定します。
	QuantumXDevice.SetShunt (List<Connector>, ShuntMode, out List<Problem>)	指定したコネクタのシャント出力の構成を設定します。

QuantumXDevice.SetShunt Method (Connector, ShuntMode, out List<Problem>)

指定したコネクタのシャント出力の構成を設定します。

C++

```
bool SetShunt(
    Connector connector,
    ShuntMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool SetShunt(
    Connector connector,
    ShuntMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetShunt(
    connector As Connector,
    mode As ShuntMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	シャント出力を設定するコネクタ。
mode	指定したコネクタのShuntMode。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXDevice.SetShunt Method (List<Connector>, ShuntMode, out List<Problem>)

指定したコネクタのシャント出力の構成を設定します。

C++

```
bool SetShunt(
    List<Connector> connectors,
    ShuntMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool SetShunt(
    List<Connector> connectors,
    ShuntMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetShunt(
    connectors As List<Connector>,
    mode As ShuntMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connectors	シャント出力を設定するコネクタ。
mode	指定したコネクタのShuntMode。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXDevice.SetSignalName Method

指定した信号に信号名を割り当てます。

C++

```
bool SetSignalName(
    Signal signal,
    string signalName,
    out List<Problem> problems
);
```

C#

```
public override bool SetSignalName(
    Signal signal,
    string signalName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetSignalName(
    signal As Signal,
    signalName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
signal	名前を変更する信号。
signalName	信号の新しい名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

実装では、信号名が正常に割り当てられた後にHbm.Api.Common.Messaging.MessageBroker.RaiseSignalNameChangedAsyncを呼び出す必要があります。

QuantumXDevice.SetTedsUsageMode Method

TEDS使用モードを設定し、コネクタ設定を再読み取りします。

C++

```
bool SetTedsUsageMode(
    Connector connector,
    TedsUsageMode tedsUsageMode,
    out List<Problem> problems
);
```

C#

```
public override bool SetTedsUsageMode(
    Connector connector,
    TedsUsageMode tedsUsageMode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetTedsUsageMode(
    connector As Connector,
    tedsUsageMode As TedsUsageMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	コネクタ。
tedsUsageMode	設定するTEDS使用モード。
problems	出力: 警告やエラーのリスト。何も発生しなかった場合は空です。

例外

例外	説明
ArgumentException	引数はQuantumXAnalogInConnector - コネクタである必要があります

QuantumXDevice.SetUnit Method

指定したチャネルに単位を割り当てます。

C++

```
bool SetUnit(
    Channel channel,
    string unit,
    out List<Problem> problems
);
```

C#

```
public override bool SetUnit(
    Channel channel,
    string unit,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetUnit(
    channel As Channel,
    unit As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	単位を設定するチャネル。
unit	信号の新しい名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

QuantumXDevice.SupportsExtendedSampleRateMode Method

デバイスが拡張サンプルレートモードをサポートしているかどうかを決定します。

C++

```
bool SupportsExtendedSampleRateMode();
```

C#

```
public bool SupportsExtendedSampleRateMode();
```

Visual Basic

```
Public Function SupportsExtendedSampleRateMode() As bool
```

戻り値

拡張サンプルレートモードがサポートされている場合は`true`、それ以外の場合は`false`。

QuantumXDevice.SupportsPacketRateChange Method

QuantumXはパケットレートの変更をサポートしています。

C++

```
bool SupportsPacketRateChange();
```

C#

```
public override bool SupportsPacketRateChange();
```

Visual Basic

```
Public override Function SupportsPacketRateChange() As bool
```

戻り値

true。

QuantumXDevice.UpdateCanConnectorsSettings Method

デバイスからすべてのCAN コネクタを読み取り、その設定を更新します。

C++

```
void UpdateCanConnectorsSettings();
```

C#

```
public void UpdateCanConnectorsSettings();
```

Visual Basic

```
Public Function UpdateCanConnectorsSettings() As void
```

QuantumXDevice.UpdateConnectorSettings Method

デバイスから提供されたIQuantumXConnector設定を更新します。

C++

```
void UpdateConnectorSettings(
    IQuantumXConnector quantumXConnector
);
```

C#

```
public void UpdateConnectorSettings(
    IQuantumXConnector quantumXConnector
);
```

Visual Basic

```
Public Function UpdateConnectorSettings(
    quantumXConnector As IQuantumXConnector
) As void
```

引数

引数	説明
quantumXConnector	IQuantumXConnector。

QuantumXDevice.WriteTeds Method

C++

```
bool WriteTeds(
    Connector connector,
    out List<Problem> problems,
    Sensor sensor,
    string tid = null,
    string channelName = null
);
```

C#

```
public override bool WriteTeds(
    Connector connector,
    out List<Problem> problems,
    Sensor sensor,
    string tid,
    string channelName
);
```

Visual Basic

```
Public override Function WriteTeds(
    connector As Connector,
    problems As out List<Problem>,
    sensor As Sensor,
    tid As string = null,
    channelName As string = null
) As bool
```

QuantumXDevice Properties

QuantumXDeviceクラスのプロパティの一覧です。

Public Properties

	名前	説明
	AdditionalFeatures	QuantumXデバイスの追加機能。
	Extension	システムの基本モジュールタイプの拡張。
R	FamilyName	このデバイスが属するファミリの名前。
	FpgaVersion	プラットフォームのFPGAバージョン。
	HardwareControllerDate	プラットフォームのハードウェアコントローラの日付。
	HardwareControllerProgramDate	プラットフォームのハードウェアコントローラプログラムの日付。
	HardwareControllerProgramVersion	プラットフォームのハードウェアコントローラプログラム バージョン。
	HardwareControllerVersion	プラットフォームのハードウェアコントローラバージョン。
	HardwareIdentification	プラットフォームのハードウェアIDバージョン。
R	HasAdditionalTedsFieldsSupport	デバイス ファミリが校正とシリアル番号のTEDSフィールドをサポートしているかどうかを示す値を取得します。
R	IsAutomaticTedsMonitoringSupported	このインスタンスが自動TEDSモニタリングをサポートしているかどうかを示す値を取得します(QuantumXの場合は常に`true`)。
R	IsElectricalSnapshotSupported	このインスタンスが電気値の要求をサポートしているかどうかを示す値を取得します。
	IsFirmwareUpdateSupported	このインスタンスがファームウェアの更新をサポートしているかどうかを示す値を取得または設定します。
R	IsReadingFbgSpectrumSupported	このインスタンスがスペクトルの読み取りをサポートしているかどうかを示す値を取得します。
	ModuleId	システムのモジュールID。
	ModuleLabel	システムのモジュール ラベル。
	Revision	システムのリビジョン。
	Selection	システムの基本モジュールタイプの仕様。
	SpecialFirmwareVersion	システムの特別なファームウェアバージョン。
	SpecialVersion	システムの特別バージョン。
	SupportsCanBusTermination	デバイスがCAN busターミネーションをサポートしているかどうかを示す値を取得します。
R	SupportsChangingConnectorType	デバイスがCAN bus間での切り替えをサポートしているかどうかを示す値を取得します。

QuantumXDevice.AdditionalFeatures Property

QuantumXデバイスの追加機能。

C++

```
[JsonIgnore]
__property QuantumXAdditionalFeatures AdditionalFeatures;
```

C#

```
[JsonIgnore]
public QuantumXAdditionalFeatures AdditionalFeatures;
```

Visual Basic

```
<JsonIgnore>
Public Property AdditionalFeatures() As QuantumXAdditionalFeatures
```

備考

追加機能は、Deviceクラスの抽象関数によって事前に定義されていないすべてのパブリック関数をカバーします。

QuantumXDevice.Extension Property

システムの基本モジュールタイプの拡張。

C++

```
__property string Extension;
```

C#

```
public string Extension;
```

Visual Basic

```
Public Property Extension() As string
```

QuantumXDevice.FamilyName Property

このデバイスが属するファミリの名前。

C++

```
__property string FamilyName;
```

C#

```
public override string FamilyName;
```

Visual Basic

```
Public override Property FamilyName() As string
```

備考

これは、クラスHbm.Api.QuantumX.QuantumXDeviceFamilyのプロパティFamilyNameとまったく同じ文字列でなければなりません。

QuantumXDevice.FpgaVersion Property

プラットフォームのFPGAバージョン。

C++

```
__property string FpgaVersion;
```

C#

```
public string FpgaVersion;
```

Visual Basic

```
Public Property FpgaVersion() As string
```

QuantumXDevice.HardwareControllerDate Property

プラットフォームのハードウェアコントローラの日付。

C++

```
__property string HardwareControllerDate;
```

C#

```
public string HardwareControllerDate;
```

Visual Basic

```
Public Property HardwareControllerDate() As string
```

QuantumXDevice.HardwareControllerProgramDate Property

プラットフォームのハードウェアコントローラプログラムの日付。

C++

```
__property string HardwareControllerProgramDate;
```

C#

```
public string HardwareControllerProgramDate;
```

Visual Basic

```
Public Property HardwareControllerProgramDate() As string
```

QuantumXDevice.HardwareControllerProgramVersion Property

プラットフォームのハードウェアコントローラプログラム バージョン。

C++

```
__property string HardwareControllerProgramVersion;
```

C#

```
public string HardwareControllerProgramVersion;
```

Visual Basic

```
Public Property HardwareControllerProgramVersion() As string
```

QuantumXDevice.HardwareControllerVersion Property

プラットフォームのハードウェアコントローラバージョン。

C++

```
__property string HardwareControllerVersion;
```

C#

```
public string HardwareControllerVersion;
```

Visual Basic

```
Public Property HardwareControllerVersion() As string
```

QuantumXDevice.HardwareIdentification Property

プラットフォームのハードウェアIDバージョン。

C++

```
__property string HardwareIdentification;
```

C#

```
public string HardwareIdentification;
```

Visual Basic

```
Public Property HardwareIdentification() As string
```

QuantumXDevice.HasAdditionalTedsFieldsSupport Property

デバイス ファミリが校正とシリアル番号のTEDSフィールドをサポートしているかどうかを示す値を取得します。

C++

```
[JsonIgnore]
__property bool HasAdditionalTedsFieldsSupport;
```

C#

```
[JsonIgnore]
public override bool HasAdditionalTedsFieldsSupport;
```

Visual Basic

```
<JsonIgnore>
Public override Property HasAdditionalTedsFieldsSupport() As bool
```

QuantumXDevice.IsAutomaticTedsMonitoringSupported Property

このインスタンスが自動TEDSモニタリングをサポートしているかどうかを示す値を取得します(QuantumXの場合は常に `true`)。

C++

```
[JsonIgnore]
__property bool IsAutomaticTedsMonitoringSupported;
```

C#

```
[JsonIgnore]
public override bool IsAutomaticTedsMonitoringSupported;
```

Visual Basic

```
<JsonIgnore>
Public override Property IsAutomaticTedsMonitoringSupported() As bool
```

QuantumXDevice.IsElectricalSnapshotSupported Property

このインスタンスが電気値の要求をサポートしているかどうかを示す値を取得します。

C++

```
[JsonIgnore]
[Obsolete("Do not use this property- it is for temporary, internal use only and is subject
to change without notice!")]
[Browsable(false)]
[EditorBrowsable(EditorBrowsableState.Never)]
__property bool IsElectricalSnapshotSupported;
```

C#

```
[JsonIgnore]
[Obsolete("Do not use this property- it is for temporary, internal use only and is subject
to change without notice!")]
[Browsable(false)]
[EditorBrowsable(EditorBrowsableState.Never)]
public override bool IsElectricalSnapshotSupported;
```

Visual Basic

```
<JsonIgnore>
<Obsolete("Do not use this property- it is for temporary, internal use only and is subject
to change without notice!")>
<Browsable(false)>
<EditorBrowsable(EditorBrowsableState.Never)>
Public override Property IsElectricalSnapshotSupported() As bool
```

QuantumXDevice.IsFirmwareUpdateSupported Property

このインスタンスがファームウェアの更新をサポートしているかどうかを示す値を取得または設定します。

C++

```
__property bool IsFirmwareUpdateSupported;
```

C#

```
public override bool IsFirmwareUpdateSupported;
```

Visual Basic

```
Public override Property IsFirmwareUpdateSupported() As bool
```

説明

このインスタンスがファームウェアの更新をサポートしている場合は`true`、そうでない場合は`false`。

QuantumXDevice.IsReadingFbgSpectrumSupported Property

このインスタンスがスペクトルの読み取りをサポートしているかどうかを示す値を取得します。

C++

```
[JsonIgnore]
__property bool IsReadingFbgSpectrumSupported;
```

C#

```
[JsonIgnore]
public override bool IsReadingFbgSpectrumSupported;
```

Visual Basic

```
<JsonIgnore>
Public override Property IsReadingFbgSpectrumSupported() As bool
```

QuantumXDevice.ModuleId Property

システムのモジュールID。

C++

```
__property string ModuleId;
```

C#

```
public string ModuleId;
```

Visual Basic

```
Public Property ModuleId() As string
```

QuantumXDevice.ModuleLabel Property

システムのモジュール ラベル。

C++

```
__property string ModuleLabel;
```

C#

```
public string ModuleLabel;
```

Visual Basic

```
Public Property ModuleLabel() As string
```

QuantumXDevice.Revision Property

システムのリビジョン。

C++

```
__property string Revision;
```

C#

```
public string Revision;
```

Visual Basic

```
Public Property Revision() As string
```

QuantumXDevice.Selection Property

システムの基本モジュールタイプの仕様。

C++

```
__property string Selection;
```

C#

```
public string Selection;
```

Visual Basic

```
Public Property Selection() As string
```

QuantumXDevice.SpecialFirmwareVersion Property

システムの特別なファームウェアバージョン。

C++

```
__property string SpecialFirmwareVersion;
```

C#

```
public string SpecialFirmwareVersion;
```

Visual Basic

```
Public Property SpecialFirmwareVersion() As string
```

QuantumXDevice.SpecialVersion Property

システムの特別バージョン。

C++

```
__property string SpecialVersion;
```

C#

```
public string SpecialVersion;
```

Visual Basic

```
Public Property SpecialVersion() As string
```

QuantumXDevice.SupportsCanBusTermination Property

デバイスがCAN busターミネーションをサポートしているかどうかを示す値を取得します。

C++

```
[JsonIgnore]
[Obsolete("Do not use this property any longer -please use IsCanBusTerminationSupported
instead!")]
[Browsable(false)]
[EditorBrowsable(EditorBrowsableState.Never)]
__property bool SupportsCanBusTermination;
```

C#

```
[JsonIgnore]
[Obsolete("Do not use this property any longer -please use IsCanBusTerminationSupported
instead!")]
[Browsable(false)]
[EditorBrowsable(EditorBrowsableState.Never)]
public bool SupportsCanBusTermination;
```

Visual Basic

```
<JsonIgnore>
<Obsolete("Do not use this property any longer -please use IsCanBusTerminationSupported
instead!")>
<Browsable(false)>
<EditorBrowsable(EditorBrowsableState.Never)>
Public Property SupportsCanBusTermination() As bool
```

説明

[CAN bus終端をサポート]する場合は`true`、それ以外の場合は`false`。

QuantumXDevice.SupportsChangingConnectorType Property

デバイスがCAN bus間での切り替えをサポートしているかどうかを示す値を取得します。

C++

```
[JsonIgnore]
__property bool SupportsChangingConnectorType;
```

C#

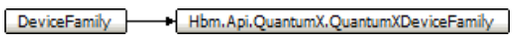
```
[JsonIgnore]
public override bool SupportsChangingConnectorType;
```

Visual Basic

```
<JsonIgnore>
Public override Property SupportsChangingConnectorType() As bool
```

QuantumXDeviceFamily Class

QuantumXデバイスの具体的なデバイスファミリの実装。



C++

```
class QuantumXDeviceFamily : public DeviceFamily;
```

C#

```
public class QuantumXDeviceFamily : DeviceFamily;
```

Visual Basic

```
Public Class QuantumXDeviceFamily
Inherits DeviceFamily
```

ファイル

QuantumXDeviceFamily.cs

QuantumXDeviceFamily Members

次の表は、QuantumXDeviceFamilyによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Find	指定したConnectionInfoに従ってデバイスを検索します。
	GetPossibleDataAcquisitionModes	ファミリの指定したすべてのデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSourceは、ファミリのすべてのデバイスで使用するタイムソースに設定します(指定したデバイスでDataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関連します)。
	PrepareDataAcquisition	指定したDataAcquisitionModeに従って最終的なデバイス設定を行う関数。この機能を使用して、トリガマスタとトリガスレーブなどの設定、指定したデータ収集モードに関連する特定のタイムスタンプ設定の選択などを行います。測定に関与する特定のデバイスファミリのすべてのデバイスもこの関数に渡されるため、特定のデバイスをさまざまな方法で処理できます(たとえば、トリガマスタはスレーブと異なる設定/準備を取得します)。
	QuantumXDeviceFamily	T:Hbm.Api.QuantumX.QuantumXDeviceFamilyクラスの新しいインスタンスを初期化します。
	Scan	ネットワーク上のQuantumXデバイスをスキャンします。

Public Properties

	名前	説明
R	FamilyName	デバイスファミリの名前を取得します(読み取り専用)。
R	IsScanSupported	デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。
	SkillReader	SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

QuantumXDeviceFamily.QuantumXDeviceFamily Constructor

T:Hbm.Api.QuantumX.QuantumXDeviceFamilyクラスの新しいインスタンスを初期化します。

C++

```
QuantumXDeviceFamily();
```

C#

```
public QuantumXDeviceFamily();
```

Visual Basic

```
Public Sub New()
```

QuantumXDeviceFamily Methods

QuantumXDeviceFamilyクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Find	指定したConnectionInfoに従ってデバイスを検索します。
	GetPossibleDataAcquisitionModes	ファミリの指定したすべてのデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSourceは、ファミリのすべてのデバイスで使用するタイムソースに設定します(指定したデバイスでDataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関連します)。
	PrepareDataAcquisition	指定したDataAcquisitionModeに従って最終的なデバイス設定を行う関数。この機能を使用して、トリガマスタとトリガスレーブなどの設定、指定したデータ収集モードに関連する特定のタイムスタンプ設定の選択などを行います。測定に関与する特定のデバイスファミリのすべてのデバイスもこの関数に渡されるため、特定のデバイスをさまざまな方法で処理できます(たとえば、トリガマスタはスレーブと異なる設定/準備を取得します)。
	Scan	ネットワーク上のQuantumXデバイスをスキャンします。

QuantumXDeviceFamily.Find Method

指定したConnectionInfoに従ってデバイスを検索します。

C++

```
List<Device> Find(
    List<ConnectionInfo> connectionInfos,
    out List<Problem> problems
);
```

C#

```
public override List<Device> Find(
    List<ConnectionInfo> connectionInfos,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function Find(
    connectionInfos As List<ConnectionInfo>,
    problems As out List<Problem>
) As List<Device>
```

引数

引数	説明
connectionInfos	接続情報。
problems	検索プロセス中に発生した警告とエラー。

戻り値

見つかったすべてのデバイスのリスト。

QuantumXDeviceFamily.GetPossibleDataAcquisitionModes Method

ファミリの指定したすべてのデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSourceは、ファミリのすべてのデバイスで使用するタイムソースに設定します(指定したデバイスでDataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関連します)。

C++

```
List<DataAcquisitionMode> GetPossibleDataAcquisitionModes(
    List<Device> devices,
    out TimeSource commonTimeSource
);
```

C#

```
public override List<DataAcquisitionMode> GetPossibleDataAcquisitionModes(
    List<Device> devices,
    out TimeSource commonTimeSource
);
```

Visual Basic

```
Public override Function GetPossibleDataAcquisitionModes(
    devices As List<Device>,
    commonTimeSource As out TimeSource
) As List<DataAcquisitionMode>
```

引数

引数	説明
devices	ファミリに属し、測定に参加するデバイス。
commonTimeSource	すべてのデバイスでサポートされているTimeSource(通常はNtpTimeSourceなど)。

戻り値

指定したデバイスの可能性に応じた可能なDataAcquisitionMode。

QuantumXDeviceFamily.PrepareDataAcquisition Method

指定したDataAcquisitionModeに従って最終的なデバイス設定を行う関数。この機能を使用して、トリガマスタとトリガスレーブなどの設定、指定したデータ収集モードに関連する特定のタイムスタンプ設定の選択などを行います。測定に関与する特定のデバイスファミリのすべてのデバイスもこの関数に渡されるため、特定のデバイスをさまざまな方法で処理できます(たとえば、トリガマスタはスレーブと異なる設定/準備を取得します)。

C++

```
void PrepareDataAcquisition(
    DataAcquisitionMode dataAcquisitionMode,
    List<Device> devicesToPrepare
);
```

C#

```
public override void PrepareDataAcquisition(
    DataAcquisitionMode dataAcquisitionMode,
    List<Device> devicesToPrepare
);
```

Visual Basic

```
Public override Function PrepareDataAcquisition(
    dataAcquisitionMode As DataAcquisitionMode,
    devicesToPrepare As List<Device>
) As void
```

引数

引数	説明
dataAcquisitionMode	DataAcquisitionModeを使用して、測定を開始する方法を記述します。
devicesToPrepare	ファミリに属し、測定に参加するデバイス。

戻り値

DataAcquisitionMode.Autoではなく、関数 GetPossibleDataAcquisitionModes(List{Device}, out TimeSource)によって返されるDataAcquisitionModeのいずれかを使用します。

現在位置: Supported devices -> Hbm.Api.QuantumX Namespace ->
Hbm.Api.QuantumX Namespace -> QuantumXDeviceFamily Class ->
QuantumXDeviceFamily.QuantumXDeviceFamily Constructor

QuantumXDeviceFamily.Scan Method

ネットワーク上のQuantumXデバイスをスキャンします。

C++

```
List<Device> Scan();
```

C#

```
public override List<Device>) Scan;
```

Visual Basic

```
Public override Function Scan() As List<Device>
```

戻り値

見つかったQuantumXデバイスのリスト。

QuantumXDeviceFamily Properties

QuantumXAdapterクラスのプロパティの一覧です。

Public Properties

	名前	説明
R	FamilyName	デバイスファミリの名前を取得します(読み取り専用)。
R	IsScanSupported	デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。
	SkillReader	SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

QuantumXDeviceFamily.FamilyName Property

デバイスファミリの名前を取得します(読み取り専用)。

C++

```
__property string FamilyName;
```

C#

```
public override string FamilyName;
```

Visual Basic

```
Public override Property FamilyName() As string
```

QuantumXDeviceFamily.IsScanSupported Property

デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。

C++

```
__property bool IsScanSupported;
```

C#

```
public override bool IsScanSupported;
```

Visual Basic

```
Public override Property IsScanSupported() As bool
```

QuantumXDeviceFamily.SkillReader Property

SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

C++

```
__property ISkillReader SkillReader;
```

C#

```
public override ISkillReader SkillReader;
```

Visual Basic

```
Public override Property SkillReader() As ISkillReader
```

Hbm.Api.QuantumX.Channels Namespace

これは名前空間Hbm.Api.QuantumX.Channelsです。

クラス

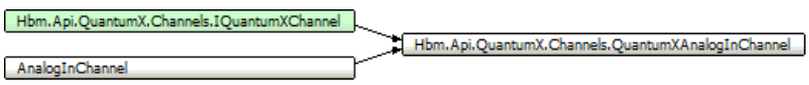
名前	説明
QuantumXAnalogInChannel	センサを接続できる物理的なQuantumXチャネルを表します。
QuantumXAnalogOutChannel	QuantumXのアナログ出力チャネルを表します。
QuantumXCanInChannel	Hbm.Api.SensorDB.Entities.Sensors.CanBusSensorによって構成された、CANコネクタのQuantumX入力チャネルを表します。
QuantumXCanRawChannel	CANコネクタ用のQuantumX CanRawチャネルを表します。
QuantumXDigitalChannel	QuantumXのデジタルチャネルを表します。
QuantumXVirtualChannel	QuantumXの仮想チャネルを表します。

インタフェース

	名前	説明
	IQuantumXChannel	特定のQuantumXチャネルタイプにキャストせずにプロパティにアクセスするための、すべてのQuantumXチャネル用のインタフェースです。

QuantumXAnalogInChannel Class

センサを接続できる物理的なQuantumXチャネルを表します。



C++

```
class QuantumXAnalogInChannel : public AnalogInChannel, public IQuantumXChannel;
```

C#

```
public class QuantumXAnalogInChannel : AnalogInChannel, IQuantumXChannel;
```

Visual Basic

```
Public Class QuantumXAnalogInChannel
Inherits AnalogInChannel
Inherits IQuantumXChannel
```

ファイル

QuantumXAnalogInChannel.cs

QuantumXAnalogInChannel Members

次の表は、QuantumXAnalogInChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	OriginOfName	チャネル名の起源(読み取り専用)。
	Position	チャネルが属する位置(コネクタごとにチャネルが1つしかないため、常に1)。
	TedsUsageMode	TEDS使用モードを取得または設定します。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXAnalogInChannel Methods

QuantumXAnalogInChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

QuantumXAnalogInChannel.GetUniqueID Method

チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXAnalogInChannel Properties

QuantumXAnalogInChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	OriginOfName	チャネル名の起源(読み取り専用)。
	Position	チャネルが属する位置(コネクタごとにチャネルが1つしかないため、常に1)。
	TedsUsageMode	TEDS使用モードを取得または設定します。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXAnalogInChannel.ConnectorPosition Property

チャネルが属するコネクタの位置。

C++

```
__property int ConnectorPosition;
```

C#

```
public int ConnectorPosition;
```

Visual Basic

```
Public Property ConnectorPosition() As Integer
```

QuantumXAnalogInChannel.OriginOfName Property

チャネル名の起源(読み取り専用)。

C++

```
[JsonProperty]
__property OriginOfNameType OriginOfName;
```

C#

```
[JsonProperty]
public OriginOfNameType OriginOfName;
```

Visual Basic

```
<JsonProperty>
Public Property OriginOfName() As OriginOfNameType
```

QuantumXAnalogInChannel.Position Property

チャネルが属する位置(コネクタごとにチャネルが1つしかないため、常に1)。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXAnalogInChannel.TedsUsageMode Property

TEDS使用モードを取得または設定します。

C++

```
__property TedsUsageMode TedsUsageMode;
```

C#

```
public override TedsUsageMode TedsUsageMode;
```

Visual Basic

```
Public override Property TedsUsageMode() As TedsUsageMode
```

備考

TedsUsageMode.NotSupportedおよびTedsUsageMode.UseManuallyはサポートされていません。

QuantumXAnalogInChannel.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

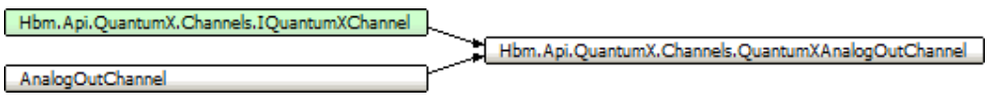
```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

QuantumXAnalogOutChannel Class

QuantumXのアナログ出力チャネルを表します。



C++

```
class QuantumXAnalogOutChannel : public AnalogOutChannel, public IQuantumXChannel;
```

C#

```
public class QuantumXAnalogOutChannel : AnalogOutChannel, IQuantumXChannel;
```

Visual Basic

```
Public Class QuantumXAnalogOutChannel
Inherits AnalogOutChannel
Inherits IQuantumXChannel
```

ファイル

QuantumXAnalogOutChannel .cs

QuantumXAnalogOutChannel Members

次の表は、QuantumXAnalogOutChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	CalibratedBy	チャンネルを調整したユーザ名。
	CalibrationDate	校正の日付。指定していない場合は`null`。
	CalibrationExpirationDate	校正の有効期限。指定していない場合は`null`。
	ConnectorPosition	チャネルが属するコネクタの位置。
	Indentifier	チャネルの識別子。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXAnalogOutChannel Methods

QuantumXAnalogOutChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

QuantumXAnalogOutChannel.GetUniqueID Method

チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXAnalogOutChannel Properties

QuantumXAnalogOutChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	CalibratedBy	チャンネルを調整したユーザ名。
	CalibrationDate	校正の日付。指定していない場合は`null`。
	CalibrationExpirationDate	校正の有効期限。指定していない場合は`null`。
	ConnectorPosition	チャネルが属するコネクタの位置。
	Indentifier	チャネルの識別子。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXAnalogOutChannel.CalibratedBy Property

チャンネルを調整したユーザ名。

C++

```
__property string CalibratedBy;
```

C#

```
public string CalibratedBy;
```

Visual Basic

```
Public Property CalibratedBy() As string
```

QuantumXAnalogOutChannel.CalibrationDate Property

校正の日付。指定していない場合はnull。

C++

```
__property DateTime? CalibrationDate;
```

C#

```
public DateTime? CalibrationDate;
```

Visual Basic

```
Public Property CalibrationDate() As DateTime?
```

QuantumXAnalogOutChannel.CalibrationExpirationDate Property

校正の有効期限。指定していない場合はnull。

C++

```
__property DateTime? CalibrationExpirationDate;
```

C#

```
public DateTime? CalibrationExpirationDate;
```

Visual Basic

```
Public Property CalibrationExpirationDate() As DateTime?
```

QuantumXAnalogOutChannel.ConnectorPosition Property

チャネルが属するコネクタの位置。

C++

```
__property int ConnectorPosition;
```

C#

```
public int ConnectorPosition;
```

Visual Basic

```
Public Property ConnectorPosition() As Integer
```

QuantumXAnalogOutChannel.Indentifier Property

チャネルの識別子。

C++

```
__property string Indentifier;
```

C#

```
public string Indentifier;
```

Visual Basic

```
Public Property Indentifier() As string
```

QuantumXAnalogOutChannel.Position Property

チャネルが属する位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXAnalogOutChannel.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

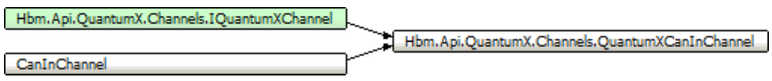
```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

QuantumXCanInChannel Class

Hbm.Api.SensorDB.Entities.Sensors.CanBusSensorによって構成された、CANコネクタのQuantumX入力チャネルを表します。



C++

```
class QuantumXCanInChannel : public CanInChannel, public IQuantumXChannel;
```

C#

```
public class QuantumXCanInChannel : CanInChannel, IQuantumXChannel;
```

Visual Basic

```
Public Class QuantumXCanInChannel
Inherits CanInChannel
Inherits IQuantumXChannel
```

ファイル

QuantumXCanInChannel.cs

QuantumXCanInChannel Members

次の表は、QuantumXCanInChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	OriginOfName	チャネル名の起源(読み取り専用)。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXCanInChannel Methods

QuantumXCanInChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

QuantumXCanInChannel.GetUniqueID Method

チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXCanInChannel Properties

QuantumXCanInChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	OriginOfName	チャネル名の起源(読み取り専用)。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXCanInChannel.ConnectorPosition Property

チャネルが属するコネクタの位置。

C++

```
__property int ConnectorPosition;
```

C#

```
public int ConnectorPosition;
```

Visual Basic

```
Public Property ConnectorPosition() As Integer
```

QuantumXCanInChannel.OriginOfName Property

チャネル名の起源(読み取り専用)。

C++

```
[JsonProperty]
__property OriginOfNameType OriginOfName;
```

C#

```
[JsonProperty]
public OriginOfNameType OriginOfName;
```

Visual Basic

```
<JsonProperty>
Public Property OriginOfName() As OriginOfNameType
```

QuantumXCanInChannel.Position Property

チャネルが属する位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXCanInChannel.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

QuantumXCanRawChannel Class

CANコネクタ用のQuantumX CanRawチャネルを表します。



C++

```
class QuantumXCanRawChannel : public CanRawChannel, public IQuantumXChannel;
```

C#

```
public class QuantumXCanRawChannel : CanRawChannel, IQuantumXChannel;
```

Visual Basic

```
Public Class QuantumXCanRawChannel
Inherits CanRawChannel
Inherits IQuantumXChannel
```

ファイル

QuantumXCanRawChannel.cs

QuantumXCanRawChannel Members

次の表は、QuantumXCanRawChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXCanRawChannel Methods

QuantumXCanRawChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

QuantumXCanRawChannel.GetUniqueID Method

チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXCanRawChannel Properties

QuantumXCanInChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXCanRawChannel.ConnectorPosition Property

チャネルが属するコネクタの位置。

C++

```
__property int ConnectorPosition;
```

C#

```
public int ConnectorPosition;
```

Visual Basic

```
Public Property ConnectorPosition() As Integer
```

QuantumXCanRawChannel.Position Property

チャネルが属する位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXCanRawChannel.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

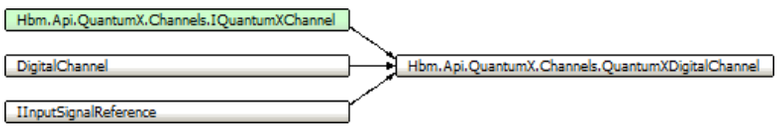
```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

QuantumXDigitalChannel Class

QuantumXのデジタルチャネルを表します。



C++

```
class QuantumXDigitalChannel : public DigitalChannel, public IQuantumXChannel, public
IInputSignalReference;
```

C#

```
public class QuantumXDigitalChannel : DigitalChannel, IQuantumXChannel,
IInputSignalReference;
```

Visual Basic

```
Public Class QuantumXDigitalChannel
Inherits DigitalChannel
Inherits IQuantumXChannel
Inherits IInputSignalReference
```

ファイル

QuantumXDigitalChannel.cs

QuantumXDigitalChannel Members

次の表は、QuantumXDigitalChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタ内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	DefaultValue	信号が接続されていないか利用できない場合に使用するデフォルトのデジタル値。このプロパティは、方向がHbm.Api.Common.Enums.DirectionType.Outの場合にのみ使用されます。
	InputModuleReference	入力信号のソースがあるモジュールへの一意のリファレンスで、QuantumXデバイスのシリアル番号(12桁16進数)で指定します。接続されている入力デバイスのHbm.Api.Common.Entities.Device.SerialNoを使用します。入力信号が存在しない場合、または入力信号ソースが同じデバイス上にある場合は、空である必要があります。このプロパティは、方向がHbm.Api.Common.Enums.DirectionType.Outの場合にのみ使用されます。
	InputSignalReference	出力コネクタの信号ソースへの一意のリファレンス(例:AnalogIn_Connector3.Signal1)。接続された入力デバイスの信号のHbm.Api.Common.Entities.Signals.Signal.GetUniqueID()を使用します。信号が別のモジュールからのものである場合は、モジュールリファレンス(InputModuleReference)も設定します。このプロパティは、方向がHbm.Api.Common.Enums.DirectionType.Outの場合にのみ使用されます。
	IsInverted	すべてのデジタル値が反転しているかどうかを示します。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXDigitalChannel Methods

QuantumXDigitalChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

QuantumXDigitalChannel.GetUniqueID Method

コネクタ内のチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXDigitalChannel Properties

QuantumXDigitalChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	DefaultValue	信号が接続されていないか利用できない場合に使用するデフォルトのデジタル値。このプロパティは、方向がHbm.Api.Common.Enums.DirectionType.Outの場合にのみ使用されます。
	InputModuleReference	入力信号のソースがあるモジュールへの一意のリファレンスで、QuantumXデバイスのシリアル番号(12桁16進数)で指定します。接続されている入力デバイスのHbm.Api.Common.Entities.Device.SerialNoを使用します。入力信号が存在しない場合、または入力信号ソースが同じデバイス上にある場合は、空である必要があります。このプロパティは、方向がHbm.Api.Common.Enums.DirectionType.Outの場合にのみ使用されます。
	InputSignalReference	出力コネクタの信号ソースへの一意のリファレンス(例:AnalogIn_Connector3.Signal1)。接続された入力デバイスの信号のHbm.Api.Common.Entities.Signals.Signal.GetUniqueID()を使用します。信号が別のモジュールからのものである場合は、モジュールリファレンス(InputModuleReference)も設定します。このプロパティは、方向がHbm.Api.Common.Enums.DirectionType.Outの場合にのみ使用されます。
	IsInverted	すべてのデジタル値が反転しているかどうかを示します。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXDigitalChannel.ConnectorPosition Property

チャネルが属するコネクタの位置。

C++

```
__property int ConnectorPosition;
```

C#

```
public int ConnectorPosition;
```

Visual Basic

```
Public Property ConnectorPosition() As Integer
```

QuantumXDigitalChannel.DefaultValue Property

信号が接続されていないか利用できない場合に使用するデフォルトのデジタル値。このプロパティは、方向がHbm.Api.Common.Enums.DirectionType.Outの場合にのみ使用されます。

C++

```
__property DigitalValueType DefaultValue;
```

C#

```
public DigitalValueType DefaultValue;
```

Visual Basic

```
Public Property DefaultValue() As DigitalValueType
```

QuantumXDigitalChannel.InputModuleReference Property

入力信号のソースがあるモジュールへの一意のリファレンスで、QuantumXデバイスのシリアル番号(12桁16進数)で指定します。接続されている入力デバイスのHbm.Api.Common.Entities.Device.SerialNoを使用します。入力信号が存在しない場合、または入力信号ソースが同じデバイス上にある場合は、空である必要があります。このプロパティは、方向がHbm.Api.Common.Enums.DirectionType.Outの場合にのみ使用されます。

C++

```
__property string InputModuleReference;
```

C#

```
public string InputModuleReference;
```

Visual Basic

```
Public Property InputModuleReference() As string
```

備考

接頭文字'UUID='は、デバイスからこの値を読み取るときに内部的に削除され、デバイスに新しい値を割り当てるときに追加されます。

QuantumXDigitalChannel.InputSignalReference Property

出力コネクタの信号ソースへの一意のリファレンス(例:AnalogIn_Connector3.Signal1)。接続された入力デバイスの信号のHbm.Api.Common.Entities.Signals.Signal.GetUniqueID()を使用します。信号が別のモジュールからのものである場合は、モジュールリファレンス(InputModuleReference)も設定します。このプロパティは、方向がHbm.Api.Common.Enums.DirectionType.Outの場合にのみ使用されます。

C++

```
__property string InputSignalReference;
```

C#

```
public string InputSignalReference;
```

Visual Basic

```
Public Property InputSignalReference() As string
```

QuantumXDigitalChannel.IsInverted Property

すべてのデジタル値が反転しているかどうかを示します。

C++

```
__property bool IsInverted;
```

C#

```
public bool IsInverted;
```

Visual Basic

```
Public Property IsInverted() As bool
```

QuantumXDigitalChannel.Position Property

チャネルが属する位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXDigitalChannel.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

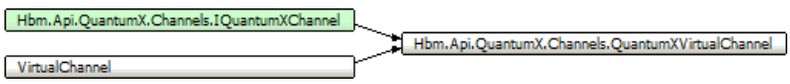
```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

QuantumXVirtualChannel Class

QuantumXの仮想チャネルを表します。



C++

```
class QuantumXVirtualChannel : public VirtualChannel, public IQuantumXChannel;
```

C#

```
internal class QuantumXVirtualChannel : VirtualChannel, IQuantumXChannel;
```

Visual Basic

```
Friend Class QuantumXVirtualChannel
Inherits VirtualChannel
Inherits IQuantumXChannel
```

ファイル

QuantumXVirtualChannel.cs

QuantumXVirtualChannel Members

次の表は、QuantumXVirtualChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXVirtualChannel Methods

QuantumXVirtualChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

QuantumXVirtualChannel.GetUniqueID Method

チャネルが属するコネクタの中でチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXVirtualChannel Properties

QuantumXVirtualChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

QuantumXVirtualChannel.ConnectorPosition Property

チャネルが属するコネクタの位置。

C++

```
__property int ConnectorPosition;
```

C#

```
public int ConnectorPosition;
```

Visual Basic

```
Public Property ConnectorPosition() As Integer
```

QuantumXVirtualChannel.Position Property

チャネルが属する位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```


現在位置: Supported devices -> Hbm.Api.QuantumX Namespace -> Hbm.Api.QuantumX.Channels Namespace -> QuantumXVirtualChannel Class -> QuantumXVirtualChannel Properties

QuantumXVirtualChannel.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。したがって、このタイプでは常に1。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

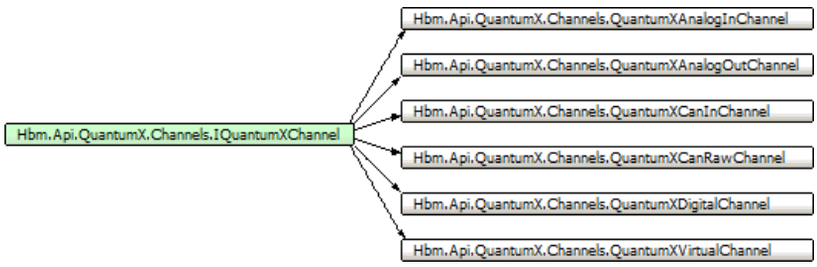
```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

IQuantumXChannel Interface

特定のQuantumXチャネルタイプにキャストせずにプロパティにアクセスするための、すべてのQuantumXチャネル用のインタフェースです。



C++

```
__interface IQuantumXChannel;
```

C#

```
public interface IQuantumXChannel;
```

Visual Basic

```
Public Interface IQuantumXChannel
```

ファイル

IQuantumXChannel.cs

IQuantumXChannel Members

次の表は、IQuantumXChannelによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。現在はHbm.Api.QuantumX.Channels.QuantumXCanInChannelとHbm.Api.QuantumX.Channels.QuantumXCanOutChannelでのみ使用されます。この位置番号は、コネクタのすべてのチャネルで一意ではありません(同じタイプのチャネル要素内でのみ一意です)。このプロパティは内部使用専用です。

IQuantumXChannel Properties

IQuantumXChannelクラスのプロパティの一覧です。

Public Properties

	名前	説明
	ConnectorPosition	チャネルが属するコネクタの位置。
	Position	チャネルが属する位置。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。現在はHbm.Api.QuantumX.Channels.QuantumXCanInChannelとHbm.Api.QuantumX.Channels.QuantumXCanOutChannelでのみ使用されます。この位置番号は、コネクタのすべてのチャネルで一意ではありません(同じタイプのチャネル要素内でのみ一意です)。このプロパティは内部使用専用です。

IQuantumXChannel.ConnectorPosition Property

チャネルが属するコネクタの位置。

C++

```
[JsonProperty]
__property int ConnectorPosition;
```

C#

```
[JsonProperty]
int ConnectorPosition;
```

Visual Basic

```
<JsonProperty>
Property ConnectorPosition() As Integer
```

IQuantumXChannel.Position Property

チャネルが属する位置。

C++

```
[JsonProperty]
__property int Position;
```

C#

```
[JsonProperty]
int Position;
```

Visual Basic

```
<JsonProperty>
Property Position() As Integer
```

IQuantumXChannel.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。現在はHbm.Api.QuantumX.Channels.QuantumXCanInChannelとHbm.Api.QuantumX.Channels.QuantumXCanOutChannelでのみ使用されます。この位置番号は、コネクタのすべてのチャネルで一意ではありません(同じタイプのチャネル要素内でのみ一意です)。このプロパティは内部使用専用です。

C++

```
[JsonProperty]
__property int XmlAddressPosition;
```

C#

```
[JsonProperty]
int XmlAddressPosition;
```

Visual Basic

```
<JsonProperty>
Property XmlAddressPosition() As Integer
```

Hbm.Api.QuantumX.Connectors Namespace

これは名前空間Hbm.Api.QuantumX.Connectorsです。

クラス

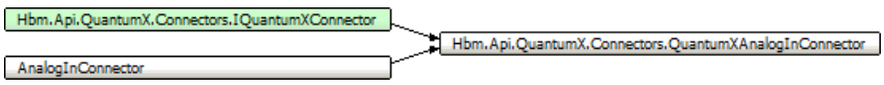
名前	説明
QuantumXAnalogInConnector	アナログ入力コネクタのQuantumX固有の実装。
QuantumXAnalogOutConnector	アナログ出力コネクタのQuantumX固有の実装。
QuantumXCanConnector	CAN busコネクタのQuantumX固有の実装。
QuantumXDigitalConnector	デジタル入出力コネクタのQuantumX固有の実装。
QuantumXOffConnector	QuantumX固有のコネクタ実装で、オフ(無効)に設定されています。

インタフェース

	名前	説明
	IQuantumXConnector	特定のQuantumXコネクタタイプにキャストせずにプロパティにアクセスするための、すべてのQuantumXコネクタ用のインタフェースです。

QuantumXAnalogInConnector Class

アナログ入力コネクタのQuantumX固有の実装。



C++

```
class QuantumXAnalogInConnector : public AnalogInConnector, public IQuantumXConnector;
```

C#

```
public class QuantumXAnalogInConnector : AnalogInConnector, IQuantumXConnector;
```

Visual Basic

```
Public Class QuantumXAnalogInConnector
Inherits AnalogInConnector
Inherits IQuantumXConnector
```

ファイル

QuantumXAnalogInConnector.cs

QuantumXAnalogInConnector Members

次の表は、QuantumXAnalogInConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	Position	デバイスのコネクタの一意の位置番号(1..n)。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

QuantumXAnalogInConnector Methods

QuantumXAnalogInConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

QuantumXAnalogInConnector.GetUniqueID Method

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタ識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXAnalogInConnector Properties

QuantumXAnalogInConnectorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Position	デバイスのコネクタの一意の位置番号(1..n)。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

QuantumXAnalogInConnector.Position Property

デバイスのコネクタの一意の位置番号(1..n)。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```


現在位置: Supported devices -> Hbm.Api.QuantumX Namespace -> Hbm.Api.QuantumX.Connectors Namespace -> QuantumXAnalogInConnector Class -> QuantumXAnalogInConnector Properties

QuantumXAnalogInConnector.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

QuantumXAnalogInConnector.XmlView Property

コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

C++

```
__property XmlViewType XmlView;
```

C#

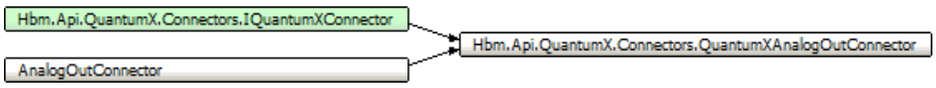
```
public XmlViewType XmlView;
```

Visual Basic

```
Public Property XmlView() As XmlViewType
```

QuantumXAnalogOutConnector Class

アナログ出力コネクタのQuantumX固有の実装。



C++

```
class QuantumXAnalogOutConnector : public AnalogOutConnector, public IQuantumXConnector;
```

C#

```
public class QuantumXAnalogOutConnector : AnalogOutConnector, IQuantumXConnector;
```

Visual Basic

```
Public Class QuantumXAnalogOutConnector
Inherits AnalogOutConnector
Inherits IQuantumXConnector
```

ファイル

QuantumXAnalogOutConnector.cs

QuantumXAnalogOutConnector Members

次の表は、QuantumXAnalogOutConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	OutputType	アナログ出力の種類(読み取り専用)。
	Position	デバイスのコネクタの一意の位置番号(1..n)。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

QuantumXAnalogOutConnector Methods

QuantumXAnalogOutConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

QuantumXAnalogOutConnector.GetUniqueID Method

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタ識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXAnalogOutConnector Properties

QuantumXAnalogOutConnectorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	OutputType	アナログ出力の種類(読み取り専用)。
	Position	デバイスのコネクタの一意の位置番号(1..n)。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

QuantumXAnalogOutConnector.OutputType Property

アナログ出力の種類(読み取り専用)。

C++

```
[JsonProperty]
__property OutputType OutputType;
```

C#

```
[JsonProperty]
public OutputType OutputType;
```

Visual Basic

```
<JsonProperty>
Public Property OutputType() As OutputType
```

QuantumXAnalogOutConnector.Position Property

デバイスのコネクタの一意の位置番号(1..n)。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXAnalogOutConnector.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

QuantumXAnalogOutConnector.XmlView Property

コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

C++

```
__property XmlViewType XmlView;
```

C#

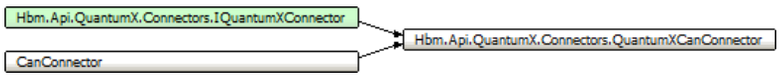
```
public XmlViewType XmlView;
```

Visual Basic

```
Public Property XmlView() As XmlViewType
```

QuantumXCanConnector Class

CAN busコネクタのQuantumX固有の実装。



C++

```
class QuantumXCanConnector : public CanConnector, public IQuantumXConnector;
```

C#

```
public class QuantumXCanConnector : CanConnector, IQuantumXConnector;
```

Visual Basic

```
Public Class QuantumXCanConnector
Inherits CanConnector
Inherits IQuantumXConnector
```

ファイル

QuantumXCanConnector.cs

QuantumXCanConnector Members

次の表は、QuantumXCanConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。
	QuantumXCanConnector	これは、QuantumXCanConnectorクラスのメンバーであるQuantumXCanConnectorです。

Public Properties

	名前	説明
	Ecus	ECU定義のリストを取得または設定します。
	Position	デバイスのコネクタの一意の位置番号(1..n)。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

QuantumXCanConnector.QuantumXCanConnector Constructor

これは、QuantumXCanConnectorクラスのメンバーであるQuantumXCanConnectorです。

C++

```
QuantumXCanConnector();
```

C#

```
public QuantumXCanConnector();
```

Visual Basic

```
Public Sub New()
```

QuantumXCanConnector Methods

QuantumXCanConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

QuantumXCanConnector.GetUniqueID Method

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタ識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXCanConnector Properties

QuantumXCanConnectorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Ecus	ECU定義のリストを取得または設定します。
	Position	デバイスのコネクタの一意の位置番号(1..n)。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

QuantumXCanConnector.Ecus Property

ECU定義のリストを取得または設定します。

C++

```
__property List<CanConnectorEcu> Ecus;
```

C#

```
public List<CanConnectorEcu> Ecus;
```

Visual Basic

```
Public Property Ecus() As List<CanConnectorEcu>
```

説明

ECU定義

QuantumXCanConnector.Position Property

デバイスのコネクタの一意の位置番号(1..n)。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXCanConnector.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

QuantumXCanConnector.XmlView Property

コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

C++

```
__property XmlViewType XmlView;
```

C#

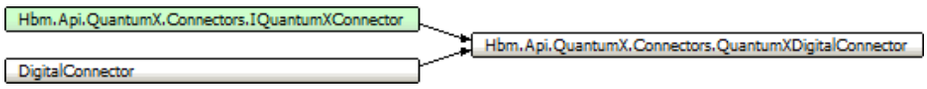
```
public XmlViewType XmlView;
```

Visual Basic

```
Public Property XmlView() As XmlViewType
```

QuantumXDigitalConnector Class

デジタル入出力コネクタのQuantumX固有の実装。



C++

```
class QuantumXDigitalConnector : public DigitalConnector, public IQuantumXConnector;
```

C#

```
public class QuantumXDigitalConnector : DigitalConnector, IQuantumXConnector;
```

Visual Basic

```
Public Class QuantumXDigitalConnector
Inherits DigitalConnector
Inherits IQuantumXConnector
```

ファイル

QuantumXDigitalConnector.cs

QuantumXDigitalConnector Members

次の表は、QuantumXDigitalConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	Pin	デジタルコネクタのピン番号。
	Position	デバイスのコネクタの一意の位置番号(1..n)。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

QuantumXDigitalConnector Methods

QuantumXDigitalConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

QuantumXDigitalConnector.GetUniqueID Method

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタ識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXDigitalConnector Properties

QuantumXDigitalConnectorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Pin	デジタルコネクタのピン番号。
	Position	デバイスのコネクタの一意の位置番号(1..n)。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

QuantumXDigitalConnector.Pin Property

デジタルコネクタのピン番号。

C++

```
__property int Pin;
```

C#

```
public int Pin;
```

Visual Basic

```
Public Property Pin() As Integer
```

QuantumXDigitalConnector.Position Property

デバイスのコネクタの一意の位置番号(1..n)。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXDigitalConnector.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

QuantumXDigitalConnector.XmlView Property

コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

C++

```
__property XmlViewType XmlView;
```

C#

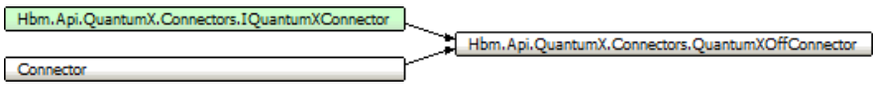
```
public XmlViewType XmlView;
```

Visual Basic

```
Public Property XmlView() As XmlViewType
```

QuantumXOffConnector Class

QuantumX固有のコネクタ実装で、オフ(無効)に設定されています。



C++

```
class QuantumXOffConnector : public Connector, public IQuantumXConnector;
```

C#

```
public class QuantumXOffConnector : Connector, IQuantumXConnector;
```

Visual Basic

```
Public Class QuantumXOffConnector
Inherits Connector
Inherits IQuantumXConnector
```

ファイル

QuantumXOffConnector.cs

QuantumXOffConnector Members

次の表は、QuantumXOffConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	Position	デバイスのコネクタの一意の位置番号(1..n)。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

QuantumXOffConnector Methods

QuantumXOffConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

QuantumXOffConnector.GetUniqueID Method

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタ識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXOffConnector Properties

QuantumXOffConnectorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Position	デバイスのコネクタの一意の位置番号(1..n)。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

QuantumXOffConnector.Position Property

デバイスのコネクタの一意の位置番号(1..n)。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXOffConnector.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。

C++

```
__property int XmlAddressPosition;
```

C#

```
public int XmlAddressPosition;
```

Visual Basic

```
Public Property XmlAddressPosition() As Integer
```

QuantumXOffConnector.XmlView Property

コネクタが配置されているXMLビュー。このプロパティは内部使用専用です。

C++

```
__property XmlViewType XmlView;
```

C#

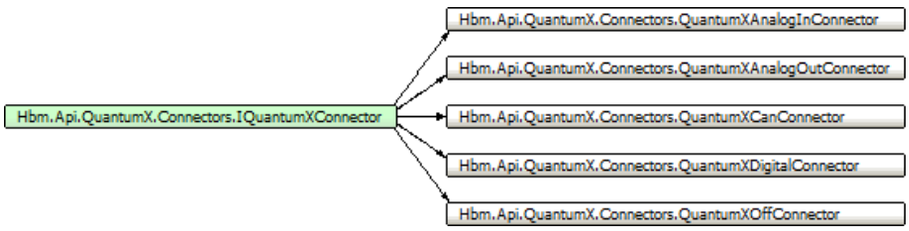
```
public XmlViewType XmlView;
```

Visual Basic

```
Public Property XmlView() As XmlViewType
```

IQuantumXConnector Interface

特定のQuantumXコネクタタイプにキャストせずにプロパティにアクセスするための、すべてのQuantumXコネクタ用のインタフェースです。



C++

```
__interface IQuantumXConnector;
```

C#

```
public interface IQuantumXConnector;
```

Visual Basic

```
Public Interface IQuantumXConnector
```

ファイル

IQuantumXConnector.cs

IQuantumXConnector Members

次の表は、IQuantumXConnectorによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	Position	デバイスのコネクタの一意の位置番号(1..n)。このプロパティは内部使用専用です。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタに含まれているXMLビューの種類。このプロパティは内部使用専用です。

IQuantumXConnector Properties

IQuantumXConnectorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Position	デバイスのコネクタの一意の位置番号(1..n)。このプロパティは内部使用専用です。
	XmlAddressPosition	XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。
	XmlView	コネクタに含まれているXMLビューの種類。このプロパティは内部使用専用です。

IQuantumXConnector.Position Property

デバイスのコネクタの一意の位置番号(1..n)。このプロパティは内部使用専用です。

C++

```
[JsonProperty]
__property int Position;
```

C#

```
[JsonProperty]
int Position;
```

Visual Basic

```
<JsonProperty>
Property Position() As Integer
```


現在位置: Supported devices -> Hbm.Api.QuantumX Namespace -> Hbm.Api.QuantumX.Connectors Namespace -> IQuantumXConnector Interface -> IQuantumXConnector Members

IQuantumXConnector.XmlAddressPosition Property

XMLアドレス属性(例:address="1")によって指定されるチャネルの位置。この位置番号は、コネクタのすべてのチャネルで一意ではありません(指定したXmlViewのコネクタ要素内でのみ一意です)。このプロパティは内部使用専用です。

C++

```
[JsonProperty]
__property int XmlAddressPosition;
```

C#

```
[JsonProperty]
int XmlAddressPosition;
```

Visual Basic

```
<JsonProperty>
Property XmlAddressPosition() As Integer
```

IQuantumXConnector.XmlView Property

コネクタに含まれているXMLビューの種類。このプロパティは内部使用専用です。

C++

```
[JsonProperty]
__property XmlViewType XmlView;
```

C#

```
[JsonProperty]
XmlViewType XmlView;
```

Visual Basic

```
<JsonProperty>
Property XmlView() As XmlViewType
```

Hbm.Api.QuantumX.Enums Namespace

これは名前空間Hbm.Api.QuantumX.Enumsです。

列挙型

	名前	説明
	ConnectorType	QuantumXコネクタの種類。
	DigitalIoInputThreshold	デジタル I/O入力のしきい値。
	FrameworkDllResultCodes	フレームワークDllラッパの結果コード。
	FrameworkError	考えられるQuantumXフレームワークのエラー(ネガティブ)および警告(ポジティブ)コード。
	LedBlinking	モジュールまたはコネクタLEDを点滅させるLED動作モード。
	LedMode	FlashLEDのLEDモード。
	LowpassFilterCharacteristic	サポートしているQuantumXのローパスフィルタのフィルタ特性。
	ModuleStatusId	モジュールのステータスIDを指定します。
	OriginOfNameType	チャネル名の起源の種類。
	OutputType	アナログ出力コネクタの出力の種類。
	QuantumXBridgeType	QuantumXのブリッジの種類。 列挙の値は、xmlのBridgeTypeのQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enBridgeTypeに対応)。 追加の値は、列挙Hbm.Api.SensorDB.Enums.BridgeTypeの値と一致する必要があります。
	QuantumXCarrierFrequencyType	QuantumXのキャリア周波数の種類。 列挙の値は、xml内の励起周波数のQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enExcitationFreqに対応)。 追加の値は、列挙Hbm.Api.SensorDB.Enums.CarrierFrequencyTypeの型関連値の一部と一致する必要があります。Hz単位の直接周波数値は、列挙Hbm.Api.QuantumX.Enums.QuantumXCarrierFrequencyValueの一部です。
	QuantumXCarrierFrequencyValue	QuantumXのキャリア周波数値。 列挙の値は、xml内の励起周波数のQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enExcitationFreqに対応)。 追加の値は、周波数値を直接定義する列挙Hbm.Api.SensorDB.Enums.CarrierFrequencyTypeの周波数値の一部と一致する必要があります。タイプ関連情報は、列挙Hbm.Api.QuantumX.Enums.QuantumXCarrierFrequencyTypeの一部です。
	QuantumXDataRateDomainType	データレートドメインの種類。 列挙の値は、デバイスのシステムビューのxml要素 Hbm.Api.QuantumX.Constants.XmlTag.SYSTEM_DATARATEDOMAIN_REQUESTおよびHbm.Api.QuantumX.Constants.XmlTag.SYSTEM_DATARATEDOMAIN_CURRENTSTATEのQuantumX固有の値と一致します。 追加の値は、列挙Hbm.Api.Common.Enums.DataRateDomainTypeの値と一致する必要があります。
	QuantumXDirectionType	QuantumXのデジタルI/Oの方向変換に使用します。 列挙の値は、Hbm.Api.Common.Enums.DirectionTypeの共通値と一致します。追加の値は、xml内の追加コネクタのビュータグHbm.Api.QuantumX.Constants.XmlTag.ADDCONN_DIGITALIO_MODE_DIGITALINまたはHbm.Api.QuantumX.Constants.XmlTag.ADDCONN_DIGITALIO_MODE_DIGITALOUTの文字列コンテンツと一致する必要があります。
	QuantumXEnableConnectorType	QuantumXの特定のコネクタを有効にするために使用します。

	QuantumXPolarityType	QuantumXの極性の種類。 列挙の値は、xml内の極性のQuantumX 固有の値と一致します(HBM_QX_Framework.hの列挙enPolarityに対応)。 追加の値は、列挙Hbm.Api.SensorDB.Enums.PolarityTypeの値と一致する必要があります。
	QuantumXPtType	QuantumXのPtの種類。 列挙型の値は、xmlのPTの種類のQuantumX 固有の値と一致します(HBM_QX_Framework.hの列挙型enPtTypeに対応)。 追加の値は、PtTypeに従って公称抵抗と一致する必要があります。
	QuantumXSignal2Type	QuantumXのSignal2の種類。 列挙の値は、xmlのsignal2の種類のQuantumX固有の値と一致します(HBM_QX_Framework.h の struct stCounter のフィールド S2enableに対応)。 追加の値は、列挙Hbm.Api.SensorDB.Enums.Signal2Typeの値と一致する必要があります。
	QuantumXSsiCodingType	QuantumXのSSIの種類タイプ。 列挙の値は、xmlのSSIコーディングの種類のQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enSSICodingに対応)。 追加の値は、列挙Hbm.Api.SensorDB.Enums.SsiCodingTypeの値と一致する必要があります。
	QuantumXSyncModeType	QuantumXの同期モードの変換に使用します。 列挙型の値は、Hbm.Api.Common.Enums.SyncModeTypeの共通値と一致します。追加の値は、xml内のシステムのビュータグHbm.Api.QuantumX.Constants.XmlTag.SYSTEM_SYNCSTATUSの文字列コンテンツと一致する必要があります。
	QuantumXTedsUsageMode	QuantumXのTEDS使用モード。 列挙の値は、xmlのTEDS使用モードのQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enTedsUsageに対応)。 追加の値は、列挙Hbm.Api.SensorDB.Enums.TedsUsageModeの値と一致する必要があります。
	QuantumXTestSignalActiveValueType	Statuses.QuantumXChannelTestSignalActiveStatusの値パラメータの値を定義します
	QuantumXThermoCoupleType	QuantumXの熱電対の種類。 列挙型の値は、xml内のThermocoupleTypeのQuantumX固有の値と一致します(HBM_QX_Framework.h内の列挙型enPtType に対応)。 追加の値は、センサデータベースのデータフィールドに従って熱電対の種類と一致する必要があります。
	TedsValidation	`ModuleStatusID.STATUS_TEDS`の値。列挙の値は、xmlのTEDS使用モードのQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enTedsUsageに対応)。追加の値は、列挙Common.Enums.TedsValidationValueTypeの値と一致する必要があります。
	XmlViewType	デバイスからXMLデータを読み書きするためのビューの種類。

Hbm.Api.QuantumX.Enums.ConnectorType Enumeration

QuantumXコネクタの種類。

C++

```
enum ConnectorType {
  AnalogInChannel,
  CanBus
};
```

C#

```
public enum ConnectorType {
  AnalogInChannel,
  CanBus
}
```

Visual Basic

```
Public Enum ConnectorType
  AnalogInChannel
  CanBus
End Enum
```

ファイル

ConnectorType.cs

Hbm.Api.QuantumX.Enums.DigitalIoInputThreshold Enumeration

デジタル I/O入力のしきい値。

C++

```
enum DigitalIoInputThreshold : public int {
  Low = 0,
  High = 2
};
```

C#

```
public enum DigitalIoInputThreshold : int {
  Low = 0,
  High = 2
}
```

Visual Basic

```
Public Enum DigitalIoInputThreshold
Inherits int
  Low = 0
  High = 2
End Enum
```

ファイル

DigitalIoInputThreshold.cs

メンバー

メンバー	説明
Low	しきい値の下限。
High	しきい値の上限。

備考

列挙型の値は、XML内のデジタルI/O入力しきい値のQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙型enDigitalThresholdに対応)。

Hbm.Api.QuantumX.Enums.FrameworkDllResultCodes Enumeration

フレームワークDllラッパの結果コード。

C++

```
enum FrameworkDllResultCodes {
  WarnModuleNotReadyForFieldbusAcq = 1515710,
  ErrEcatFunctionblockSetPdoStatus = -1515708,
  ErrEcatFunctionblockSetPdoIndex = -1515707,
  ErrEcatFunctionblockCreateProblem = -1515706,
  ErrEcatRtsignaleNotValid = -1515705,
  WarnEcatDemuxSignaleNotValid = 1515704,
  WarnEcatModuleParamNotReady = 1515703,
  ErrEcatModuleStatusNotReadable = -1515702,
  WarnEcatCorrectedFwSignaleListSize = 1515701,
  ErrEcatCountOfFwSignalsLimitReached = -1515700,
  ErrNtpServerNotReached = -1515600,
  ErrFwNoNecChannel = -1018005,
  ErrFwNoNecDownload = -1018006,
  ErrFwNoMspChannel = -1018000,
  WarnExcitationcalibrationDataMissing = 1017998,
  WarnCalcalibrationDataMissing = 1017997,
  ErrMspDataCorrupt = -1017996,
  ErrInitjustFromMsp = -1017995,
  ErrNoValidInitjust = -1017993,
  ErrStoringInitjust = -1017992,
  ErrGettingInitjust = -1017991,
  ErrPllvolOutofrange = -1017990,
  ErrPllvolCorrupt = -1017989,
  ErrPllvolFromMsp = -1017988,
  ErrDataAddrBustest = -1017987,
  ErrDataRateDomainNotSupported = -1016100,
  ErrSysFilename = -1016050,
  ErrSysFileTooLarge = -1016049,
  ErrSysBadDelivery = -1016045,
  ErrSysOpenlogfile = -1016040,
  ErrSysFrameworkStart = -1016035,
  ErrSysOpenCloseWd = -1016030,
  ErrSysInvalidTime = -1016025,
  ErrSysTooManyEvents = -1016003,
  ErrSysCreatingObject = -1016002,
  ErrSpiOpenDriver = -1015050,
  ErrSpiWakeupDac = -1015049,
  ErrSpiWriteDriverDigits = -1015048,
  ErrSpiWriteDriverCommand = -1015047,
  ErrSpiVoltOutofRange = -1015040,
  ErrSpiPlldac = -1015039,
  ErrSyncOpenDriver = -1015033,
  ErrSyncAnyTsdProblem = -1015032,
  ErrSyncNotAllModulesReady = -1015031,
  ErrSyncTimesourceDataNotValid = -1015030,
  ErrSyncActivateEthercat = -1015029,
  ErrSyncActivateIrig = -1015028,
  ErrSyncActivatePtp = -1015027,
  ErrSyncActivateNtp = -1015026,
  ErrSyncActivateAuto = -1015025,
  ErrParamIpSubnetmaskSyntax = -1015024,
  ErrParamIpAddressSyntax = -1015023,
  ErrParamSyncTimesourceNotSupported = -1015022,
```

```
ErrSyncCyclediffTooLarge = -1015021,
ErrSyncFreeChannel = -1015019,
ErrSyncRecvCtxRunning = -1015017,
ErrSyncAllocTx = -1015014,
ErrSyncFreeTx = -1015013,
ErrSyncAllocRx = -1015012,
ErrSyncFreeRx = -1015011,
ErrSyncDriver = -1015010,
ErrSyncNotInSync = -1015009,
ErrSyncInitVoltage = -1015008,
ErrSyncAdjustPll = -1015007,
ErrSyncSyncbufEmpty = -1015006,
ErrSyncCtx = -1015005,
ErrAdjSetInitCalFacts = -1014999,
ErrAdjSetActJustFacts = -1014998,
ErrAdjGetActJustFacts = -1014997,
ErrAdjGetInitCalFacts = -1014996,
ErrAdjGetInitMeasFacts = -1014995,
ErrAdjMeasrangeInvalid = -1014994,
ErrAdjMeasrangeOverrun = -1014993,
ErrAdjWrongConnector = -1014992,
ErrThermoWrongConnector = -1014982,
ErrConnWrongConnector = -1014801,
ErrConnInstanceNull = -1014800,
ErrAnalogoutNoconn = -1014603,
ErrAnalogoutSigovfl = -1014602,
ErrAnalogoutOccf = -1014601,
ErrAnalogoutOvfl = -1014600,
ErrSignalreferenceDecoderindexNotSupported = -1014515,
ErrSignalreferenceDecoderindexLocalNotSupported = -1014514,
ErrSignalreferenceSignalgeneratorindexNotSupported = -1014513,
ErrSignalreferenceSignalgeneratorindexLocalNotSupported = -1014512,
ErrSignalreferenceDigitalioindexLocalNotSupported = -1014511,
ErrSignalreferencePeakindexLocalNotSupported = -1014510,
ErrSignalreferenceMathindexLocalNotSupported = -1014509,
ErrSignalreferenceAlarmindexLocalNotSupported = -1014508,
ErrSignalreferenceConnectorLocalNotSupported = -1014507,
ErrSignalreferenceSignalindexLocalNotSupported = -1014506,
ErrSignalreferenceCannotCreate = -1014505,
ErrSignalreferenceUnknown = -1014504,
WarnSignalreferenceUnknown = 1014504,
ErrParamLevelsIdentical = -1014503,
ErrSignalreferenceConnectorNotSupported = -1014502,
WarnMaxIsoSignalindexReached = 1014501,
ErrSignalreferenceSignalindexNotSupported = -1014500,
ErrIndexNotSupported = -1014499,
ErrSignalreferenceSignaltypeNotSupportedOrSyntaxError = -1014498,
WarnSignalreferenceEmpty = 1014497,
ErrModulereferenceSyntax = -1014496,
ErrSignalreferencePeakindexNotSupported = -1014495,
ErrSignalreferenceMathindexNotSupported = -1014494,
ErrSignalreferenceAlarmindexNotSupported = -1014493,
ErrModulereferenceNotLocal = -1014492,
ErrModulereferenceNotExternal = -1014491,
ErrSignalreferenceSyntax = -1014490,
ErrPhysunitNotAvailable = -1014489,
ErrSignalreferenceAsyncsignalNotAllowedOrSyntaxError = -1014488,
ErrSignalreferenceDigitalioindexNotSupported = -1014487,
ErrIoctlDevice = -1014414,
ErrDeviceNotOpen = -1014413,
ErrCloseDevice = -1014412,
```

```
    ErrDeviceAlreadyOpen = -1014411,
    ErrOpenDevice = -1014410,
    ErrPinfoNull = -1014406,
    ErrPointerNull = -1014405,
    ErrParamsettingsNull = -1014404,
    ErrParamCalcsyncsignal = -1014403,
    ErrPplatformNull = -1014402,
    ErrPampNull = -1014401,
    ErrPanalogNull = -1014400,
    ErrTedsData = -1014374,
    ErrNoSignalGeneratorUnitAvailable = -1014348,
    ErrNoGateUnitAvailable = -1014347,
    ErrNoPeakUnitAvailable = -1014346,
    ErrNoMathUnitAvailable = -1014345,
    ErrNoAlarmUnitAvailable = -1014344,
    ErrNoDigitalIoAvailable = -1014343,
    ErrNoCanConnectorsAvailable = -1014342,
    ErrNoAnalogOutConnectorsAvalibale = -1014341,
    ErrNoAnalogInConnectorsAvailable = -1014340,
    ErrHandlingExternalAdaptor = -1014334,
    ErrModeNotSupportedBySensortype = -1014333,
    WarnModeNotSupportedBySensortype = 1014333,
    ErrNotAvailableIfExternalAdaptorPresent = -1014332,
    WarnNotAvailableIfExternalAdaptorPresent = 1014332,
    ErrExecuteControl = -1014330,
    WarnExecuteControl = 1014330,
    ErrValueNotSupported = -1014324,
    WarnValueNotSupported = 1014324,
    ErrAvailableOnlyWithExternalAdaptor = -1014323,
    WarnAvailableOnlyWithExternalAdaptor = 1014323,
    ErrControlNotSupportedBySensortype = -1014322,
    ErrControlNotSupportedAtConnector = -1014321,
    ErrControlNotSupported = -1014320,
    WarnControlNotSupported = 1014320,
    ErrModeNotSupported = -1014319,
    WarnModeNotSupported = 1014319,
    ErrKeyNotFound = -1014318,
    ErrNoSpaceLeft = -1014317,
    ErrAssemblySizeNotSupported = -1014316,
    WarnCtrlPeakmodeSignalctrl = 1014310,
    WarnCtrlRunmodeSignalctrl = 1014309,
    ErrCtrlUnableToSetSignalgenerator = -1014308,
    ErrCtrlUnableToControlAlarmUnit = -1014307,
    ErrCtrlUnableToControlPeakUnit = -1014306,
    ErrCtrlUnableToSetZero = -1014305,
    ErrCtrlUnableToControlMathUnit = -1014304,
    ErrCtrlUnableToControlSignalGenerator = -1014303,
    ErrCtrlUnspecified = -1014302,
    ErrWrongSecurityCode = -1014301,
    ErrCtrlWrongPar = -1014300,
    ErrCalpointTypeProtected = -1014299,
    ErrCanRawReceiverEnableDisable = -1014285,
    ErrCanResourceOverload = -1014284,
    ErrTransmissiontypeSourcechangeNoSupport64Bit = -1014282,
    ErrCanWrongIndex = -1014281,
    ErrCanValueOnErrorTypeNotSupported = -1014280,
    ErrCanValueOnErrorTypeNotSupportedByDataformat = -1014279,
    ErrCanSetBitrate = -1014274,
    WarnCanBitrateChanged = 1014273,
    WarnCanBitrateSamplepointratioChanged = 1014272,
    ErrCanAnyDecoderTimeout = -1014271,
```

```
WarnCanAnyDecoderTimeout = 1014271,
ErrCanAnyDecoderLossOfSignal = -1014270,
WarnCanAnyDecoderLossOfSignal = 1014270,
ErrCanMappingMassageInactive = -1014269,
ErrCanMappingSourcesOverlap = -1014268,
WarnCanMappingNoSourceDefined = 1014266,
ErrOnlyOneMappingsourcePossible = -1014265,
WarnTimerSetToMinimumPossible = 1014264,
WarnSignallengthChanged = 1014261,
ErrCanCcp = -1014259,
ErrCanWriteCansignal = -1014258,
ErrCanTransmitSignalreferenceInvalid = -1014257,
ErrCanTransmitModulereferenceNotLocal = -1014256,
ErrCanTransmitMsg = -1014255,
WarnCanTransmissiontypeChanged = 1014254,
ErrCanTransmissiontypeNotSupported = -1014253,
ErrCanConnectorNotSupported = -1014252,
ErrCanTransmitChannel = -1014250,
ErrCanTransmitUserdefMsg = -1014245,
ErrCanTransmitByteorderNotSupported = -1014244,
ErrCanTransmitTable = -1014243,
ErrCanTransmitBytevalue = -1014242,
ErrCanTransmitBytecount = -1014241,
ErrCanTransmitterUserdefMsgSetTable = -1014240,
ErrCanDecoderModelength64 = -1014238,
ErrCanDecoderModestartbit63 = -1014237,
ErrCanDecoderLength64 = -1014236,
ErrCanDecoderStartbit63 = -1014235,
ErrCanCombinationIdentifierFrameformat = -1014234,
ErrCanChannelno = -1014233,
ErrCanDecoderModeStartbitLength = -1014232,
ErrCanCombinationStartbitLength = -1014231,
ErrCanDecoderFormatDouble = -1014230,
ErrCanDecoderFormatFloat = -1014229,
ErrCanDecoderFormat32 = -1014228,
ErrCanDecoderRbufnoInvalid = -1014227,
ErrCanDecoderRbufnoUsedTwice = -1014226,
ErrCanDecoderGetTable = -1014225,
ErrCanDecoderSetTable = -1014224,
ErrCanDecoderTableFull = -1014223,
ErrCanDecoderDisable = -1014222,
ErrCanDecoderEnable = -1014221,
ErrCanDecoderTableNotValid = -1014220,
ErrCanDecoderParamCanSignal = -1014219,
ErrCanNoResponse = -1014218,
WarnCanResetRequest = 1014217,
ErrCanTransmitterOverrun = -1014216,
ErrCanBusErrorTransmitter = -1014215,
ErrCanBusWarningTransmitter = -1014214,
WarnCanBusWarningTransmitter = 1014214,
ErrCanReceiverOverrun = -1014213,
ErrCanBusOff = -1014212,
ErrCanBusErrorReceiver = -1014211,
ErrCanBusWarningReceiver = -1014210,
WarnCanBusWarningReceiver = 1014210,
ErrCanInitCansignalParams = -1014209,
ErrCanGetBitrate = -1014208,
ErrCanGetState = -1014207,
ErrCanInvalidHandle = -1014206,
ErrCanIndex = -1014205,
ErrCanIoctl = -1014204,
```

```
    ErrCanBitrateNotSupported = -1014203,
    ErrCanDisable = -1014202,
    ErrCanEnable = -1014201,
    ErrCanOpendrv = -1014200,
    ErrFbfsAccessLed = -1014191,
    ErrFbfsAccessCan = -1014190,
    ErrDaqSetCarrierfreqMode = -1014143,
    ErrDaqSetFreezedFlag = -1014142,
    ErrDaqSetRtmeasmode = -1014141,
    ErrDaqTrigtime = -1014140,
    ErrDaqTriggerunknown = -1014139,
    ErrDaqIsosignallistVersion = -1014138,
    ErrDaqAmpNull = -1014137,
    ErrDaqRtdriverNull = -1014133,
    ErrDaqWrongChanType = -1014131,
    WarnScalingAnalogoutGainIfAnaloginLintable = 1014128,
    ErrDaqGetDemodInfo = -1014127,
    ErrDaqSetErrorbuf = -1014125,
    ErrDaqDrvSetPtCoeffs = -1014124,
    ErrDaqDrvSetColdjunction = -1014123,
    ErrDaqDrvSetInitjust = -1014122,
    ErrScalingPolynomNotSupportedIfAnalogoutEnabled = -1014121,
    ErrScalingPolynomTooMuchCoeffs = -1014120,
    ErrScalingPolynomNotEnoughCoeffs = -1014119,
    ErrScalingPolynomTooMuchSegments = -1014118,
    ErrScalingLintableTooMuchPoints = -1014117,
    ErrScalingPolynomNotEnoughSegments = -1014116,
    ErrAccessRtdriver = -1014115,
    ErrScalingLintableNotEnoughPoints = -1014114,
    ErrDaqScalingInternal = -1014113,
    ErrDaqScaletype = -1014112,
    ErrWrongChannelIndex = -1014111,
    ErrWrongSigIndex = -1014110,
    ErrWrongConnector = -1014109,
    ErrDaqInitjustNotfound = -1014108,
    ErrDaqInitjustPointnr = -1014107,
    ErrDaqInitjustWrongpar = -1014106,
    ErrDaqSigidLen = -1014105,
    ErrDaqSigidUnknown = -1014102,
    ErrDaqFilter = -1014100,
    ErrDaqSignalNotAvailable = -1014099,
    ErrDaqAddFbtype = -1014098,
    ErrDaqAddFbInstance = -1014097,
    ErrDaqSetFbInput = -1014096,
    ErrDaqOutrate = -1014095,
    ErrDaqSetBufferPars = -1014094,
    ErrDaqSetZeroval = -1014093,
    ErrDaqFifoParams = -1014092,
    ErrDaqFiforeaderCalParams = -1014091,
    ErrDaqScalingParams = -1014090,
    ErrDaqSetIsoModuleId = -1014089,
    ErrDaqSetIsoTransmit = -1014088,
    ErrDaqGetIsoTransmit = -1014087,
    ErrDaqOverrun = -1014085,
    ErrDaqUndefined = -1014082,
    ErrDaqFifoSync = -1014070,
    ErrDaqOverload = -1014060,
    ErrDaqProcess = -1014061,
    ErrCriticalHighVoltage = -1014056,
    ErrDaqOccf = -1014055,
    ErrDaqLimse = -1014050,
```

```
ErrDaqNosignalsubscribed = -1014030,
ErrDaqBuffersize = -1014025,
ErrDaqAmpoff = -1014023,
ErrDaqOpenrtdrv = -1014010,
ErrDaqSubscrRunning = -1014006,
ErrDaqSubscrUuid = -1014005,
ErrDaqSubscrState = -1014004,
ErrDaqSubscrition = -1014002,
ErrDataFormatNotSupported = -1010000,
ErrSrvStatComposer = -1004305,
ErrMthdSysNorestartfkt = -1004260,
ErrMthdSysWrongchantype = -1004242,
ErrMthdSysWrongchan = -1004240,
ErrMthdSysInvalidfwfile = -1004208,
ErrMthdSysFilenotfound = -1004201,
ErrOpenfile = -1004200,
ErrWriteFile = -1004199,
ErrReadFile = -1004198,
ErrParseXmlSettingRefused = -1004160,
ErrParWrongType = -1004153,
WarnParNodefset = 1004148,
ErrParViewidInuse = -1004146,
ErrParReadingVal = -1004144,
WarnParEmptyVal = 1004143,
ErrParViewinvalid = -1004142,
ErrParWrongxpathstate = -1004140,
ErrParXpathinvalid = -1004138,
ErrParNoroot = -1004136,
WarnParParsing = 1004134,
ErrParParsing = -1004135,
ErrParInvalidaddr = -1004118,
WarnParTagunkown = 1004112,
ErrParTagunkown = -1004111,
ErrParOutofrange = -1004110,
WarnParOutofrange = 1004110,
ErrParInvalidattr = -1004109,
WarnParReadonly = 1004108,
ErrParChoiceinvalid = -1004107,
ErrParCntxtlost = -1004106,
ErrParChoicevaluesyntaxinvalid = -1004105,
ErrParRestorefailed = -1004104,
ErrParXmlinvalid = -1004103,
ErrTxtelcontainerTooSmall = -1003001,
ErrParseProcessingData = -1002002,
ErrParseInvalidData = -1002001,
ErrUtilFileRestore = -1001140,
ErrUtilInvalidFilehandle = -1001136,
ErrUtilStrconvert = -1001132,
ErrUtilNoPatternMatch = -1001131,
ErrEvflgTimeout = -1001130,
ErrEvflgPthreads = -1001128,
ErrUtilTimerInvalid = -1001125,
WarnUtilTimerWasrunning = 1001025,
ErrUtilLanguage = -1001110,
ErrComposerTagNotDefined = -1001002,
ErrNetNetworkadapter = -1000680,
ErrUdpBindtodevice = -1000658,
ErrUdpBind = -1000656,
ErrUdpSetbroadcast = -1000654,
ErrUdpSetreuseadr = -1000652,
ErrUdpOpensock = -1000650,
```

```
  ErrTcpBind = -1000644,
  ErrTcpListen = -1000642,
  ErrTcpCheckip = -1000638,
  ErrTcpGetadr = -1000636,
  ErrTcpGetport = -1000635,
  ErrTcpSetnodelay = -1000632,
  ErrTcpSetnonblock = -1000630,
  ErrTcpOpensock = -1000628,
  ErrTcpConnect = -1000626,
  ErrTcpAccept = -1000624,
  ErrTcpConnectFailed = -1000620,
  ErrTcpIpadrAlreadyConnected = -1000615,
  ErrTcpAdrUnknown = -1000612,
  ErrTcpSocketEintr = -1000604,
  ErrTcpSocketHup = -1000603,
  ErrTcpSocketDescriptor = -1000602,
  ErrConRec = -1000524,
  ErrConSending = -1000522,
  ErrConInvalidTeltype = -1000510,
  ErrConAdrresiniFailed = -1000504,
  ErrConConhndlrUuidunknown = -1000500,
  ErrTransReadinghdr = -1000430,
  ErrTransNotfound = -1000420,
  ErrTransCanceled = -1000419,
  ErrTransSendReqTimeout = -1000416,
  ErrTransAckTimeout = -1000417,
  ErrTransReqTimeout = -1000418,
  ErrTransInvalidTelformat = -1000414,
  ErrTransRequestorWrongprottype = -1000404,
  ErrTransResponderWrongprottype = -1000405,
  ErrTransNotForUs = -1000406,
  ErrNoMspChksum = -1000301,
  ErrSrvControl = -1000300,
  ErrSrvNodata = -1000275,
  ErrSrvNoresponseYet = -1000271,
  ErrSrvReqrmthdNoreqtel = -1000265,
  ErrSrvParamerror = -1000255,
  ErrSrvMthdoutUnknownmthd = -1000203,
  ErrSrvMthdoutWrongansw = -1000202,
  ErrSrvMthdoutToomuch = -1000201,
  ErrSrvUnknownsrv = -1000107,
  ErrSrvUnknownmthd = -1000106,
  ErrIndexOutOfRange = -1000000
};
```

C#

```
public enum FrameworkDllResultCodes {
  WarnModuleNotReadyForFieldbusAcq = 1515710,
  ErrEcatFunctionblockSetPdoStatus = -1515708,
  ErrEcatFunctionblockSetPdoIndex = -1515707,
  ErrEcatFunctionblockCreateProblem = -1515706,
  ErrEcatRtsignaleNotValid = -1515705,
  WarnEcatDemuxSignaleNotValid = 1515704,
  WarnEcatModuleParamNotReady = 1515703,
  ErrEcatModuleStatusNotReadable = -1515702,
  WarnEcatCorrectedFwSignaleListSize = 1515701,
  ErrEcatCountOfFwSignalsLimitReached = -1515700,
  ErrNtpServerNotReached = -1515600,
  ErrFwNoNecChannel = -1018005,
  ErrFwNoNecDownload = -1018006,
  ErrFwNoMspChannel = -1018000,
  WarnExcitationcalibrationDataMissing = 1017998,
  WarnCalcalibrationDataMissing = 1017997,
  ErrMspDataCorrupt = -1017996,
  ErrInitjustFromMsp = -1017995,
  ErrNoValidInitjust = -1017993,
  ErrStoringInitjust = -1017992,
  ErrGettingInitjust = -1017991,
  ErrPllvolOutofrange = -1017990,
  ErrPllvolCorrupt = -1017989,
  ErrPllvolFromMsp = -1017988,
  ErrDataAddrBustest = -1017987,
  ErrDataRateDomainNotSupported = -1016100,
  ErrSysFilename = -1016050,
  ErrSysFileTooLarge = -1016049,
  ErrSysBadDelivery = -1016045,
  ErrSysOpenlogfile = -1016040,
  ErrSysFrameworkStart = -1016035,
  ErrSysOpenCloseWd = -1016030,
  ErrSysInvalidTime = -1016025,
  ErrSysTooManyEvents = -1016003,
  ErrSysCreatingObject = -1016002,
  ErrSpiOpenDriver = -1015050,
  ErrSpiWakeupDac = -1015049,
  ErrSpiWriteDriverDigits = -1015048,
  ErrSpiWriteDriverCommand = -1015047,
  ErrSpiVoltOutofRange = -1015040,
  ErrSpiPlldac = -1015039,
  ErrSyncOpenDriver = -1015033,
  ErrSyncAnyTsdProblem = -1015032,
  ErrSyncNotAllModulesReady = -1015031,
  ErrSyncTimesourceDataNotValid = -1015030,
  ErrSyncActivateEthercat = -1015029,
  ErrSyncActivateIrig = -1015028,
  ErrSyncActivatePtp = -1015027,
  ErrSyncActivateNtp = -1015026,
  ErrSyncActivateAuto = -1015025,
  ErrParamIpSubnetmaskSyntax = -1015024,
  ErrParamIpAddressSyntax = -1015023,
  ErrParamSyncTimesourceNotSupported = -1015022,
  ErrSyncCyclediffTooLarge = -1015021,
  ErrSyncFreeChannel = -1015019,
  ErrSyncRecvCtxRunning = -1015017,
  ErrSyncAllocTx = -1015014,
  ErrSyncFreeTx = -1015013,
  ErrSyncAllocRx = -1015012,
```

```
    ErrSyncFreeRx = -1015011,
    ErrSyncDriver = -1015010,
    ErrSyncNotInSync = -1015009,
    ErrSyncInitVoltage = -1015008,
    ErrSyncAdjustPll = -1015007,
    ErrSyncSyncbufEmpty = -1015006,
    ErrSyncCtx = -1015005,
    ErrAdjSetInitCalFacts = -1014999,
    ErrAdjSetActJustFacts = -1014998,
    ErrAdjGetActJustFacts = -1014997,
    ErrAdjGetInitCalFacts = -1014996,
    ErrAdjGetInitMeasFacts = -1014995,
    ErrAdjMeasrangeInvalid = -1014994,
    ErrAdjMeasrangeOverrun = -1014993,
    ErrAdjWrongConnector = -1014992,
    ErrThermoWrongConnector = -1014982,
    ErrConnWrongConnector = -1014801,
    ErrConnInstanceNull = -1014800,
    ErrAnalogoutNoconn = -1014603,
    ErrAnalogoutSigovfl = -1014602,
    ErrAnalogoutOccf = -1014601,
    ErrAnalogoutOvfl = -1014600,
    ErrSignalreferenceDecoderindexNotSupported = -1014515,
    ErrSignalreferenceDecoderindexLocalNotSupported = -1014514,
    ErrSignalreferenceSignalgeneratorindexNotSupported = -1014513,
    ErrSignalreferenceSignalgeneratorindexLocalNotSupported = -1014512,
    ErrSignalreferenceDigitalioindexLocalNotSupported = -1014511,
    ErrSignalreferencePeakindexLocalNotSupported = -1014510,
    ErrSignalreferenceMathindexLocalNotSupported = -1014509,
    ErrSignalreferenceAlarmindexLocalNotSupported = -1014508,
    ErrSignalreferenceConnectorLocalNotSupported = -1014507,
    ErrSignalreferenceSignalindexLocalNotSupported = -1014506,
    ErrSignalreferenceCannotCreate = -1014505,
    ErrSignalreferenceUnknown = -1014504,
    WarnSignalreferenceUnknown = 1014504,
    ErrParamLevelsIdentical = -1014503,
    ErrSignalreferenceConnectorNotSupported = -1014502,
    WarnMaxIsoSignalindexReached = 1014501,
    ErrSignalreferenceSignalindexNotSupported = -1014500,
    ErrIndexNotSupported = -1014499,
    ErrSignalreferenceSignaltypeNotSupportedOrSyntaxError = -1014498,
    WarnSignalreferenceEmpty = 1014497,
    ErrModulereferenceSyntax = -1014496,
    ErrSignalreferencePeakindexNotSupported = -1014495,
    ErrSignalreferenceMathindexNotSupported = -1014494,
    ErrSignalreferenceAlarmindexNotSupported = -1014493,
    ErrModulereferenceNotLocal = -1014492,
    ErrModulereferenceNotExternal = -1014491,
    ErrSignalreferenceSyntax = -1014490,
    ErrPhysunitNotAvailable = -1014489,
    ErrSignalreferenceAsyncsignalNotAllowedOrSyntaxError = -1014488,
    ErrSignalreferenceDigitalioindexNotSupported = -1014487,
    ErrIoctlDevice = -1014414,
    ErrDeviceNotOpen = -1014413,
    ErrCloseDevice = -1014412,
    ErrDeviceAlreadyOpen = -1014411,
    ErrOpenDevice = -1014410,
    ErrPinfoNull = -1014406,
    ErrPointerNull = -1014405,
    ErrParamsettingsNull = -1014404,
    ErrParamCalcsyncsignal = -1014403,
```

```
ErrPplatformNull = -1014402,
ErrPampNull = -1014401,
ErrPanalogNull = -1014400,
ErrTedsData = -1014374,
ErrNoSignalGeneratorUnitAvailable = -1014348,
ErrNoGateUnitAvailable = -1014347,
ErrNoPeakUnitAvailable = -1014346,
ErrNoMathUnitAvailable = -1014345,
ErrNoAlarmUnitAvailable = -1014344,
ErrNoDigitalIoAvailable = -1014343,
ErrNoCanConnectorsAvailable = -1014342,
ErrNoAnalogOutConnectorsAvalibale = -1014341,
ErrNoAnalogInConnectorsAvailable = -1014340,
ErrHandlingExternalAdaptor = -1014334,
ErrModeNotSupportedBySensortype = -1014333,
WarnModeNotSupportedBySensortype = 1014333,
ErrNotAvailableIfExternalAdaptorPresent = -1014332,
WarnNotAvailableIfExternalAdaptorPresent = 1014332,
ErrExecuteControl = -1014330,
WarnExecuteControl = 1014330,
ErrValueNotSupported = -1014324,
WarnValueNotSupported = 1014324,
ErrAvailableOnlyWithExternalAdaptor = -1014323,
WarnAvailableOnlyWithExternalAdaptor = 1014323,
ErrControlNotSupportedBySensortype = -1014322,
ErrControlNotSupportedAtConnector = -1014321,
ErrControlNotSupported = -1014320,
WarnControlNotSupported = 1014320,
ErrModeNotSupported = -1014319,
WarnModeNotSupported = 1014319,
ErrKeyNotFound = -1014318,
ErrNoSpaceLeft = -1014317,
ErrAssemblySizeNotSupported = -1014316,
WarnCtrlPeakmodeSignalctrl = 1014310,
WarnCtrlRunmodeSignalctrl = 1014309,
ErrCtrlUnableToSetSignalgenerator = -1014308,
ErrCtrlUnableToControlAlarmUnit = -1014307,
ErrCtrlUnableToControlPeakUnit = -1014306,
ErrCtrlUnableToSetZero = -1014305,
ErrCtrlUnableToControlMathUnit = -1014304,
ErrCtrlUnableToControlSignalGenerator = -1014303,
ErrCtrlUnspecified = -1014302,
ErrWrongSecurityCode = -1014301,
ErrCtrlWrongPar = -1014300,
ErrCalpointTypeProtected = -1014299,
ErrCanRawReceiverEnableDisable = -1014285,
ErrCanResourceOverload = -1014284,
ErrTransmissiontypeSourcechangeNoSupport64Bit = -1014282,
ErrCanWrongIndex = -1014281,
ErrCanValueOnErrorTypeNotSupported = -1014280,
ErrCanValueOnErrorTypeNotSupportedByDataformat = -1014279,
ErrCanSetBitrate = -1014274,
WarnCanBitrateChanged = 1014273,
WarnCanBitrateSamplepointratioChanged = 1014272,
ErrCanAnyDecoderTimeout = -1014271,
WarnCanAnyDecoderTimeout = 1014271,
ErrCanAnyDecoderLossOfSignal = -1014270,
WarnCanAnyDecoderLossOfSignal = 1014270,
ErrCanMappingMassageInactive = -1014269,
ErrCanMappingSourcesOverlap = -1014268,
WarnCanMappingNoSourceDefined = 1014266,
```

```
    ErrOnlyOneMappingsourcePossible = -1014265,
    WarnTimerSetToMinimumPossible = 1014264,
    WarnSignallengthChanged = 1014261,
    ErrCanCcp = -1014259,
    ErrCanWriteCansignal = -1014258,
    ErrCanTransmitSignalreferenceInvalid = -1014257,
    ErrCanTransmitModulereferenceNotLocal = -1014256,
    ErrCanTransmitMsg = -1014255,
    WarnCanTransmissiontypeChanged = 1014254,
    ErrCanTransmissiontypeNotSupported = -1014253,
    ErrCanConnectorNotSupported = -1014252,
    ErrCanTransmitChannel = -1014250,
    ErrCanTransmitUserdefMsg = -1014245,
    ErrCanTransmitByteorderNotSupported = -1014244,
    ErrCanTransmitTable = -1014243,
    ErrCanTransmitBytevalue = -1014242,
    ErrCanTransmitBytecount = -1014241,
    ErrCanTransmitterUserdefMsgSetTable = -1014240,
    ErrCanDecoderModelength64 = -1014238,
    ErrCanDecoderModestartbit63 = -1014237,
    ErrCanDecoderLength64 = -1014236,
    ErrCanDecoderStartbit63 = -1014235,
    ErrCanCombinationIdentifierFrameformat = -1014234,
    ErrCanChannelno = -1014233,
    ErrCanDecoderModeStartbitLength = -1014232,
    ErrCanCombinationStartbitLength = -1014231,
    ErrCanDecoderFormatDouble = -1014230,
    ErrCanDecoderFormatFloat = -1014229,
    ErrCanDecoderFormat32 = -1014228,
    ErrCanDecoderRbufnoInvalid = -1014227,
    ErrCanDecoderRbufnoUsedTwice = -1014226,
    ErrCanDecoderGetTable = -1014225,
    ErrCanDecoderSetTable = -1014224,
    ErrCanDecoderTableFull = -1014223,
    ErrCanDecoderDisable = -1014222,
    ErrCanDecoderEnable = -1014221,
    ErrCanDecoderTableNotValid = -1014220,
    ErrCanDecoderParamCanSignal = -1014219,
    ErrCanNoResponse = -1014218,
    WarnCanResetRequest = 1014217,
    ErrCanTransmitterOverrun = -1014216,
    ErrCanBusErrorTransmitter = -1014215,
    ErrCanBusWarningTransmitter = -1014214,
    WarnCanBusWarningTransmitter = 1014214,
    ErrCanReceiverOverrun = -1014213,
    ErrCanBusOff = -1014212,
    ErrCanBusErrorReceiver = -1014211,
    ErrCanBusWarningReceiver = -1014210,
    WarnCanBusWarningReceiver = 1014210,
    ErrCanInitCansignalParams = -1014209,
    ErrCanGetBitrate = -1014208,
    ErrCanGetState = -1014207,
    ErrCanInvalidHandle = -1014206,
    ErrCanIndex = -1014205,
    ErrCanIoctl = -1014204,
    ErrCanBitrateNotSupported = -1014203,
    ErrCanDisable = -1014202,
    ErrCanEnable = -1014201,
    ErrCanOpendrv = -1014200,
    ErrFbfsAccessLed = -1014191,
    ErrFbfsAccessCan = -1014190,
```

```
ErrDaqSetCarrierfreqMode = -1014143,
ErrDaqSetFreezedFlag = -1014142,
ErrDaqSetRtmeasmode = -1014141,
ErrDaqTrigtime = -1014140,
ErrDaqTriggerunknown = -1014139,
ErrDaqIsosignallistVersion = -1014138,
ErrDaqAmpNull = -1014137,
ErrDaqRtdriverNull = -1014133,
ErrDaqWrongChanType = -1014131,
WarnScalingAnalogoutGainIfAnaloginLintable = 1014128,
ErrDaqGetDemodInfo = -1014127,
ErrDaqSetErrorbuf = -1014125,
ErrDaqDrvSetPtCoeffs = -1014124,
ErrDaqDrvSetColdjunction = -1014123,
ErrDaqDrvSetInitjust = -1014122,
ErrScalingPolynomNotSupportedIfAnalogoutEnabled = -1014121,
ErrScalingPolynomTooMuchCoeffs = -1014120,
ErrScalingPolynomNotEnoughCoeffs = -1014119,
ErrScalingPolynomTooMuchSegments = -1014118,
ErrScalingLintableTooMuchPoints = -1014117,
ErrScalingPolynomNotEnoughSegments = -1014116,
ErrAccessRtdriver = -1014115,
ErrScalingLintableNotEnoughPoints = -1014114,
ErrDaqScalingInternal = -1014113,
ErrDaqScaletype = -1014112,
ErrWrongChannelIndex = -1014111,
ErrWrongSigIndex = -1014110,
ErrWrongConnector = -1014109,
ErrDaqInitjustNotfound = -1014108,
ErrDaqInitjustPointnr = -1014107,
ErrDaqInitjustWrongpar = -1014106,
ErrDaqSigidLen = -1014105,
ErrDaqSigidUnknown = -1014102,
ErrDaqFilter = -1014100,
ErrDaqSignalNotAvailable = -1014099,
ErrDaqAddFbtype = -1014098,
ErrDaqAddFbInstance = -1014097,
ErrDaqSetFbInput = -1014096,
ErrDaqOutrate = -1014095,
ErrDaqSetBufferPars = -1014094,
ErrDaqSetZeroval = -1014093,
ErrDaqFifoParams = -1014092,
ErrDaqFiforeaderCalParams = -1014091,
ErrDaqScalingParams = -1014090,
ErrDaqSetIsoModuleId = -1014089,
ErrDaqSetIsoTransmit = -1014088,
ErrDaqGetIsoTransmit = -1014087,
ErrDaqOverrun = -1014085,
ErrDaqUndefined = -1014082,
ErrDaqFifoSync = -1014070,
ErrDaqOverload = -1014060,
ErrDaqProcess = -1014061,
ErrCriticalHighVoltage = -1014056,
ErrDaqOccf = -1014055,
ErrDaqLimse = -1014050,
ErrDaqNosignalsubscribed = -1014030,
ErrDaqBuffersize = -1014025,
ErrDaqAmpoff = -1014023,
ErrDaqOpenrtdrv = -1014010,
ErrDaqSubscrRunning = -1014006,
ErrDaqSubscrUuid = -1014005,
```


現在位置: Supported devices -> Hbm.Api.QuantumX Namespace -> Hbm.Api.QuantumX.Enums Namespace

```
    ErrDaqSubscrState = -1014004,
    ErrDaqSubscrition = -1014002,
    ErrDataFormatNotSupported = -1010000,
    ErrSrvStatComposer = -1004305,
    ErrMthdSysNorestartfkt = -1004260,
    ErrMthdSysWrongchantype = -1004242,
    ErrMthdSysWrongchan = -1004240,
    ErrMthdSysInvalidfwfile = -1004208,
    ErrMthdSysFilenotfound = -1004201,
    ErrOpenfile = -1004200,
    ErrWriteFile = -1004199,
    ErrReadFile = -1004198,
    ErrParseXmlSettingRefused = -1004160,
    ErrParWrongType = -1004153,
    WarnParNodefset = 1004148,
    ErrParViewidInuse = -1004146,
    ErrParReadingVal = -1004144,
    WarnParEmptyVal = 1004143,
    ErrParViewinvalid = -1004142,
    ErrParWrongxpathstate = -1004140,
    ErrParXpathinvalid = -1004138,
    ErrParNoroot = -1004136,
    WarnParParsing = 1004134,
    ErrParParsing = -1004135,
    ErrParInvalidaddr = -1004118,
    WarnParTagunkown = 1004112,
    ErrParTagunkown = -1004111,
    ErrParOutofrange = -1004110,
    WarnParOutofrange = 1004110,
    ErrParInvalidattr = -1004109,
    WarnParReadonly = 1004108,
    ErrParChoiceinvalid = -1004107,
    ErrParCntxtlost = -1004106,
    ErrParChoicevaluesyntaxinvalid = -1004105,
    ErrParRestorefailed = -1004104,
    ErrParXmlinvalid = -1004103,
    ErrTxtelcontainerTooSmall = -1003001,
    ErrParseProcessingData = -1002002,
    ErrParseInvalidData = -1002001,
    ErrUtilFileRestore = -1001140,
    ErrUtilInvalidFilehandle = -1001136,
    ErrUtilStrconvert = -1001132,
    ErrUtilNoPatternMatch = -1001131,
    ErrEvflgTimeout = -1001130,
    ErrEvflgPthreads = -1001128,
    ErrUtilTimerInvalid = -1001125,
    WarnUtilTimerWasrunning = 1001025,
    ErrUtilLanguage = -1001110,
    ErrComposerTagNotDefined = -1001002,
    ErrNetNetworkadapter = -1000680,
    ErrUdpBindtodevice = -1000658,
    ErrUdpBind = -1000656,
    ErrUdpSetbroadcast = -1000654,
    ErrUdpSetreuseadr = -1000652,
    ErrUdpOpensock = -1000650,
    ErrTcpBind = -1000644,
    ErrTcpListen = -1000642,
    ErrTcpCheckip = -1000638,
    ErrTcpGetadr = -1000636,
    ErrTcpGetport = -1000635,
    ErrTcpSetnodelay = -1000632,
```

```
  ErrTcpSetnonblock = -1000630,
  ErrTcpOpensock = -1000628,
  ErrTcpConnect = -1000626,
  ErrTcpAccept = -1000624,
  ErrTcpConnectFailed = -1000620,
  ErrTcpIpadrAlreadyConnected = -1000615,
  ErrTcpAdrUnknown = -1000612,
  ErrTcpSocketEintr = -1000604,
  ErrTcpSocketHup = -1000603,
  ErrTcpSocketDescriptor = -1000602,
  ErrConRec = -1000524,
  ErrConSending = -1000522,
  ErrConInvalidTeltype = -1000510,
  ErrConAdrresiniFailed = -1000504,
  ErrConConhndlrUuidunknown = -1000500,
  ErrTransReadinghdr = -1000430,
  ErrTransNotfound = -1000420,
  ErrTransCanceled = -1000419,
  ErrTransSendReqTimeout = -1000416,
  ErrTransAckTimeout = -1000417,
  ErrTransReqTimeout = -1000418,
  ErrTransInvalidTelformat = -1000414,
  ErrTransRequestorWrongprottype = -1000404,
  ErrTransResponderWrongprottype = -1000405,
  ErrTransNotForUs = -1000406,
  ErrNoMspChksum = -1000301,
  ErrSrvControl = -1000300,
  ErrSrvNodata = -1000275,
  ErrSrvNoresponseYet = -1000271,
  ErrSrvReqrmthdNoreqtel = -1000265,
  ErrSrvParamerror = -1000255,
  ErrSrvMthdoutUnknownmthd = -1000203,
  ErrSrvMthdoutWrongansw = -1000202,
  ErrSrvMthdoutToomuch = -1000201,
  ErrSrvUnknownsrv = -1000107,
  ErrSrvUnknownmthd = -1000106,
  ErrIndexOutOfRange = -1000000
}
```

Visual Basic

```
Public Enum FrameworkDllResultCodes
  WarnModuleNotReadyForFieldbusAcq = 1515710
  ErrEcatFunctionblockSetPdoStatus = -1515708
  ErrEcatFunctionblockSetPdoIndex = -1515707
  ErrEcatFunctionblockCreateProblem = -1515706
  ErrEcatRtsignaleNotValid = -1515705
  WarnEcatDemuxSignaleNotValid = 1515704
  WarnEcatModuleParamNotReady = 1515703
  ErrEcatModuleStatusNotReadable = -1515702
  WarnEcatCorrectedFwSignaleListSize = 1515701
  ErrEcatCountOfFwSignalsLimitReached = -1515700
  ErrNtpServerNotReached = -1515600
  ErrFwNoNecChannel = -1018005
  ErrFwNoNecDownload = -1018006
  ErrFwNoMspChannel = -1018000
  WarnExcitationcalibrationDataMissing = 1017998
  WarnCalcalibrationDataMissing = 1017997
  ErrMspDataCorrupt = -1017996
  ErrInitjustFromMsp = -1017995
  ErrNoValidInitjust = -1017993
  ErrStoringInitjust = -1017992
  ErrGettingInitjust = -1017991
  ErrPllvolOutofrange = -1017990
  ErrPllvolCorrupt = -1017989
  ErrPllvolFromMsp = -1017988
  ErrDataAddrBustest = -1017987
  ErrDataRateDomainNotSupported = -1016100
  ErrSysFilename = -1016050
  ErrSysFileTooLarge = -1016049
  ErrSysBadDelivery = -1016045
  ErrSysOpenlogfile = -1016040
  ErrSysFrameworkStart = -1016035
  ErrSysOpenCloseWd = -1016030
  ErrSysInvalidTime = -1016025
  ErrSysTooManyEvents = -1016003
  ErrSysCreatingObject = -1016002
  ErrSpiOpenDriver = -1015050
  ErrSpiWakeupDac = -1015049
  ErrSpiWriteDriverDigits = -1015048
  ErrSpiWriteDriverCommand = -1015047
  ErrSpiVoltOutofRange = -1015040
  ErrSpiPlldac = -1015039
  ErrSyncOpenDriver = -1015033
  ErrSyncAnyTsdProblem = -1015032
  ErrSyncNotAllModulesReady = -1015031
  ErrSyncTimesourceDataNotValid = -1015030
  ErrSyncActivateEthercat = -1015029
  ErrSyncActivateIrig = -1015028
  ErrSyncActivatePtp = -1015027
  ErrSyncActivateNtp = -1015026
  ErrSyncActivateAuto = -1015025
  ErrParamIpSubnetmaskSyntax = -1015024
  ErrParamIpAddressSyntax = -1015023
  ErrParamSyncTimesourceNotSupported = -1015022
  ErrSyncCyclediffTooLarge = -1015021
  ErrSyncFreeChannel = -1015019
  ErrSyncRecvCtxRunning = -1015017
  ErrSyncAllocTx = -1015014
  ErrSyncFreeTx = -1015013
  ErrSyncAllocRx = -1015012
```

```
ErrSyncFreeRx = -1015011
ErrSyncDriver = -1015010
ErrSyncNotInSync = -1015009
ErrSyncInitVoltage = -1015008
ErrSyncAdjustPll = -1015007
ErrSyncSyncbufEmpty = -1015006
ErrSyncCtx = -1015005
ErrAdjSetInitCalFacts = -1014999
ErrAdjSetActJustFacts = -1014998
ErrAdjGetActJustFacts = -1014997
ErrAdjGetInitCalFacts = -1014996
ErrAdjGetInitMeasFacts = -1014995
ErrAdjMeasrangeInvalid = -1014994
ErrAdjMeasrangeOverrun = -1014993
ErrAdjWrongConnector = -1014992
ErrThermoWrongConnector = -1014982
ErrConnWrongConnector = -1014801
ErrConnInstanceNull = -1014800
ErrAnalogoutNoconn = -1014603
ErrAnalogoutSigovfl = -1014602
ErrAnalogoutOccf = -1014601
ErrAnalogoutOvfl = -1014600
ErrSignalreferenceDecoderindexNotSupported = -1014515
ErrSignalreferenceDecoderindexLocalNotSupported = -1014514
ErrSignalreferenceSignalgeneratorindexNotSupported = -1014513
ErrSignalreferenceSignalgeneratorindexLocalNotSupported = -1014512
ErrSignalreferenceDigitalioindexLocalNotSupported = -1014511
ErrSignalreferencePeakindexLocalNotSupported = -1014510
ErrSignalreferenceMathindexLocalNotSupported = -1014509
ErrSignalreferenceAlarmindexLocalNotSupported = -1014508
ErrSignalreferenceConnectorLocalNotSupported = -1014507
ErrSignalreferenceSignalindexLocalNotSupported = -1014506
ErrSignalreferenceCannotCreate = -1014505
ErrSignalreferenceUnknown = -1014504
WarnSignalreferenceUnknown = 1014504
ErrParamLevelsIdentical = -1014503
ErrSignalreferenceConnectorNotSupported = -1014502
WarnMaxIsoSignalindexReached = 1014501
ErrSignalreferenceSignalindexNotSupported = -1014500
ErrIndexNotSupported = -1014499
ErrSignalreferenceSignaltypeNotSupportedOrSyntaxError = -1014498
WarnSignalreferenceEmpty = 1014497
ErrModulereferenceSyntax = -1014496
ErrSignalreferencePeakindexNotSupported = -1014495
ErrSignalreferenceMathindexNotSupported = -1014494
ErrSignalreferenceAlarmindexNotSupported = -1014493
ErrModulereferenceNotLocal = -1014492
ErrModulereferenceNotExternal = -1014491
ErrSignalreferenceSyntax = -1014490
ErrPhysunitNotAvailable = -1014489
ErrSignalreferenceAsyncsignalNotAllowedOrSyntaxError = -1014488
ErrSignalreferenceDigitalioindexNotSupported = -1014487
ErrIoctlDevice = -1014414
ErrDeviceNotOpen = -1014413
ErrCloseDevice = -1014412
ErrDeviceAlreadyOpen = -1014411
ErrOpenDevice = -1014410
ErrPinfoNull = -1014406
ErrPointerNull = -1014405
ErrParamsettingsNull = -1014404
ErrParamCalcsyncsignal = -1014403
```


現在位置: Supported devices -> Hbm.Api.QuantumX Namespace -> Hbm.Api.QuantumX.Enums Namespace

```
ErrPplatformNull = -1014402
ErrPampNull = -1014401
ErrPanalogNull = -1014400
ErrTedsData = -1014374
ErrNoSignalGeneratorUnitAvailable = -1014348
ErrNoGateUnitAvailable = -1014347
ErrNoPeakUnitAvailable = -1014346
ErrNoMathUnitAvailable = -1014345
ErrNoAlarmUnitAvailable = -1014344
ErrNoDigitalIoAvailable = -1014343
ErrNoCanConnectorsAvailable = -1014342
ErrNoAnalogOutConnectorsAvalibale = -1014341
ErrNoAnalogInConnectorsAvailable = -1014340
ErrHandlingExternalAdaptor = -1014334
ErrModeNotSupportedBySensortype = -1014333
WarnModeNotSupportedBySensortype = 1014333
ErrNotAvailableIfExternalAdaptorPresent = -1014332
WarnNotAvailableIfExternalAdaptorPresent = 1014332
ErrExecuteControl = -1014330
WarnExecuteControl = 1014330
ErrValueNotSupported = -1014324
WarnValueNotSupported = 1014324
ErrAvailableOnlyWithExternalAdaptor = -1014323
WarnAvailableOnlyWithExternalAdaptor = 1014323
ErrControlNotSupportedBySensortype = -1014322
ErrControlNotSupportedAtConnector = -1014321
ErrControlNotSupported = -1014320
WarnControlNotSupported = 1014320
ErrModeNotSupported = -1014319
WarnModeNotSupported = 1014319
ErrKeyNotFound = -1014318
ErrNoSpaceLeft = -1014317
ErrAssemblySizeNotSupported = -1014316
WarnCtrlPeakmodeSignalctrl = 1014310
WarnCtrlRunmodeSignalctrl = 1014309
ErrCtrlUnableToSetSignalgenerator = -1014308
ErrCtrlUnableToControlAlarmUnit = -1014307
ErrCtrlUnableToControlPeakUnit = -1014306
ErrCtrlUnableToSetZero = -1014305
ErrCtrlUnableToControlMathUnit = -1014304
ErrCtrlUnableToControlSignalGenerator = -1014303
ErrCtrlUnspecified = -1014302
ErrWrongSecurityCode = -1014301
ErrCtrlWrongPar = -1014300
ErrCalpointTypeProtected = -1014299
ErrCanRawReceiverEnableDisable = -1014285
ErrCanResourceOverload = -1014284
ErrTransmissiontypeSourcechangeNoSupport64Bit = -1014282
ErrCanWrongIndex = -1014281
ErrCanValueOnErrorTypeNotSupported = -1014280
ErrCanValueOnErrorTypeNotSupportedByDataformat = -1014279
ErrCanSetBitrate = -1014274
WarnCanBitrateChanged = 1014273
WarnCanBitrateSamplepointratioChanged = 1014272
ErrCanAnyDecoderTimeout = -1014271
WarnCanAnyDecoderTimeout = 1014271
ErrCanAnyDecoderLossOfSignal = -1014270
WarnCanAnyDecoderLossOfSignal = 1014270
ErrCanMappingMassageInactive = -1014269
ErrCanMappingSourcesOverlap = -1014268
WarnCanMappingNoSourceDefined = 1014266
```

```
ErrOnlyOneMappingsourcePossible = -1014265
WarnTimerSetToMinimumPossible = 1014264
WarnSignallengthChanged = 1014261
ErrCanCcp = -1014259
ErrCanWriteCansignal = -1014258
ErrCanTransmitSignalreferenceInvalid = -1014257
ErrCanTransmitModulereferenceNotLocal = -1014256
ErrCanTransmitMsg = -1014255
WarnCanTransmissiontypeChanged = 1014254
ErrCanTransmissiontypeNotSupported = -1014253
ErrCanConnectorNotSupported = -1014252
ErrCanTransmitChannel = -1014250
ErrCanTransmitUserdefMsg = -1014245
ErrCanTransmitByteorderNotSupported = -1014244
ErrCanTransmitTable = -1014243
ErrCanTransmitBytevalue = -1014242
ErrCanTransmitBytecount = -1014241
ErrCanTransmitterUserdefMsgSetTable = -1014240
ErrCanDecoderModelength64 = -1014238
ErrCanDecoderModestartbit63 = -1014237
ErrCanDecoderLength64 = -1014236
ErrCanDecoderStartbit63 = -1014235
ErrCanCombinationIdentifierFrameformat = -1014234
ErrCanChannelno = -1014233
ErrCanDecoderModeStartbitLength = -1014232
ErrCanCombinationStartbitLength = -1014231
ErrCanDecoderFormatDouble = -1014230
ErrCanDecoderFormatFloat = -1014229
ErrCanDecoderFormat32 = -1014228
ErrCanDecoderRbufnoInvalid = -1014227
ErrCanDecoderRbufnoUsedTwice = -1014226
ErrCanDecoderGetTable = -1014225
ErrCanDecoderSetTable = -1014224
ErrCanDecoderTableFull = -1014223
ErrCanDecoderDisable = -1014222
ErrCanDecoderEnable = -1014221
ErrCanDecoderTableNotValid = -1014220
ErrCanDecoderParamCanSignal = -1014219
ErrCanNoResponse = -1014218
WarnCanResetRequest = 1014217
ErrCanTransmitterOverrun = -1014216
ErrCanBusErrorTransmitter = -1014215
ErrCanBusWarningTransmitter = -1014214
WarnCanBusWarningTransmitter = 1014214
ErrCanReceiverOverrun = -1014213
ErrCanBusOff = -1014212
ErrCanBusErrorReceiver = -1014211
ErrCanBusWarningReceiver = -1014210
WarnCanBusWarningReceiver = 1014210
ErrCanInitCansignalParams = -1014209
ErrCanGetBitrate = -1014208
ErrCanGetState = -1014207
ErrCanInvalidHandle = -1014206
ErrCanIndex = -1014205
ErrCanIoctl = -1014204
ErrCanBitrateNotSupported = -1014203
ErrCanDisable = -1014202
ErrCanEnable = -1014201
ErrCanOpendrv = -1014200
ErrFbfsAccessLed = -1014191
ErrFbfsAccessCan = -1014190
```


現在位置: Supported devices -> Hbm.Api.QuantumX Namespace -> Hbm.Api.QuantumX.Enums Namespace

```
ErrDaqSetCarrierfreqMode = -1014143
ErrDaqSetFreezedFlag = -1014142
ErrDaqSetRtmeasmode = -1014141
ErrDaqTrigtime = -1014140
ErrDaqTriggerunknown = -1014139
ErrDaqIsosignallistVersion = -1014138
ErrDaqAmpNull = -1014137
ErrDaqRtdriverNull = -1014133
ErrDaqWrongChanType = -1014131
WarnScalingAnalogoutGainIfAnaloginLintable = 1014128
ErrDaqGetDemodInfo = -1014127
ErrDaqSetErrorbuf = -1014125
ErrDaqDrvSetPtCoeffs = -1014124
ErrDaqDrvSetColdjunction = -1014123
ErrDaqDrvSetInitjust = -1014122
ErrScalingPolynomNotSupportedIfAnalogoutEnabled = -1014121
ErrScalingPolynomTooMuchCoeffs = -1014120
ErrScalingPolynomNotEnoughCoeffs = -1014119
ErrScalingPolynomTooMuchSegments = -1014118
ErrScalingLintableTooMuchPoints = -1014117
ErrScalingPolynomNotEnoughSegments = -1014116
ErrAccessRtdriver = -1014115
ErrScalingLintableNotEnoughPoints = -1014114
ErrDaqScalingInternal = -1014113
ErrDaqScaletype = -1014112
ErrWrongChannelIndex = -1014111
ErrWrongSigIndex = -1014110
ErrWrongConnector = -1014109
ErrDaqInitjustNotfound = -1014108
ErrDaqInitjustPointnr = -1014107
ErrDaqInitjustWrongpar = -1014106
ErrDaqSigidLen = -1014105
ErrDaqSigidUnknown = -1014102
ErrDaqFilter = -1014100
ErrDaqSignalNotAvailable = -1014099
ErrDaqAddFbtype = -1014098
ErrDaqAddFbInstance = -1014097
ErrDaqSetFbInput = -1014096
ErrDaqOutrate = -1014095
ErrDaqSetBufferPars = -1014094
ErrDaqSetZeroval = -1014093
ErrDaqFifoParams = -1014092
ErrDaqFiforeaderCalParams = -1014091
ErrDaqScalingParams = -1014090
ErrDaqSetIsoModuleId = -1014089
ErrDaqSetIsoTransmit = -1014088
ErrDaqGetIsoTransmit = -1014087
ErrDaqOverrun = -1014085
ErrDaqUndefined = -1014082
ErrDaqFifoSync = -1014070
ErrDaqOverload = -1014060
ErrDaqProcess = -1014061
ErrCriticalHighVoltage = -1014056
ErrDaqOccf = -1014055
ErrDaqLimse = -1014050
ErrDaqNosignalsubscribed = -1014030
ErrDaqBuffersize = -1014025
ErrDaqAmpoff = -1014023
ErrDaqOpenrtdrv = -1014010
ErrDaqSubscrRunning = -1014006
ErrDaqSubscrUuid = -1014005
```

```
ErrDaqSubscrState = -1014004
ErrDaqSubscrition = -1014002
ErrDataFormatNotSupported = -1010000
ErrSrvStatComposer = -1004305
ErrMthdSysNorestartfkt = -1004260
ErrMthdSysWrongchantype = -1004242
ErrMthdSysWrongchan = -1004240
ErrMthdSysInvalidfwfile = -1004208
ErrMthdSysFilenotfound = -1004201
ErrOpenfile = -1004200
ErrWriteFile = -1004199
ErrReadFile = -1004198
ErrParseXmlSettingRefused = -1004160
ErrParWrongType = -1004153
WarnParNodefset = 1004148
ErrParViewidInuse = -1004146
ErrParReadingVal = -1004144
WarnParEmptyVal = 1004143
ErrParViewinvalid = -1004142
ErrParWrongxpathstate = -1004140
ErrParXpathinvalid = -1004138
ErrParNoroot = -1004136
WarnParParsing = 1004134
ErrParParsing = -1004135
ErrParInvalidaddr = -1004118
WarnParTagunkown = 1004112
ErrParTagunkown = -1004111
ErrParOutofrange = -1004110
WarnParOutofrange = 1004110
ErrParInvalidattr = -1004109
WarnParReadonly = 1004108
ErrParChoiceinvalid = -1004107
ErrParCntxtlost = -1004106
ErrParChoicevaluesyntaxinvalid = -1004105
ErrParRestorefailed = -1004104
ErrParXmlinvalid = -1004103
ErrTxtelcontainerTooSmall = -1003001
ErrParseProcessingData = -1002002
ErrParseInvalidData = -1002001
ErrUtilFileRestore = -1001140
ErrUtilInvalidFilehandle = -1001136
ErrUtilStrconvert = -1001132
ErrUtilNoPatternMatch = -1001131
ErrEvflgTimeout = -1001130
ErrEvflgPthreads = -1001128
ErrUtilTimerInvalid = -1001125
WarnUtilTimerWasrunning = 1001025
ErrUtilLanguage = -1001110
ErrComposerTagNotDefined = -1001002
ErrNetNetworkadapter = -1000680
ErrUdpBindtodevice = -1000658
ErrUdpBind = -1000656
ErrUdpSetbroadcast = -1000654
ErrUdpSetreuseadr = -1000652
ErrUdpOpensock = -1000650
ErrTcpBind = -1000644
ErrTcpListen = -1000642
ErrTcpCheckip = -1000638
ErrTcpGetadr = -1000636
ErrTcpGetport = -1000635
ErrTcpSetnodelay = -1000632
```

```
    ErrTcpSetnonblock = -1000630
    ErrTcpOpensock = -1000628
    ErrTcpConnect = -1000626
    ErrTcpAccept = -1000624
    ErrTcpConnectFailed = -1000620
    ErrTcpIpadrAlreadyConnected = -1000615
    ErrTcpAdrUnknown = -1000612
    ErrTcpSocketEintr = -1000604
    ErrTcpSocketHup = -1000603
    ErrTcpSocketDescriptor = -1000602
    ErrConRec = -1000524
    ErrConSending = -1000522
    ErrConInvalidTeltype = -1000510
    ErrConAdrresiniFailed = -1000504
    ErrConConhndlrUuidunknown = -1000500
    ErrTransReadinghdr = -1000430
    ErrTransNotfound = -1000420
    ErrTransCanceled = -1000419
    ErrTransSendReqTimeout = -1000416
    ErrTransAckTimeout = -1000417
    ErrTransReqTimeout = -1000418
    ErrTransInvalidTelformat = -1000414
    ErrTransRequestorWrongprottype = -1000404
    ErrTransResponderWrongprottype = -1000405
    ErrTransNotForUs = -1000406
    ErrNoMspChksum = -1000301
    ErrSrvControl = -1000300
    ErrSrvNodata = -1000275
    ErrSrvNoresponseYet = -1000271
    ErrSrvReqrmthdNoreqtel = -1000265
    ErrSrvParamerror = -1000255
    ErrSrvMthdoutUnknownmthd = -1000203
    ErrSrvMthdoutWrongansw = -1000202
    ErrSrvMthdoutToomuch = -1000201
    ErrSrvUnknownsrv = -1000107
    ErrSrvUnknownmthd = -1000106
    ErrIndexOutOfRange = -1000000
End Enum
```

ファイル

FrameworkDllResultCodes.cs

Hbm.Api.QuantumX.Enums.FrameworkError Enumeration

考えられるQuantumXフレームワークのエラー(ネガティブ)および警告(ポジティブ)コード。

C++

```
enum FrameworkError {
  ERR_NO_SUCCESS = -1,
  ERR_SUCCESS = 0,
  ERR_NOT_SUPPORTED = -110,
  WARN_UNSPECIFIED = 1000,
  WARN_VALUE_CHANGED = 9,
  INTERNAL_ACTIVATE_PROFINET = 1,
  INTERNAL_ACTIVATE_PROFINET_DONE = 2,
  ERR_FIELDBUS_INIT_DRIVER = -1000,
  ERR_FIELDBUS_ACCESS_OD = -1001,
  ERR_FIELDBUS_READ_ODXML = -1002,
  ERR_FIELDBUS_START_OD = -1005,
  ERR_FIELDBUS_STOP_OD = -1006,
  ERR_FIELDBUS = -1010,
  ERR_FIELDBUS_STATECHANGE = -1106,
  ERR_FIELDBUS_INVALIDTIME = -1107,
  ERR_FIELDBUS_UNAVAILABLPAR = -1109,
  ERR_INVALID_REVISION_STRING = -9000,
  ERR_MODULETYPE_STRING_EMPTY = -9001,
  ERR_MODULELABEL_STRING_EMPTY = -9002,
  ERR_INVALID_MAC_ADDRESS_STRING = -9003,
  ERR_SCALING_NOT_SUPPORTED = -100101,
  WARN_SCALING_CHANGED_TO_STANDARD_FOR_SENSORTYPE = 100102,
  ERR_SCALING_NOT_SUPPORTED_BY_SENSORTYPE = -100103,
  ERR_SCALING_INVALID_TYPE = -100104,
  ERR_SCALING_STRAINGAGE_NOT_SUPPORTED = -100105,
  ERR_RESTORING_FACTORY_SCALING = -100106,
  ERR_SCALING_DIV_BY_ZERO = -100110,
  ERR_FILTERTYPE_NOT_SUPPORTED = -100201,
  ERR_FILTERCHARACT_NOT_SUPPORTED = -100202,
  ERR_FILTERFREQ_NOT_SUPPORTED = -100203,
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_2ND_SIGNAL = -100204,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_2ND_SIGNAL = 100204,
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_CLASSIC = -100205,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_CLASSIC = 100205,
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_MODULE = -100206,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_MODULE = 100206,
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = -100207,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = 100207,
  ERR_IIR_FILTER_NOT_SUPPORTED_BY_MODULE = -100208,
  WARN_IIR_FILTER_NOT_SUPPORTED_BY_MODULE = 100208,
  ERR_IIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = -100209,
  WARN_IIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = 100209,
  WARN_FILTERCHAR_CHANGED_TO_IIR_BESSEL_FIR_NO_HIGHSPEED = 100210,
  WARN_FILTERCHAR_CHANGED_TO_IIR_BUTTERWORTH_FIR_NO_HIGHSPEED = 100211,
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_HIGHSPEED = -100212,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_HIGHSPEED = 100212,
  WARN_FILTER_SWITCHED_OFF_FIR_NO_HIGHSPEED = 100213,
  WARN_FILTER_CHARACT_NOT_SUPPORTED_CHANGED_TO_OFF = 100214,
  WARN_FILTER_CHARACT_NOT_SUPPORTED_BY_2ND_SIGNAL_CHANGED_TO_OFF = 100215,
  WARN_FILTER_CHARACT_NOT_SUPPORTED_BY_HIGHSPEED_CHANGED_TO_OFF = 100216,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_CLASSIC_CHANGED_TO_OFF = 100217,
  WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_IIR_BESSEL = 100218,
```

```
    WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_IIR_BUTTERWORTH = 100219,
    WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_FIR_LINEARPHASE = 100220,
    WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_FIR_BUTTERWORTH = 100221,
    ERR_CONNMODE_NOT_SUPPORTED = -100450,
    ERR_SENSORTYPE_NOT_SUPPORTED = -100506,
    ERR_OUTPUTRATE_VS_FILTER_NOT_SUPPORTED = -100510,
    ERR_OUTPUTRATE_NOT_SUPPORTED = -100511,
    ERR_PHYSICAL_UNIT_NOT_NATIVE = -100520,
    WARN_PHYSICAL_UNIT_NOT_NATIVE = 100520,
    WARN_PHYSICAL_UNIT_CHANGED_TO_NATIVE = 100521,
    ERR_VOLTAGERANGE_NOT_SUPP = -100700,
    ERR_CURRENTRANGE_NOT_SUPP = -100710,
    ERR_WIRING_NOT_SUPPORTED = -100802,
    ERR_WIRING_NOT_SUPPORTED_CHECK_HW_REVISION = -100803,
    ERR_EXCITFREQ_NOT_SUPPORTED = -100804,
    ERR_EXCITAMPL_NOT_SUPPORTED = -100806,
    ERR_IMPEDANCE_NOT_SUPPORTED = -100807,
    ERR_PT_RESISTANCE_NOT_SUPPORTED = -100850,
    ERR_TC_TYPE_NOT_SUPPORTED = -100860,
    ERR_RESISTOR_NOT_SUPPORTED = -100904,
    ERR_SSI_CODING_NOT_SUPPORTED = -101002,
    ERR_PRESSURE_RANGE_NOT_SUPPORTED = -101010,
    ERR_PRESSURE_MAX_AVAILABLE = -101011,
    ERR_PRESSURE_TYPE_ONLY_ABSOLUTE_SUPPORTED = -101012,
    ERR_PRESSURE_TYPE_ONLY_RELATIVE_SUPPORTED = -101013,
    ERR_PRESSURE_TYPE_NOT_SUPPORTED = -101014,
    ERR_PRESSURE_SENSOR_NOT_AVAILABLE = -101015,
    ERR_PRESSURE_MIN_AVAILABLE = -101016,
    ERR_FREQUENCYRANGE_NOT_SUPPORTED = -101020,
    WARN_REQUESTED_MAX_COUNTER_EXCEEDS_POSSIBLE = 101010,
    ERR_SENSORSUPPLY_NOT_SUPPORTED = -101100,
    ERR_SET_SENSOR_SUPPLY_NOT_ALLOWED_IF_USED_BY_PARAMETER_SET = -101102,
    ERR_SETZERO_INHIBITED = -101110,
    ERR_SETZERO_NOT_SUPPORTED = -101111,
    ERR_ANALOG_OUT_SETZERO = -101202,
    ERR_PARAMSETTINGS_IND_COUNTER = -102017,
    ERR_PARAMSETTINGS_IND_FREQUENCY = -102018,
    ERR_PARAMSETTINGS_BRIDGE = -102020,
    ERR_PARAMSETTINGS_VOLTAGE = -102021,
    ERR_PARAMSETTINGS_CURRENT = -102022,
    ERR_PARAMSETTINGS_THERMOCOUPLE = -102023,
    ERR_PARAMSETTINGS_PT_ELEMENT = -102024,
    ERR_PARAMSETTINGS_POTI = -102025,
    ERR_PARAMSETTINGS_SSI = -102026,
    ERR_PARAMSETTINGS_COUNTER = -102027,
    ERR_PARAMSETTINGS_FREQUENCY = -102028,
    ERR_PARAMSETTINGS_INDBRIDGE = -102029,
    ERR_PARAMSETTINGS_LVDT = -102030,
    ERR_PARAMSETTINGS_PRESSURE = -102031,
    ERR_PARAMSETTINGS_IEPE = -102032,
    ERR_PARAMSETTINGS_PWM = -102033,
    ERR_PARAMSETTINGS_PULSEWIDTH = -102034,
    ERR_PARAMSETTINGS_PERIOD = -102035,
    ERR_PARAMSETTINGS_RESISTOR = -102036,
    ERR_IRIG_FORMAT_NOT_SUPPORTED = -102037,
    WARN_CONNECTOR_SET_TO_VOLTAGE_BECAUSE_ANOTHER_SET_TO_IRIG = 102038,
    WARN_IRIG_SENSOR_CAN_ONLY_BE_ACTIVE_ONCE = 102039,
    ERR_PARAMSETTINGS_IND_CRANK_COUNTER = -102040,
    ERR_PARAMSETTINGS_IND_CRANK_FREQUENCY = -102041,
    ERR_GAP_POSITION_SYNTAX = -102042,
    ERR_GAP_POSITION_ZERO_NOT_ALLOWED = -102043,
```

```
ERR_GAP_POSITION_REPETITION_NOT_ALLOWED = -102044,
ERR_GAP_POSITION_VALUE_GREATER_THAN_NUMBER_OF_TEETH = -102045,
ERR_INDEX_POSITION_VALUE_GREATER_THAN_NUMBER_OF_TEETH = -102046,
ERR_TOO_MUCH_GAP_POSITIONS_DEFINED = -102047,
ERR_PULSES_PER_ROUND_OUT_OF_RANGE = -102048,
WARN_PULSES_PER_ROUND_OUT_OF_RANGE_SET_TO_ZERO = 102048,
ERR_INDEX_SHIFT_OUT_OF_RANGE = -102049,
WARN_INDEX_SHIFT_OUT_OF_RANGE_SET_TO_ZERO = 102049,
WARN_INDEX_DIVISOR_OUT_OF_RANGE_SET_TO_MINIMUM = 102050,
WARN_INDEX_DIVISOR_OUT_OF_RANGE_SET_TO_MAXIMUM = 102051,
ERR_PARAMSETTINGS_CRANK_COUNTER = -102052,
ERR_PARAMSETTINGS_CRANK_FREQUENCY = -102053,
ERR_PARAM_ZERO_NOT_VALID = -103000,
ERR_PREVIOUS_SETTING_RESTORED = -104010,
ERR_PARAMETERIZATION_REFUSED = -104011,
ERR_SCALING_LINTABLE_WRONG_NUMBER_OF_POINTS = -104099,
WARN_SCALING_NR_OF_SEGMENTS_CHANGED = 104100,
WARN_SCALING_NR_OF_COEFFS_CHANGED = 104101,
ERR_SPACE_IN_STRING_NOT_ALLOWED = -104103,
ERR_STRING_SIZE_TOO_LONG = -104104,
ERR_SCM_HV_NOT_SUPPORTED_BY_VOLTAGE_RANGE = -104105,
ERR_VOLTAGE_EXCEEDS_MEAS_RANGE = -104106,
ERR_HIGHVOLTAGE_ADAPTOR_NOT_SUPPORTED = -104107,
ERR_VOLTAGE_RANGE_OR_HIGHVOLTAGE_ADAPTOR_NOT_SUPPORTED = -104108,
WARN_EXTERNAL_HIGHVOLTAGE_ADAPTOR_NOT_CONNECTED = 104109,
ERR_EXTERNAL_HIGHVOLTAGE_ADAPTOR_NOT_CONNECTED = -104109,
WARN_EXTERNAL_QUARTER_BRIDGE_ADAPTOR_NOT_CONNECTED = 104110,
ERR_EXTERNAL_QUARTER_BRIDGE_ADAPTOR_NOT_CONNECTED = -104110,
ERR_EXTERNAL_ADAPTOR_COEFFS_NOT_AVAILABLE = -104111,
WARN_OUTPUTUNIT_INCOMPATIBLE_TO_SENSORUNIT = 104112,
WARN_OUTPUTUNIT_DISABLED_BECAUSE_PHYS_UNIT_USER_DEF = 104113,
WARN_OUTPUTUNIT_IS_USER_DEFINED = 104114,
ERR_AUTOCAL_REPTIME_NOT_SUPPORTED = -104200,
ERR_AUTOCAL_NOT_SUPPORTED_SENSOR = -104201,
ERR_AUTOCAL_DISABLED = -104203,
ERR_AUTOCAL_SENSOR_NOT_CONNECTED = -104204,
ERR_AUTOCAL_NOT_SUPPORTED_MODULE = -104205,
ERR_ADJUSTMENT_SENSOR_NOT_SUPPORTED = -104206,
WARN_ADJUSTMENT_SENSOR_NOT_SUPPORTED = 104206,
WARN_ADJUSTMENT_REPTIME_TOO_SHORT = 104207,
WARN_ADJUSTMENT_REPTIME_TOO_LONG = 104208,
WARN_ADJUSTMENT_NOT_DEFAULT = 104209,
ERR_AUTOADJUST_NOT_SUPPORTED_SENSOR = -104211,
ERR_AUTOCAL_CYCLE_IS_RUNNING_REPEAT_LATER = -104212,
ERR_TEDS_NO_SENSOR_TEMPLATE = -105000,
ERR_TEDS_NO_DATA_OR_INVALID = -105001,
ERR_TEDS_NO_MIN_PHYS_VAL = -105002,
ERR_TEDS_NO_MAX_PHYS_VAL = -105003,
ERR_TEDS_NO_MIN_ELEC_VAL = -105004,
ERR_TEDS_NO_MAX_ELEC_VAL = -105005,
ERR_TEDS_NO_MDEF_UCC = -105007,
ERR_TEDS_CALCURVE_COEFFS_INVALID = -105008,
ERR_TEDS_NODATA = -105010,
ERR_TEDS_NOPARSEBIN = -105014,
ERR_TEDS_NOAMP = -105018,
ERR_TEDS_NO_EXCITE_AMPL_NOM = -105020,
ERR_TEDS_NO_EXCITE_AMPL_MIN = -105022,
ERR_TEDS_NO_EXCITE_AMPL_MAX = -105024,
ERR_TEDS_NO_ATTACHED_SERIALNUM = -105025,
ERR_TEDS_NO_SENSOR_IMPED = -105030,
ERR_TEDS_GAUGEFACTOR = -105031,
```

```
    ERR_TEDS_NO_BRIDGETYPE = -105032,
    ERR_TEDS_TC_TYPE = -105034,
    ERR_TEDS_PT_TYPE = -105036,
    ERR_TEDS_LOWPASSFILTER = -105038,
    ERR_TEDS_ZEROSPANSCALING = -105040,
    ERR_TEDS_GAINSCALING = -105041,
    ERR_TEDS_PARAM = -105050,
    ERR_TEDS_NOT_SUPPORTED_BY_CONNECTOR = -105051,
    WARN_TEDS_NOT_SUPPORTED_CHANGED_TO_IGNORE = 105052,
    ERR_TEDS_NO_CALTABLE = -105054,
    ERR_TEDS_NO_CALCURVE = -105055,
    ERR_TEDS_CALCURVE_POWER_INVALID = -105056,
    ERR_TEDS_CALTABLE_NOT_ENOUGH_POINTS = -105059,
    ERR_TEDS_CALTABLE_TOO_MUCH_POINTS = -105060,
    ERR_TEDS_BUFFER_TOO_SMALL = -105061,
    ERR_TEDS_WRITE_NOT_ALLOWED = -105062,
    ERR_TEDS_TID_DONT_MATCH = -105063,
    ERR_TEDS_SCALINGTYPE_NOT_SUPPORTED = -105100,
    ERR_TEDS_SCALING_PARAMETER_INVALID = -105101,
    ERR_TEDS_UNIT_CONVERSION_NOT_SUPPORTED = -105102,
    ERR_SPEEDMODE_CHANGING_NOT_POSSIBLE_SWITCH_OFF = -105201,
    ERR_SPEEDMODE_TOO_MANY_CONNECTORS_HIGH = -105202,
    ERR_SPEEDMODE_MIXED_MODE_NOT_POSSIBLE = -105204,
    WARN_SPEEDMODE_CHANGED_TO_HIGH = 105205,
    WARN_HIGHSPEEDMODE_NOT_SUPPORTED = 105206,
    WARN_SCALING_PARAMS_DONT_SUPPORT_HIGHSPEEDMODE = 105207,
    ERR_SPEEDMODE_HIGH_NOT_SUPPORTED = -105208,
    WARN_SENSOR_PARAMS_DONT_SUPPORT_HIGHSPEEDMODE = 105209,
    ERR_CONNECTORSOURCE = -105210,
    ERR_PHYSUNIT_NOT_FOUND = -106000,
    ERR_PHYSUNIT_NOT_RELATED = -106004,
    ERR_PHYSUNIT_SCALING_FACTOR_ZERO = -106006,
    WARN_SUPPLY_VOLTAGE_SWITCHED_OFF = 119000,
    WARN_WIRING_ALTERED = 119001,
    WARN_SUPPLY_VOLTAGE_CHANGED_TO_MAX_POSSIBLE = 119002,
    WARN_EXCITATION_FREQ_CHANGED = 119003,
    WARN_EXCITATION_AMPL_CHANGED = 119004,
    ERR_SUPPLY_VOLTAGE_BELOW_MINIMUM = -119005,
    WARN_SSI_CLK_FREQ_CHANGED = 119005,
    WARN_SSI_RESOLUTION_CHANGED = 119006,
    WARN_FILTER_CUTOFF_FREQ_CHANGED = 119010,
    WARN_FILTERTYPE_CHANGED = 119011,
    WARN_FILTERCHAR_CHANGED = 119012,
    WARN_OUTPUTRATE_CHANGED = 119015,
    ERR_GLITCHFILTER_NOT_SUPPORTED = -119024,
    WARN_GLITCHFILTER_CHANGED = 119025,
    ERRHAL_ERROR = -120000,
    ERRHAL_OPENMSPDRV = -120100,
    ERRHAL_OPENCPLDDRV = -120110,
    ERRHAL_GETAMPTEMP = -120120,
    ERRHAL_GETAMPTEMPCAL = -120121,
    ERRHAL_READ_I2C = -120125,
    ERRHAL_WRITE_I2C = -120126,
    ERRHAL_WRITE_MSP = -120127,
    ERRHAL_ENABLECHN = -120130,
    ERRHAL_SETMEASINPUT = -120134,
    ERRHAL_SETSENSCIRCUIT = -120135,
    ERRHAL_SETCALINPUT = -120140,
    ERRHAL_SETGAINCAL = -120145,
    ERRHAL_SETGAINMEAS = -120150,
    ERRHAL_SET_SENSORSUPPLY = -120155,
```

```
ERRHAL_SETFEEDBACK = -120159,
ERRHAL_SETEXCIT = -120160,
ERRHAL_GETSNAMP = -120165,
WARN_HAL_GETSNAMP = 120166,
ERRHAL_SETSNAMP = -120170,
ERRHAL_GETHWRAMP = -120175,
WARN_HAL_GETHWRAMP = 120176,
ERRHAL_SETHWRAMP = -120180,
ERRHAL_GETTID = -120185,
ERRHAL_GETTEDSDATHW = -120189,
ERRHAL_GETTEDSDAT = -120190,
ERRHAL_TEDSCHECKSUM = -120191,
ERRHAL_NOTEDS = -120192,
ERRHAL_TEDS_ALLZERO = -120193,
ERRHAL_TEDSCHECKSUMINSERT = -120194,
ERRHAL_SETTEDSDAT = -120195,
ERRHAL_SETTEDSCON = -120196,
ERRHAL_SIZETEDSDAT = -120197,
ERRHAL_SETLEDAMP = -120200,
ERRHAL_SETSYSLEDAMP = -120201,
ERRHAL_WRSETTING = -120205,
WARNHAL_WRSETTING = 120205,
ERRHAL_SETTING_NOT_AVAILABLE = -120206,
ERRHAL_SETADCCLK = -120210,
ERRHAL_GETMAXSRAMP = -120211,
ERRHAL_SET_FM_CONF = -120212,
ERRHAL_SET_SSI_CONF = -120213,
ERRHAL_GETMSPVERAMP = -120215,
ERRHAL_GETMSPSTATAMP = -120216,
ERRHAL_SYNCADC = -120220,
ERRHAL_SETCHNPROP = -120225,
ERRHAL_GETMSPREG = -120230,
ERRHAL_SETMSPREG = -120235,
ERRHAL_SETTSTSIGMEAS = -120240,
ERRHAL_SETTSTSIGCAL = -120245,
ERRHAL_SETSIMUVAL = -120246,
ERRHAL_SETSIMUVALFORMAT = -120248,
ERRHAL_SETSIMUVALTYPE = -120249,
ERRHAL_GETSNAPSHOT = -120250,
ERRHAL_SNAPSHOT_MAXDATA = -120251,
ERRHAL_CALCPHASE = -120252,
ERRHAL_NO_VALID_SNAPSHOTDATA = -120253,
ERRHAL_GETCALDATHW = -120256,
ERRHAL_GETCALDAT = -120257,
ERRHAL_SETCALDAT = -120258,
ERRHAL_SETCSURCAL = -120260,
ERRHAL_GETCSURCAL = -120265,
ERRHAL_LDRCHKSUM = -120266,
ERRHAL_STORESENSSUPPLYOFFS = -120267,
ERRHAL_GETSENSSUPPLYOFFS = -120268,
WARN_HAL_GETCSURCAL = 120269,
ERRHAL_SETMEASVALINVAL = -120270,
WARN_HAL_GETSENSSUPPLYOFFS = 120271,
ERRHAL_SELFPGACHN = -120272,
ERRHAL_REDDATRATE = -120274,
ERRHAL_RATE_NOT_SUP_BY_FPGA = -120276,
ERRHAL_FB_NOT_AVAILABLE = -120277,
ERRHAL_FB_UNKNOWN_PROPERTY = -120278,
ERRHAL_FPGATEST_NOT_SUPPORTED = -120279,
WARN_HAL_GETPOTIWIPERFROMMSP = 120280,
ERRHAL_GETPOTIWIPERFROMMSP = -120281,
```

```
    ERRHAL_POTIWIPERCOMPARE = -120282,
    ERRHAL_SETIEPEHP = -120283,
    ERRHAL_FB_ARBSIGGEN_BUFFER_BUSY = -120284,
    ERRHAL_GETHWIDENTPF = -120300,
    WARN_HAL_GETHWIDENTPF = 120301,
    ERRHAL_SETHWIDENTPF = -120305,
    ERRHAL_SETADDHWREVPF = -120306,
    ERRHAL_SETADDHWIDENTPF = -120307,
    ERRHAL_GETHWRPF = -120310,
    ERRHAL_GETADDHWRPF = -120311,
    ERRHAL_GETADDHWIDENTPF = -120312,
    WARN_HAL_GETHWRPF = 120313,
    WARN_HAL_GETADDHWRPF = 120314,
    ERRHAL_SETHWRPF = -120315,
    WARN_HAL_GETADDHWIDENTPF = 120316,
    ERRHAL_GETPLLVOLTAGE = -120317,
    WARN_HAL_GETPLLVOLTAGE = 120318,
    ERRHAL_SETPLLVOLTAGE = -120319,
    ERRHAL_SETMACADDR = -120320,
    ERRHAL_GETMACADDR = -120325,
    ERRHAL_SETLEDPF = -120330,
    ERRHAL_GETPFTEMP = -120335,
    ERRHAL_GETMODVOL = -120340,
    ERRHAL_GETMODCUR = -120345,
    ERRHAL_GETEXTCUR = -120350,
    ERRHAL_GETMSPAPPLDATEPF = -120358,
    ERRHAL_GETMSPLOADERDATEPF = -120359,
    ERRHAL_GETMSPVERPF = -120360,
    ERRHAL_GETMSPSTATPF = -120361,
    ERRHAL_GETLIMCURR = -120362,
    WARN_HAL_GETMSPVERPF = 120363,
    ERRHAL_SETTSTSIGGEN = -120365,
    ERRHAL_STRTDATSTRM = -120367,
    ERRHAL_STOPDATSTRM = -120369,
    ERRHAL_IISDUMDAT = -120371,
    ERRHAL_ENIIS = -120373,
    ERRHAL_RSTIIS = -120375,
    ERRHAL_CHKFLHETRY = -120377,
    ERRHAL_GETMAXSRPF = -120379,
    ERRHAL_OPENSNAPSHOTDRV = -120405,
    ERRHAL_CLOSESNAPSHOTDRV = -120406,
    ERRHAL_ENCXUPPER = -120408,
    ERRHAL_SIGNUMBER = -120410,
    ERRHAL_FOSDELAY = -120412,
    ERRHAL_SETSCALPARS = -120414,
    ERRHAL_COMPENSATION_OFFSET = -120415,
    ERRHAL_DACINPUTTYPE = -120416,
    ERRHAL_UNKNOWNSYNCCODE = -120417,
    ERRHAL_ADDFUNCMATH = -120418,
    ERRHAL_UNKNOWN_MATH_TYPE = -120420,
    ERRHAL_PEAK_OUTRATE = -120422,
    ERRHAL_CX_SYNC_TEST = -120424,
    ERRHAL_SETZERO_NO_VALID_MEASVAL = -120426,
    ERRHAL_SET_FILTER_FIR = -120428,
    ERRHAL_PARAM_IS_ZERO = -120429,
    ERRHAL_FIR_TAB_CNT = -120430,
    ERRHAL_FIR_TAB_CNT_SUM = -120431,
    ERRHAL_SETDEMODFACTORS = -120432,
    ERRHAL_SETFIRDEMODFILTER = -120433,
    ERRHAL_SETPOLYCOEFFSTOFPGA = -120434,
    ERRHAL_FPGAOPENDRV = -120500,
```

```
ERRHAL_FPGACLOSEDRV = -120501,
ERRHAL_FPGABUSTST = -120502,
ERRHAL_FPGAREAD = -120503,
ERRHAL_FPGAWRITE = -120504,
ERRHAL_FPGAINITI2C = -120505,
ERRHAL_FPGARSTI2C = -120506,
ERRHAL_FPGARSTHZR = -120507,
ERRHAL_FPGAMMAP = -120509,
ERRHAL_FPGAMUNMAP = -120510,
ERRHAL_FPGAGETTIME = -120511,
ERRHAL_FPGARESET = -120512,
ERRHAL_FPGAIOCTL = -120513,
ERRHAL_MSPGETTMP = -120700,
ERRHAL_MSPDISOVRCURRPROT = -120701,
ERRHAL_MSPGETALLSWITCHES = -120702,
ERRHAL_MSPSETALLSWITCHES = -120703,
ERRHAL_MSPMEAS = -120705,
ERRHAL_MSPRDCTRREG = -120710,
ERRHAL_MSPWRCTRREG = -120715,
ERRHAL_MSPWRREGBIT = -120720,
ERRHAL_MSPSETSHUNT = -120721,
ERRHAL_MSPSETVARIANT = -120722,
ERRHAL_MSPSETWIRING = -120723,
ERRHAL_MSPWRINFO = -120725,
ERRHAL_MSPRDINFO = -120730,
ERRHAL_MSPSETPOTI = -120740,
ERRHAL_MSPGETPOTI = -120745,
ERRHAL_MSPGETSLNUM = -120750,
ERRHAL_MSPGETVERSION = -120755,
ERRHAL_MSPGETSTAT = -120756,
ERRHAL_MSPGETLIMITEDSENSORSUPPLY = -120757,
ERRHAL_MSPGETERR = -120760,
ERRHAL_MSPDOOWRST = -120761,
ERRHAL_MSPSETOWMODE = -120762,
ERRHAL_TEDSCOM_NOT_POSSIBLE = -120763,
ERRHAL_MSPGETFOWNUM = -120765,
ERRHAL_MSPGETOWDAT = -120775,
ERRHAL_MSPSETOWDAT = -120780,
ERRHAL_MSPBEEP = -120796,
ERRHAL_MSPPRGBLKV = -120800,
ERRHAL_MSPPRGBLK = -120805,
ERRHAL_MSPRDBLK = -120810,
ERRHAL_MSPPRERDBLK = -120811,
ERRHAL_MSPRDRFID = -120815,
ERRHAL_MSPWRRFID = -120816,
ERRHAL_MSPSETLED = -120817,
ERRHAL_MSPRD_NO_RFID = -120818,
ERRHAL_MSPRD_ERR_RFID = -120819,
ERRHAL_MSP_EEPROM_SIZE = -120820,
ERRHAL_MSPWR_EEPROM = -120821,
ERRHAL_MSPRD_EEPROM = -120822,
ERRHAL_MSP_SET_ADC_LPM = -120823,
ERRHAL_MSP_SET_VARIANT = -120824,
ERRHAL_SPISETDACPLL = -120900,
ERRHAL_SPIOPENDRV = -120905,
ERRHAL_SPISETSS = -120910,
ERRHAL_MAX11245_REG_ACCESS = -120950,
ERRHAL_MAX11245_CMD_REG_BUSY = -120951,
ERRHAL_MAX11245_UNABLE_TO_POWER_DOWN = -120952,
ERR_ADJUST_VALUE_OUT_OF_RANGE = -130490,
ERR_ADJUSTMENT_TIMEOUT = -130491,
```

```
    ERR_ADJUSTMENT_CANCELLED = -130492,
    ERR_ADJUST_ZERO_VALUE_OUT_OF_RANGE = -130493,
    ERR_ADJUST_CAL_VALUE_OUT_OF_RANGE = -130494,
    ERR_HAL_CALCAL = -130500,
    ERR_HAL_CALMEAS = -130505,
    ERR_HAL_CORRECTPHASE = -130510,
    ERR_HAL_CALC_PT_COEFFS = -130513,
    ERR_HAL_ACTIVATESET = -130515,
    WARN_HAL_ACTIVATESET = 130515,
    WARN_SET_ZERO_INCL_ANALOGOUT = 130517,
    WARN_SET_ZERO = 130518,
    WARN_SIG_NOT_AVAILABLE = 130519,
    WARN_ADJUST_RMS_PERIOD = 130522,
    ERRHAL_LOOP = -130524,
    ERRHAL_CORRTEMPERATURE = -130526,
    ERRHAL_COLDJUNCTION = -130528,
    WARN_COLDJUNCTION_INITJUST_MISSING = 130529,
    ERRHAL_AUTOADJUST = -130530,
    ERRHAL_EXTERNAL_ADAPTOR = -130532,
    ERRHAL_WRONG_CHANTYPE = -130560,
    ERRHAL_NOT_SUPPORTED = -130600,
    ERR_CONNECTOR_NOT_ENABLED = -130610,
    WARN_6_WIRING_POSSIBLE = 130620,
    ERR_WIRING_2_BROKEN = -130622,
    ERR_WIRING_3_BROKEN = -130623,
    ERRHAL_INTERPOLENABLE = -130700,
    ERRHAL_INTERPOLINIT = -130701,
    ERRHAL_INTERPOLSETDATARATE = -130703,
    ERRHAL_INTERPOLSETSKALINGOVERFLOW = -130704,
    ERRHAL_INTERPOLSETSCALING = -130705,
    ERRHAL_SCALDACNULL = -130706,
    ERRHAL_SCALDACGAIN = -130707,
    ERRHAL_SCALDACNULL_MX410 = -130708,
    WARNHAL_SCALDACNULL_MX410 = 130708,
    ERRHAL_SCALDACGAIN_MX410 = -130709,
    WARNHAL_SCALDACGAIN_MX410 = 130709,
    WARN_CONFIGURE_FIREWIRE_FUNCTIONBLOCK = 132000,
    ERR_CREATE_FIREWIRE_FUNCTIONBLOCK = -132001,
    ERR_RT_IOCTL = -140020,
    ERR_RT_COPY_TO_USER = -140024,
    ERR_RT_COPY_FROM_USER = -140025,
    ERR_TEDS_PROPERTY_NOT_FOUND = -160002,
    ERR_TEDS_DATA_CORRUPT = -160010,
    ERR_TEDS_NO_DISCSIGCONFIG = -160020,
    ERR_TEDS_NO_ELECSIGTYPE = -160021,
    ERR_TEDS_NO_MAPMETH = -160022,
    ERR_TEDS_CALCURVE_DOMAIN_INVALID = -160023,
    ERR_TEDS_CALTABLE_DOMAIN_INVALID = -160024,
    ERR_TYPE_NOT_SUPPORTED = -165000,
    ERR_SELECTION_NOT_SUPPORTED = -165001,
    ERR_MATH_TYPE_NOT_SUPPORTED = -165002,
    ERR_ONLY_SINGLE_CONNECTOR_POSSIBLE = -165003,
    ERR_CAN_DECODER_MODESTARTBIT_511 = -178000,
    ERR_CAN_DECODER_STARTBIT_511 = -178001,
    ERR_CAN_MODE_NOT_SUPPORTED = -178010,
    ERR_CAN_FDFRAME_ONLY_CLASSICAL_SUPPORTED = -178011,
    WARN_CAN_FDFRAME_ONLY_CLASSICAL_SUPPORTED = 178011,
    ERR_CAN_FDFRAME_ONLY_CLASSICAL_AVAILABLE = -178012,
    WARN_CAN_FDFRAME_ONLY_CLASSICAL_AVAILABLE = 178012,
    ERR_CAN_ONLY_ENDLESS_RETRANSMISSION_SUPPORTED = -178013,
    ERR_CAN_RETRANSMISSION_OUT_OF_RANGE = -178014,
```

```
ERR_CAN_TRANSMIT_BYTE_NOT_VALID_HEX_VALUE = -178020,
ERR_ETHERCAT_PARAMETERS = -179800,
ERR_PROFINET_PARAMETERS = -179850,
WARN_PROFINET_PARAMETERS = 179850,
ERR_PROFINET_PARAM_TIMEOUT = -179851,
ERR_ADDCONN_MODE_NOT_SUPPORTED = -179900,
WARN_ANALOG_OUT_GUARD_TIMEOUT = 179901,
WARN_ANALOG_OUT_ONTIMEOUT_VOLTAGE_MAX_POSSIBLE = 179902,
WARN_ANALOG_OUT_ONTIMEOUT_VOLTAGE_MIN_POSSIBLE = 179903,
WARN_ANALOG_OUT_GUARDTIME_MIN_POSSIBLE = 179904,
WARN_ANALOG_OUT_GUARDTIME_MAX_POSSIBLE = 179905,
ERR_ANALOG_OUT_MODE_NOT_SUPPORTED = -179910,
WARN_ANALOG_OUT = 179911,
WARN_ANALOG_OUT_SCALING_NOT_SUPPORTED_DISABLED = 179914,
ERR_ANALOGOUT_NOT_AVAILABLE_IF_ANALOGIN_POLYNOMIAL_SCALING = -179915,
ERR_DIGITALIO_MODE_NOT_OUTPUT = -179919,
ERR_DIGITALIO_MODE_NOT_SUPPORTED = -179920,
ERR_DIGITALIO_TRIGGER_LEVEL_TOO_LOW = -179921,
ERR_DIGITALIO_TRIGGER_LEVEL_TOO_HIGH = -179922,
WARN_SET_DIGITAL_IO_IS_INPUT = 179930,
WARN_SET_DIGITAL_IO_IS_CONNECTED = 179931,
ERR_MASK_RESULT_LONGER_THAN_64_BITS = -179937,
ERR_MASK_SYNTAX = -179938,
ERR_DELTA_MUST_BE_GREATER_THAN_ZERO = -179939,
ERR_DIVISOR_MUST_BE_GREATER_THAN_ZERO = -179940,
WARN_DIVISOR_SET_TO_ZERO = 179940,
ERR_CAN_LISTEN_ONLY = -179941,
WARN_CAN_TRANSMIT_LISTEN_ONLY = 179943,
WARN_TIMER_SET_TO_NEAREST_POSSIBLE = 179944,
WARN_TIMER_SET_TO_MINIMUM_POSSIBLE = 179945,
ERR_CAN_CCP_DBC_DATA_INVALID = -179950,
ERR_CAN_CCP_ERROR_CREATING_LOGGER = -179951,
ERR_CAN_CCP_NOT_INITIALIZED = -179952,
ERR_MATHTYPE_NOT_SUPPORTED = -185100,
WARN_CHECK_ADDFUNC_PARAMETERS = 185101,
WARN_MATH_DISABLED_SIGNALREFERENCE_OFF = 185102,
WARN_PEAK_DISABLED_SIGNALREFERENCE_OFF = 185103,
WARN_PEAK_DISABLED_ANALOGIN_SCALING_NOT_POSSIBLE = 185104,
ERR_PEAK_NOT_POSSIBLE_ANALOGIN_SCALING = -185105,
ERR_TIME_NEGATIVE_OR_ZERO = -185309,
ERR_TIME_NEGATIVE = -185310,
WARN_MATH_RMS_PERIOD_CHANGED_TO_MIN = 185311,
WARN_MATH_ROTVIBE_OUTPUTFILTER_SWITCHED_OFF = 185317,
WARN_MATH_ROTVIBE_OUPUT_HIGHPASS_ALIGNED = 185318,
WARN_MATH_ROTVIBE_FILTER_SWITCHED_OFF = 185319,
WARN_MATH_ROTVIBE_HIGHPASS_ALIGNED = 185320,
WARN_MATH_SCALINGFACTOR_ZERO = 185322,
ERR_MATH_PULSESPERROUND_LESS_THAN_1 = -185323,
ERR_MATH_INTEGRATOR_NOT_ENABLED = -185324,
ERR_MATH_UNIT_NOT_ENABLED = -185325,
WARN_MATH_UNIT_NOT_ENABLED = 185325,
ERR_MATH_PID_MODE_NOT_SUPPORTED = -185326,
ERR_SEGMENT_COUNT_TOO_SMALL = -185327,
ERR_SEGMENT_COUNT_TOO_BIG = -185328,
ERR_MATH_PARAM_REFUSED = -185396,
ERR_PEAK_PARAM_REFUSED = -185397,
ERR_GATE_PARAM_REFUSED = -185398,
ERR_TESTSIGNAL_MODE_NOT_SUPPORTED = -185400,
ERR_TESTSIGNAL_NOT_SUPPORTED = -185401,
ERR_TESTSIGNAL_AMPLITUDETYPE_NOT_SUPPORTED = -185402,
ERR_TESTSIGNAL_AMPLITUDE_LEVEL_MIN_MAX = -185403,
```

```
    ERR_TESTSIGNAL_AMPLITUDE_PEAKPEAK_DELTA = -185404,
    ERR_TESTSIGNAL_ZERO_DIVISION = -185405,
    ERR_TESTSIGNAL_FREQUENCY = -185407,
    ERR_TESTSIGNAL_WRONG_INDEX = -185410,
    WARN_TESTSIGNAL_DUTYCYCLE_SET_TO_MIN = 185412,
    WARN_TESTSIGNAL_DUTYCYCLE_SET_TO_MAX = 185413,
    WARN_TESTSIGNAL_PHASE_SET_TO_MIN = 185414,
    WARN_TESTSIGNAL_PHASE_SET_TO_MAX = 185415,
    ERR_TESTSIGNAL_VOLTAGE_OUT_OF_RANGE = -185416,
    ERR_TESTSIGNAL_ARBITRATION_SOURCETYPE_NOT_SUPPORTED = -185420,
    ERR_TESTSIGNAL_ARBITRATION_RUNMODE_NOT_SUPPORTED = -185421,
    ERR_TESTSIGNAL_ARBITRATION_REPETITIONRATE_NOT_SUPPORTED = -185422,
    ERR_TESTSIGNAL_ARBITRATION_BURSTMODE_NOT_SUPPORTED = -185423,
    ERR_TESTSIGNAL_NOT_ACTIVE_OR_NOT_CONSTANT = -185430,
    ERR_SIGNAL_GENERATOR_MODE_NOT_SUPPORTED = -185440,
    ERR_SIGNAL_GENERATOR_NOT_ENABLED = -185441,
    ERR_TESTSIGNAL_PARAM_REFUSED = -185499,
    WARN_MODULE_NOT_READY_FOR_FIELDBUS_ACQ = 1515710,
    ERR_ECAT_FUNCTIONBLOCK_SET_PDO_STATUS = -1515708,
    ERR_ECAT_FUNCTIONBLOCK_SET_PDO_INDEX = -1515707,
    ERR_ECAT_FUNCTIONBLOCK_CREATE_PROBLEM = -1515706,
    ERR_ECAT_RTSIGNALE_NOT_VALID = -1515705,
    WARN_ECAT_DEMUX_SIGNALE_NOT_VALID = 1515704,
    WARN_ECAT_MODULE_PARAM_NOT_READY = 1515703,
    ERR_ECAT_MODULE_STATUS_NOT_READABLE = -1515702,
    WARN_ECAT_CORRECTED_FW_SIGNALE_LIST_SIZE = 1515701,
    ERR_ECAT_COUNT_OF_FW_SIGNALS_LIMIT_REACHED = -1515700,
    ERR_NTP_SERVER_NOT_REACHED = -1515600,
    ERR_FW_NO_NEC_CHANNEL = -1018005,
    ERR_FW_NO_NEC_DOWNLOAD = -1018006,
    ERR_FW_NO_MSP_CHANNEL = -1018000,
    WARN_EXCITATIONCALIBRATION_DATA_MISSING = 1017998,
    WARN_CALCALIBRATION_DATA_MISSING = 1017997,
    ERR_MSP_DATA_CORRUPT = -1017996,
    ERR_INITJUST_FROM_MSP = -1017995,
    ERR_NO_VALID_INITJUST = -1017993,
    ERR_STORING_INITJUST = -1017992,
    ERR_GETTING_INITJUST = -1017991,
    ERR_PLLVOL_OUTOFRANGE = -1017990,
    ERR_PLLVOL_CORRUPT = -1017989,
    ERR_PLLVOL_FROM_MSP = -1017988,
    ERR_DATA_RATE_DOMAIN_NOT_SUPPORTED = -1016100,
    ERR_SYS_FILENAME = -1016050,
    ERR_SYS_FILE_TOO_LARGE = -1016049,
    ERR_SYS_BAD_DELIVERY = -1016045,
    ERR_SYS_FRAMEWORK_START = -1016035,
    ERR_SYS_OPEN_CLOSE_WD = -1016030,
    ERR_SYS_INVALID_TIME = -1016025,
    ERR_SYS_TOO_MANY_EVENTS = -1016003,
    ERR_SYS_CREATING_OBJECT = -1016002,
    ERR_SPI_OPEN_DRIVER = -1015050,
    ERR_SPI_WAKEUP_DAC = -1015049,
    ERR_SPI_WRITE_DRIVER_DIGITS = -1015048,
    ERR_SPI_WRITE_DRIVER_COMMAND = -1015047,
    ERR_SPI_VOLT_OUTOF_RANGE = -1015040,
    ERR_SPI_PLLDAC = -1015039,
    ERR_SYNC_OPEN_DRIVER = -1015033,
    ERR_SYNC_ANY_TSD_PROBLEM = -1015032,
    ERR_SYNC_NOT_ALL_MODULES_READY = -1015031,
    ERR_SYNC_TIMESOURCE_DATA_NOT_VALID = -1015030,
    ERR_SYNC_ACTIVATE_ETHERCAT = -1015029,
```

```
ERR_SYNC_ACTIVATE_IRIG = -1015028,
ERR_SYNC_ACTIVATE_PTP = -1015027,
ERR_SYNC_ACTIVATE_NTP = -1015026,
ERR_SYNC_ACTIVATE_AUTO = -1015025,
ERR_PARAM_IP_SUBNETMASK_SYNTAX = -1015024,
ERR_PARAM_IP_ADDRESS_SYNTAX = -1015023,
ERR_PARAM_SYNC_TIMESOURCE_NOT_SUPPORTED = -1015022,
ERR_SYNC_CYCLEDIFF_TOO_LARGE = -1015021,
ERR_SYNC_NOT_IN_SYNC = -1015009,
ERR_ADJ_MEASRANGE_INVALID = -1014994,
ERR_ADJ_MEASRANGE_OVERRUN = -1014993,
ERR_ADJ_WRONG_CONNECTOR = -1014992,
ERR_CONN_WRONG_CONNECTOR = -1014801,
ERR_ANALOGOUT_NOCONN = -1014603,
ERR_ANALOGOUT_SIGOVFL = -1014602,
ERR_ANALOGOUT_OCCF = -1014601,
ERR_ANALOGOUT_OVFL = -1014600,
ERR_SIGNALREFERENCE_DECODERINDEX_NOT_SUPPORTED = -1014515,
ERR_SIGNALREFERENCE_DECODERINDEX_LOCAL_NOT_SUPPORTED = -1014514,
ERR_SIGNALREFERENCE_SIGNALGENERATORINDEX_NOT_SUPPORTED = -1014513,
ERR_SIGNALREFERENCE_SIGNALGENERATORINDEX_LOCAL_NOT_SUPPORTED = -1014512,
ERR_SIGNALREFERENCE_DIGITALIOINDEX_LOCAL_NOT_SUPPORTED = -1014511,
ERR_SIGNALREFERENCE_PEAKINDEX_LOCAL_NOT_SUPPORTED = -1014510,
ERR_SIGNALREFERENCE_MATHINDEX_LOCAL_NOT_SUPPORTED = -1014509,
ERR_SIGNALREFERENCE_ALARMINDEX_LOCAL_NOT_SUPPORTED = -1014508,
ERR_SIGNALREFERENCE_CONNECTOR_LOCAL_NOT_SUPPORTED = -1014507,
ERR_SIGNALREFERENCE_SIGNALINDEX_LOCAL_NOT_SUPPORTED = -1014506,
ERR_SIGNALREFERENCE_CANNOT_CREATE = -1014505,
ERR_SIGNALREFERENCE_UNKNOWN = -1014504,
WARN_SIGNALREFERENCE_UNKNOWN = 1014504,
ERR_PARAM_LEVELS_IDENTICAL = -1014503,
ERR_SIGNALREFERENCE_CONNECTOR_NOT_SUPPORTED = -1014502,
WARN_MAX_ISO_SIGNALINDEX_REACHED = 1014501,
ERR_SIGNALREFERENCE_SIGNALINDEX_NOT_SUPPORTED = -1014500,
ERR_INDEX_NOT_SUPPORTED = -1014499,
ERR_SIGNALREFERENCE_SIGNALTYPE_NOT_SUPPORTED_OR_SYNTAX_ERROR = -1014498,
WARN_SIGNALREFERENCE_EMPTY = 1014497,
ERR_MODULEREFERENCE_SYNTAX = -1014496,
ERR_SIGNALREFERENCE_PEAKINDEX_NOT_SUPPORTED = -1014495,
ERR_SIGNALREFERENCE_MATHINDEX_NOT_SUPPORTED = -1014494,
ERR_SIGNALREFERENCE_ALARMINDEX_NOT_SUPPORTED = -1014493,
ERR_MODULEREFERENCE_NOT_LOCAL = -1014492,
ERR_MODULEREFERENCE_NOT_EXTERNAL = -1014491,
ERR_SIGNALREFERENCE_SYNTAX = -1014490,
ERR_PHYSUNIT_NOT_AVAILABLE = -1014489,
ERR_SIGNALREFERENCE_ASYNCSIGNAL_NOT_ALLOWED_OR_SYNTAX_ERROR = -1014488,
ERR_SIGNALREFERENCE_DIGITALIOINDEX_NOT_SUPPORTED = -1014487,
ERR_IOCTL_DEVICE = -1014414,
ERR_DEVICE_NOT_OPEN = -1014413,
ERR_CLOSE_DEVICE = -1014412,
ERR_DEVICE_ALREADY_OPEN = -1014411,
ERR_OPEN_DEVICE = -1014410,
ERR_PINFO_NULL = -1014406,
ERR_POINTER_NULL = -1014405,
ERR_PARAMSETTINGS_NULL = -1014404,
ERR_PARAM_CALCSYNCSIGNAL = -1014403,
ERR_PPLATFORM_NULL = -1014402,
ERR_PAMP_NULL = -1014401,
ERR_PANALOG_NULL = -1014400,
ERR_TEDS_DATA = -1014374,
ERR_NO_SIGNAL_GENERATOR_UNIT_AVAILABLE = -1014348,
```

```
    ERR_NO_GATE_UNIT_AVAILABLE = -1014347,
    ERR_NO_PEAK_UNIT_AVAILABLE = -1014346,
    ERR_NO_MATH_UNIT_AVAILABLE = -1014345,
    ERR_NO_ALARM_UNIT_AVAILABLE = -1014344,
    ERR_NO_DIGITAL_IO_AVAILABLE = -1014343,
    ERR_NO_CAN_CONNECTORS_AVAILABLE = -1014342,
    ERR_NO_ANALOG_OUT_CONNECTORS_AVALIBALE = -1014341,
    ERR_NO_ANALOG_IN_CONNECTORS_AVAILABLE = -1014340,
    ERR_HANDLING_EXTERNAL_ADAPTOR = -1014334,
    ERR_MODE_NOT_SUPPORTED_BY_SENSORTYPE = -1014333,
    WARN_MODE_NOT_SUPPORTED_BY_SENSORTYPE = 1014333,
    ERR_NOT_AVAILABLE_IF_EXTERNAL_ADAPTOR_PRESENT = -1014332,
    WARN_NOT_AVAILABLE_IF_EXTERNAL_ADAPTOR_PRESENT = 1014332,
    ERR_EXECUTE_CONTROL = -1014330,
    WARN_EXECUTE_CONTROL = 1014330,
    ERR_VALUE_NOT_SUPPORTED = -1014324,
    WARN_VALUE_NOT_SUPPORTED = 1014324,
    ERR_AVAILABLE_ONLY_WITH_EXTERNAL_ADAPTOR = -1014323,
    WARN_AVAILABLE_ONLY_WITH_EXTERNAL_ADAPTOR = 1014323,
    ERR_CONTROL_NOT_SUPPORTED_BY_SENSORTYPE = -1014322,
    ERR_CONTROL_NOT_SUPPORTED_AT_CONNECTOR = -1014321,
    ERR_CONTROL_NOT_SUPPORTED = -1014320,
    WARN_CONTROL_NOT_SUPPORTED = 1014320,
    ERR_MODE_NOT_SUPPORTED = -1014319,
    WARN_MODE_NOT_SUPPORTED = 1014319,
    ERR_KEY_NOT_FOUND = -1014318,
    ERR_NO_SPACE_LEFT = -1014317,
    ERR_ASSEMBLY_SIZE_NOT_SUPPORTED = -1014316,
    WARN_CTRL_PEAKMODE_SIGNALCTRL = 1014310,
    WARN_CTRL_RUNMODE_SIGNALCTRL = 1014309,
    ERR_CTRL_UNABLE_TO_SET_SIGNALGENERATOR = -1014308,
    ERR_CTRL_UNABLE_TO_CONTROL_ALARM_UNIT = -1014307,
    ERR_CTRL_UNABLE_TO_CONTROL_PEAK_UNIT = -1014306,
    ERR_CTRL_UNABLE_TO_SET_ZERO = -1014305,
    ERR_CTRL_UNABLE_TO_CONTROL_MATH_UNIT = -1014304,
    ERR_CTRL_UNABLE_TO_CONTROL_SIGNAL_GENERATOR = -1014303,
    ERR_CTRL_UNSPECIFIED = -1014302,
    ERR_WRONG_SECURITY_CODE = -1014301,
    ERR_CTRL_WRONG_PAR = -1014300,
    ERR_CALPOINT_TYPE_PROTECTED = -1014299,
    ERR_CAN_RAW_RECEIVER_ENABLE_DISABLE = -1014285,
    ERR_CAN_RESOURCE_OVERLOAD = -1014284,
    ERR_TRANSMISSIONTYPE_SOURCECHANGE_NO_SUPPORT_64_BIT = -1014282,
    ERR_CAN_WRONG_INDEX = -1014281,
    ERR_CAN_VALUE_ON_ERROR_TYPE_NOT_SUPPORTED = -1014280,
    ERR_CAN_VALUE_ON_ERROR_TYPE_NOT_SUPPORTED_BY_DATAFORMAT = -1014279,
    ERR_CAN_SET_BITRATE = -1014274,
    WARN_CAN_BITRATE_CHANGED = 1014273,
    WARN_CAN_BITRATE_SAMPLEPOINTRATIO_CHANGED = 1014272,
    ERR_CAN_ANY_DECODER_TIMEOUT = -1014271,
    WARN_CAN_ANY_DECODER_TIMEOUT = 1014271,
    ERR_CAN_ANY_DECODER_LOSS_OF_SIGNAL = -1014270,
    WARN_CAN_ANY_DECODER_LOSS_OF_SIGNAL = 1014270,
    ERR_CAN_MAPPING_MASSAGE_INACTIVE = -1014269,
    ERR_CAN_MAPPING_SOURCES_OVERLAP = -1014268,
    WARN_CAN_MAPPING_NO_SOURCE_DEFINED = 1014266,
    ERR_ONLY_ONE_MAPPINGSOURCE_POSSIBLE = -1014265,
    WARN_SIGNALLENGTH_CHANGED = 1014261,
    ERR_CAN_CCP = -1014259,
    ERR_CAN_WRITE_CANSIGNAL = -1014258,
    ERR_CAN_TRANSMIT_SIGNALREFERENCE_INVALID = -1014257,
```

```
ERR_CAN_TRANSMIT_MODULEREFERENCE_NOT_LOCAL = -1014256,
ERR_CAN_TRANSMIT_MSG = -1014255,
WARN_CAN_TRANSMISSIONTYPE_CHANGED = 1014254,
ERR_CAN_TRANSMISSIONTYPE_NOT_SUPPORTED = -1014253,
ERR_CAN_CONNECTOR_NOT_SUPPORTED = -1014252,
ERR_CAN_TRANSMIT_CHANNEL = -1014250,
ERR_CAN_TRANSMIT_USERDEF_MSG = -1014245,
ERR_CAN_TRANSMIT_BYTEORDER_NOT_SUPPORTED = -1014244,
ERR_CAN_TRANSMIT_TABLE = -1014243,
ERR_CAN_TRANSMIT_BYTEVALUE = -1014242,
ERR_CAN_TRANSMIT_BYTECOUNT = -1014241,
ERR_CAN_TRANSMITTER_USERDEF_MSG_SET_TABLE = -1014240,
ERR_CAN_DECODER_MODELENGTH_64 = -1014238,
ERR_CAN_DECODER_MODESTARTBIT_63 = -1014237,
ERR_CAN_DECODER_LENGTH_64 = -1014236,
ERR_CAN_DECODER_STARTBIT_63 = -1014235,
ERR_CAN_COMBINATION_IDENTIFIER_FRAMEFORMAT = -1014234,
ERR_CAN_CHANNELNO = -1014233,
ERR_CAN_DECODER_MODE_STARTBIT_LENGTH = -1014232,
ERR_CAN_COMBINATION_STARTBIT_LENGTH = -1014231,
ERR_CAN_DECODER_FORMAT_DOUBLE = -1014230,
ERR_CAN_DECODER_FORMAT_FLOAT = -1014229,
ERR_CAN_DECODER_FORMAT_32 = -1014228,
ERR_CAN_DECODER_RBUFNO_INVALID = -1014227,
ERR_CAN_DECODER_RBUFNO_USED_TWICE = -1014226,
ERR_CAN_DECODER_GET_TABLE = -1014225,
ERR_CAN_DECODER_SET_TABLE = -1014224,
ERR_CAN_DECODER_TABLE_FULL = -1014223,
ERR_CAN_DECODER_DISABLE = -1014222,
ERR_CAN_DECODER_ENABLE = -1014221,
ERR_CAN_DECODER_TABLE_NOT_VALID = -1014220,
ERR_CAN_DECODER_PARAM_CAN_SIGNAL = -1014219,
ERR_CAN_NO_RESPONSE = -1014218,
WARN_CAN_RESET_REQUEST = 1014217,
ERR_CAN_TRANSMITTER_OVERRUN = -1014216,
ERR_CAN_BUS_ERROR_TRANSMITTER = -1014215,
ERR_CAN_BUS_WARNING_TRANSMITTER = -1014214,
WARN_CAN_BUS_WARNING_TRANSMITTER = 1014214,
ERR_CAN_RECEIVER_OVERRUN = -1014213,
ERR_CAN_BUS_OFF = -1014212,
ERR_CAN_BUS_ERROR_RECEIVER = -1014211,
ERR_CAN_BUS_WARNING_RECEIVER = -1014210,
WARN_CAN_BUS_WARNING_RECEIVER = 1014210,
ERR_CAN_INIT_CANSIGNAL_PARAMS = -1014209,
ERR_CAN_GET_BITRATE = -1014208,
ERR_CAN_GET_STATE = -1014207,
ERR_CAN_INVALID_HANDLE = -1014206,
ERR_CAN_INDEX = -1014205,
ERR_CAN_IOCTL = -1014204,
ERR_CAN_BITRATE_NOT_SUPPORTED = -1014203,
ERR_CAN_DISABLE = -1014202,
ERR_CAN_ENABLE = -1014201,
ERR_CAN_OPENDRV = -1014200,
ERR_FBFS_ACCESS_LED = -1014191,
ERR_FBFS_ACCESS_CAN = -1014190,
ERR_DAQ_SET_CARRIERFREQ_MODE = -1014143,
ERR_DAQ_SET_FREEZED_FLAG = -1014142,
ERR_DAQ_SET_RTMEASMODE = -1014141,
ERR_DAQ_TRIGTIME = -1014140,
ERR_DAQ_TRIGGERUNKNOWN = -1014139,
ERR_DAQ_ISOSIGNALLIST_VERSION = -1014138,
```

```
    ERR_DAQ_AMP_NULL = -1014137,
    ERR_DAQ_RTDRIVER_NULL = -1014133,
    ERR_DAQ_WRONG_CHAN_TYPE = -1014131,
    WARN_SCALING_ANALOGOUT_GAIN_IF_ANALOGIN_LINTABLE = 1014128,
    ERR_DAQ_GET_DEMOD_INFO = -1014127,
    ERR_DAQ_SET_ERRORBUF = -1014125,
    ERR_DAQ_DRV_SET_PT_COEFFS = -1014124,
    ERR_DAQ_DRV_SET_COLDJUNCTION = -1014123,
    ERR_DAQ_DRV_SET_INITJUST = -1014122,
    ERR_SCALING_POLYNOM_NOT_SUPPORTED_IF_ANALOGOUT_ENABLED = -1014121,
    ERR_SCALING_POLYNOM_TOO_MUCH_COEFFS = -1014120,
    ERR_SCALING_POLYNOM_NOT_ENOUGH_COEFFS = -1014119,
    ERR_SCALING_POLYNOM_TOO_MUCH_SEGMENTS = -1014118,
    ERR_SCALING_LINTABLE_TOO_MUCH_POINTS = -1014117,
    ERR_SCALING_POLYNOM_NOT_ENOUGH_SEGMENTS = -1014116,
    ERR_ACCESS_RTDRIVER = -1014115,
    ERR_SCALING_LINTABLE_NOT_ENOUGH_POINTS = -1014114,
    ERR_DAQ_SCALING_INTERNAL = -1014113,
    ERR_DAQ_SCALETYPE = -1014112,
    ERR_WRONG_CHANNEL_INDEX = -1014111,
    ERR_WRONG_SIG_INDEX = -1014110,
    ERR_WRONG_CONNECTOR = -1014109,
    ERR_DAQ_INITJUST_NOTFOUND = -1014108,
    ERR_DAQ_INITJUST_POINTNR = -1014107,
    ERR_DAQ_INITJUST_WRONGPAR = -1014106,
    ERR_DAQ_SIGID_LEN = -1014105,
    ERR_DAQ_SIGID_UNKNOWN = -1014102,
    ERR_DAQ_FILTER = -1014100,
    ERR_DAQ_SIGNAL_NOT_AVAILABLE = -1014099,
    ERR_DAQ_ADD_FBTYPE = -1014098,
    ERR_DAQ_ADD_FB_INSTANCE = -1014097,
    ERR_DAQ_SET_FB_INPUT = -1014096,
    ERR_DAQ_OUTRATE = -1014095,
    ERR_DAQ_SET_BUFFER_PARS = -1014094,
    ERR_DAQ_SET_ZEROVAL = -1014093,
    ERR_DAQ_FIFO_PARAMS = -1014092,
    ERR_DAQ_FIFOREADER_CAL_PARAMS = -1014091,
    ERR_DAQ_SCALING_PARAMS = -1014090,
    ERR_DAQ_SET_ISO_MODULE_ID = -1014089,
    ERR_DAQ_SET_ISO_TRANSMIT = -1014088,
    ERR_DAQ_GET_ISO_TRANSMIT = -1014087,
    ERR_DAQ_OVERRUN = -1014085,
    ERR_DAQ_UNDEFINED = -1014082,
    ERR_DAQ_FIFO_SYNC = -1014070,
    ERR_DAQ_OVERLOAD = -1014060,
    ERR_DAQ_PROCESS = -1014061,
    ERR_DAQ_OCCF = -1014055,
    ERR_DAQ_LIMSE = -1014050,
    ERR_DAQ_NOSIGNALSUBSCRIBED = -1014030,
    ERR_DAQ_BUFFERSIZE = -1014025,
    ERR_DAQ_AMPOFF = -1014023,
    ERR_DAQ_OPENRTDRV = -1014010,
    ERR_DAQ_SUBSCR_RUNNING = -1014006,
    ERR_DAQ_SUBSCR_UUID = -1014005,
    ERR_DAQ_SUBSCR_STATE = -1014004,
    ERR_DAQ_SUBSCRITION = -1014002,
    ERR_DATA_FORMAT_NOT_SUPPORTED = -1010000,
    ERR_SRV_STAT_COMPOSER = -1004305,
    ERR_MTHD_SYS_NORESTARTFKT = -1004260,
    ERR_MTHD_SYS_WRONGCHANTYPE = -1004242,
    ERR_MTHD_SYS_WRONGCHAN = -1004240,
```

```
ERR_MTHD_SYS_INVALIDFWFILE = -1004208,
ERR_MTHD_SYS_FILENOTFOUND = -1004201,
ERR_OPENFILE = -1004200,
ERR_WRITE_FILE = -1004199,
ERR_READ_FILE = -1004198,
ERR_PARSE_XML_SETTING_REFUSED = -1004160,
ERR_PAR_WRONG_TYPE = -1004153,
WARN_PAR_NODEFSET = 1004148,
ERR_PAR_VIEWID_INUSE = -1004146,
ERR_PAR_READING_VAL = -1004144,
WARN_PAR_EMPTY_VAL = 1004143,
ERR_PAR_VIEWINVALID = -1004142,
ERR_PAR_WRONGXPATHSTATE = -1004140,
ERR_PAR_XPATHINVALID = -1004138,
ERR_PAR_NOROOT = -1004136,
WARN_PAR_PARSING = 1004134,
ERR_PAR_PARSING = -1004135,
ERR_PAR_INVALIDADDR = -1004118,
WARN_PAR_TAGUNKOWN = 1004112,
ERR_PAR_TAGUNKOWN = -1004111,
ERR_PAR_OUTOFRANGE = -1004110,
WARN_PAR_OUTOFRANGE = 1004110,
ERR_PAR_INVALIDATTR = -1004109,
WARN_PAR_READONLY = 1004108,
ERR_PAR_CHOICEINVALID = -1004107,
ERR_PAR_CNTXTLOST = -1004106,
ERR_PAR_CHOICEVALUESYNTAXINVALID = -1004105,
ERR_PAR_RESTOREFAILED = -1004104,
ERR_PAR_XMLINVALID = -1004103,
ERR_TXTELCONTAINER_TOO_SMALL = -1003001,
ERR_PARSE_PROCESSING_DATA = -1002002,
ERR_PARSE_INVALID_DATA = -1002001,
ERR_UTIL_FILE_RESTORE = -1001140,
ERR_UTIL_INVALID_FILEHANDLE = -1001136,
ERR_UTIL_STRCONVERT = -1001132,
ERR_UTIL_NO_PATTERN_MATCH = -1001131,
ERR_EVFLG_TIMEOUT = -1001130,
ERR_EVFLG_PTHREADS = -1001128,
ERR_UTIL_TIMER_INVALID = -1001125,
WARN_UTIL_TIMER_WASRUNNING = 1001025,
ERR_UTIL_LANGUAGE = -1001110,
ERR_COMPOSER_TAG_NOT_DEFINED = -1001002,
ERR_NET_NETWORKADAPTER = -1000680,
ERR_UDP_BINDTODEVICE = -1000658,
ERR_UDP_BIND = -1000656,
ERR_UDP_SETBROADCAST = -1000654,
ERR_UDP_SETREUSEADR = -1000652,
ERR_UDP_OPENSOCK = -1000650,
ERR_TCP_BIND = -1000644,
ERR_TCP_LISTEN = -1000642,
ERR_TCP_CHECKIP = -1000638,
ERR_TCP_GETADR = -1000636,
ERR_TCP_GETPORT = -1000635,
ERR_TCP_SETNODELAY = -1000632,
ERR_TCP_SETNONBLOCK = -1000630,
ERR_TCP_OPENSOCK = -1000628,
ERR_TCP_CONNECT = -1000626,
ERR_TCP_ACCEPT = -1000624,
ERR_TCP_CONNECT_FAILED = -1000620,
ERR_TCP_IPADR_ALREADY_CONNECTED = -1000615,
ERR_TCP_ADR_UNKNOWN = -1000612,
```

```
  ERR_TCP_SOCKET_EINTR = -1000604,
  ERR_TCP_SOCKET_HUP = -1000603,
  ERR_TCP_SOCKET_DESCRIPTOR = -1000602,
  ERR_CON_REC = -1000524,
  ERR_CON_SENDING = -1000522,
  ERR_CON_INVALID_TELTYPE = -1000510,
  ERR_CON_ADRRESINI_FAILED = -1000504,
  ERR_CON_CONHNDLR_UUIDUNKNOWN = -1000500,
  ERR_TRANS_READINGHDR = -1000430,
  ERR_TRANS_NOTFOUND = -1000420,
  ERR_TRANS_CANCELED = -1000419,
  ERR_TRANS_SEND_REQ_TIMEOUT = -1000416,
  ERR_TRANS_ACK_TIMEOUT = -1000417,
  ERR_TRANS_REQ_TIMEOUT = -1000418,
  ERR_TRANS_INVALID_TELFORMAT = -1000414,
  ERR_TRANS_REQUESTOR_WRONGPROTTYPE = -1000404,
  ERR_TRANS_RESPONDER_WRONGPROTTYPE = -1000405,
  ERR_TRANS_NOT_FOR_US = -1000406,
  ERR_NO_MSP_CHKSUM = -1000301,
  ERR_SRV_CONTROL = -1000300,
  ERR_SRV_NODATA = -1000275,
  ERR_SRV_NORESPONSE_YET = -1000271,
  ERR_SRV_REQRMTHD_NOREQTEL = -1000265,
  ERR_SRV_PARAMERROR = -1000255,
  ERR_SRV_MTHDOUT_UNKNOWNMTHD = -1000203,
  ERR_SRV_MTHDOUT_WRONGANSW = -1000202,
  ERR_SRV_MTHDOUT_TOOMUCH = -1000201,
  ERR_SRV_UNKNOWNSRV = -1000107,
  ERR_SRV_UNKNOWNMTHD = -1000106,
  ERR_INDEX_OUT_OF_RANGE = -1000000
};
```

C#

```
public enum FrameworkError {
  ERR_NO_SUCCESS = -1,
  ERR_SUCCESS = 0,
  ERR_NOT_SUPPORTED = -110,
  WARN_UNSPECIFIED = 1000,
  WARN_VALUE_CHANGED = 9,
  INTERNAL_ACTIVATE_PROFINET = 1,
  INTERNAL_ACTIVATE_PROFINET_DONE = 2,
  ERR_FIELDBUS_INIT_DRIVER = -1000,
  ERR_FIELDBUS_ACCESS_OD = -1001,
  ERR_FIELDBUS_READ_ODXML = -1002,
  ERR_FIELDBUS_START_OD = -1005,
  ERR_FIELDBUS_STOP_OD = -1006,
  ERR_FIELDBUS = -1010,
  ERR_FIELDBUS_STATECHANGE = -1106,
  ERR_FIELDBUS_INVALIDTIME = -1107,
  ERR_FIELDBUS_UNAVAILABLPAR = -1109,
  ERR_INVALID_REVISION_STRING = -9000,
  ERR_MODULETYPE_STRING_EMPTY = -9001,
  ERR_MODULELABEL_STRING_EMPTY = -9002,
  ERR_INVALID_MAC_ADDRESS_STRING = -9003,
  ERR_SCALING_NOT_SUPPORTED = -100101,
  WARN_SCALING_CHANGED_TO_STANDARD_FOR_SENSORTYPE = 100102,
  ERR_SCALING_NOT_SUPPORTED_BY_SENSORTYPE = -100103,
  ERR_SCALING_INVALID_TYPE = -100104,
  ERR_SCALING_STRAINGAGE_NOT_SUPPORTED = -100105,
  ERR_RESTORING_FACTORY_SCALING = -100106,
  ERR_SCALING_DIV_BY_ZERO = -100110,
  ERR_FILTERTYPE_NOT_SUPPORTED = -100201,
  ERR_FILTERCHARACT_NOT_SUPPORTED = -100202,
  ERR_FILTERFREQ_NOT_SUPPORTED = -100203,
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_2ND_SIGNAL = -100204,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_2ND_SIGNAL = 100204,
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_CLASSIC = -100205,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_CLASSIC = 100205,
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_MODULE = -100206,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_MODULE = 100206,
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = -100207,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = 100207,
  ERR_IIR_FILTER_NOT_SUPPORTED_BY_MODULE = -100208,
  WARN_IIR_FILTER_NOT_SUPPORTED_BY_MODULE = 100208,
  ERR_IIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = -100209,
  WARN_IIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = 100209,
  WARN_FILTERCHAR_CHANGED_TO_IIR_BESSEL_FIR_NO_HIGHSPEED = 100210,
  WARN_FILTERCHAR_CHANGED_TO_IIR_BUTTERWORTH_FIR_NO_HIGHSPEED = 100211,
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_HIGHSPEED = -100212,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_HIGHSPEED = 100212,
  WARN_FILTER_SWITCHED_OFF_FIR_NO_HIGHSPEED = 100213,
  WARN_FILTER_CHARACT_NOT_SUPPORTED_CHANGED_TO_OFF = 100214,
  WARN_FILTER_CHARACT_NOT_SUPPORTED_BY_2ND_SIGNAL_CHANGED_TO_OFF = 100215,
  WARN_FILTER_CHARACT_NOT_SUPPORTED_BY_HIGHSPEED_CHANGED_TO_OFF = 100216,
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_CLASSIC_CHANGED_TO_OFF = 100217,
  WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_IIR_BESSEL = 100218,
  WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_IIR_BUTTERWORTH = 100219,
  WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_FIR_LINEARPHASE = 100220,
  WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_FIR_BUTTERWORTH = 100221,
  ERR_CONNMODE_NOT_SUPPORTED = -100450,
  ERR_SENSORTYPE_NOT_SUPPORTED = -100506,
  ERR_OUTPUTRATE_VS_FILTER_NOT_SUPPORTED = -100510,
  ERR_OUTPUTRATE_NOT_SUPPORTED = -100511,
```


現在位置: Supported devices -> Hbm.Api.QuantumX Namespace -> Hbm.Api.QuantumX.Enums Namespace

```
ERR_PHYSICAL_UNIT_NOT_NATIVE = -100520,
WARN_PHYSICAL_UNIT_NOT_NATIVE = 100520,
WARN_PHYSICAL_UNIT_CHANGED_TO_NATIVE = 100521,
ERR_VOLTAGERANGE_NOT_SUPP = -100700,
ERR_CURRENTRANGE_NOT_SUPP = -100710,
ERR_WIRING_NOT_SUPPORTED = -100802,
ERR_WIRING_NOT_SUPPORTED_CHECK_HW_REVISION = -100803,
ERR_EXCITFREQ_NOT_SUPPORTED = -100804,
ERR_EXCITAMPL_NOT_SUPPORTED = -100806,
ERR_IMPEDANCE_NOT_SUPPORTED = -100807,
ERR_PT_RESISTANCE_NOT_SUPPORTED = -100850,
ERR_TC_TYPE_NOT_SUPPORTED = -100860,
ERR_RESISTOR_NOT_SUPPORTED = -100904,
ERR_SSI_CODING_NOT_SUPPORTED = -101002,
ERR_PRESSURE_RANGE_NOT_SUPPORTED = -101010,
ERR_PRESSURE_MAX_AVAILABLE = -101011,
ERR_PRESSURE_TYPE_ONLY_ABSOLUTE_SUPPORTED = -101012,
ERR_PRESSURE_TYPE_ONLY_RELATIVE_SUPPORTED = -101013,
ERR_PRESSURE_TYPE_NOT_SUPPORTED = -101014,
ERR_PRESSURE_SENSOR_NOT_AVAILABLE = -101015,
ERR_PRESSURE_MIN_AVAILABLE = -101016,
ERR_FREQUENCYRANGE_NOT_SUPPORTED = -101020,
WARN_REQUESTED_MAX_COUNTER_EXCEEDS_POSSIBLE = 101010,
ERR_SENSORSUPPLY_NOT_SUPPORTED = -101100,
ERR_SET_SENSOR_SUPPLY_NOT_ALLOWED_IF_USED_BY_PARAMETER_SET = -101102,
ERR_SETZERO_INHIBITED = -101110,
ERR_SETZERO_NOT_SUPPORTED = -101111,
ERR_ANALOG_OUT_SETZERO = -101202,
ERR_PARAMSETTINGS_IND_COUNTER = -102017,
ERR_PARAMSETTINGS_IND_FREQUENCY = -102018,
ERR_PARAMSETTINGS_BRIDGE = -102020,
ERR_PARAMSETTINGS_VOLTAGE = -102021,
ERR_PARAMSETTINGS_CURRENT = -102022,
ERR_PARAMSETTINGS_THERMOCOUPLE = -102023,
ERR_PARAMSETTINGS_PT_ELEMENT = -102024,
ERR_PARAMSETTINGS_POTI = -102025,
ERR_PARAMSETTINGS_SSI = -102026,
ERR_PARAMSETTINGS_COUNTER = -102027,
ERR_PARAMSETTINGS_FREQUENCY = -102028,
ERR_PARAMSETTINGS_INDBRIDGE = -102029,
ERR_PARAMSETTINGS_LVDT = -102030,
ERR_PARAMSETTINGS_PRESSURE = -102031,
ERR_PARAMSETTINGS_IEPE = -102032,
ERR_PARAMSETTINGS_PWM = -102033,
ERR_PARAMSETTINGS_PULSEWIDTH = -102034,
ERR_PARAMSETTINGS_PERIOD = -102035,
ERR_PARAMSETTINGS_RESISTOR = -102036,
ERR_IRIG_FORMAT_NOT_SUPPORTED = -102037,
WARN_CONNECTOR_SET_TO_VOLTAGE_BECAUSE_ANOTHER_SET_TO_IRIG = 102038,
WARN_IRIG_SENSOR_CAN_ONLY_BE_ACTIVE_ONCE = 102039,
ERR_PARAMSETTINGS_IND_CRANK_COUNTER = -102040,
ERR_PARAMSETTINGS_IND_CRANK_FREQUENCY = -102041,
ERR_GAP_POSITION_SYNTAX = -102042,
ERR_GAP_POSITION_ZERO_NOT_ALLOWED = -102043,
ERR_GAP_POSITION_REPETITION_NOT_ALLOWED = -102044,
ERR_GAP_POSITION_VALUE_GREATER_THAN_NUMBER_OF_TEETH = -102045,
ERR_INDEX_POSITION_VALUE_GREATER_THAN_NUMBER_OF_TEETH = -102046,
ERR_TOO_MUCH_GAP_POSITIONS_DEFINED = -102047,
ERR_PULSES_PER_ROUND_OUT_OF_RANGE = -102048,
WARN_PULSES_PER_ROUND_OUT_OF_RANGE_SET_TO_ZERO = 102048,
ERR_INDEX_SHIFT_OUT_OF_RANGE = -102049,
```

```
WARN_INDEX_SHIFT_OUT_OF_RANGE_SET_TO_ZERO = 102049,
WARN_INDEX_DIVISOR_OUT_OF_RANGE_SET_TO_MINIMUM = 102050,
WARN_INDEX_DIVISOR_OUT_OF_RANGE_SET_TO_MAXIMUM = 102051,
ERR_PARAMSETTINGS_CRANK_COUNTER = -102052,
ERR_PARAMSETTINGS_CRANK_FREQUENCY = -102053,
ERR_PARAM_ZERO_NOT_VALID = -103000,
ERR_PREVIOUS_SETTING_RESTORED = -104010,
ERR_PARAMETERIZATION_REFUSED = -104011,
ERR_SCALING_LINTABLE_WRONG_NUMBER_OF_POINTS = -104099,
WARN_SCALING_NR_OF_SEGMENTS_CHANGED = 104100,
WARN_SCALING_NR_OF_COEFFS_CHANGED = 104101,
ERR_SPACE_IN_STRING_NOT_ALLOWED = -104103,
ERR_STRING_SIZE_TOO_LONG = -104104,
ERR_SCM_HV_NOT_SUPPORTED_BY_VOLTAGE_RANGE = -104105,
ERR_VOLTAGE_EXCEEDS_MEAS_RANGE = -104106,
ERR_HIGHVOLTAGE_ADAPTOR_NOT_SUPPORTED = -104107,
ERR_VOLTAGE_RANGE_OR_HIGHVOLTAGE_ADAPTOR_NOT_SUPPORTED = -104108,
WARN_EXTERNAL_HIGHVOLTAGE_ADAPTOR_NOT_CONNECTED = 104109,
ERR_EXTERNAL_HIGHVOLTAGE_ADAPTOR_NOT_CONNECTED = -104109,
WARN_EXTERNAL_QUARTER_BRIDGE_ADAPTOR_NOT_CONNECTED = 104110,
ERR_EXTERNAL_QUARTER_BRIDGE_ADAPTOR_NOT_CONNECTED = -104110,
ERR_EXTERNAL_ADAPTOR_COEFFS_NOT_AVAILABLE = -104111,
WARN_OUTPUTUNIT_INCOMPATIBLE_TO_SENSORUNIT = 104112,
WARN_OUTPUTUNIT_DISABLED_BECAUSE_PHYS_UNIT_USER_DEF = 104113,
WARN_OUTPUTUNIT_IS_USER_DEFINED = 104114,
ERR_AUTOCAL_REPTIME_NOT_SUPPORTED = -104200,
ERR_AUTOCAL_NOT_SUPPORTED_SENSOR = -104201,
ERR_AUTOCAL_DISABLED = -104203,
ERR_AUTOCAL_SENSOR_NOT_CONNECTED = -104204,
ERR_AUTOCAL_NOT_SUPPORTED_MODULE = -104205,
ERR_ADJUSTMENT_SENSOR_NOT_SUPPORTED = -104206,
WARN_ADJUSTMENT_SENSOR_NOT_SUPPORTED = 104206,
WARN_ADJUSTMENT_REPTIME_TOO_SHORT = 104207,
WARN_ADJUSTMENT_REPTIME_TOO_LONG = 104208,
WARN_ADJUSTMENT_NOT_DEFAULT = 104209,
ERR_AUTOADJUST_NOT_SUPPORTED_SENSOR = -104211,
ERR_AUTOCAL_CYCLE_IS_RUNNING_REPEAT_LATER = -104212,
ERR_TEDS_NO_SENSOR_TEMPLATE = -105000,
ERR_TEDS_NO_DATA_OR_INVALID = -105001,
ERR_TEDS_NO_MIN_PHYS_VAL = -105002,
ERR_TEDS_NO_MAX_PHYS_VAL = -105003,
ERR_TEDS_NO_MIN_ELEC_VAL = -105004,
ERR_TEDS_NO_MAX_ELEC_VAL = -105005,
ERR_TEDS_NO_MDEF_UCC = -105007,
ERR_TEDS_CALCURVE_COEFFS_INVALID = -105008,
ERR_TEDS_NODATA = -105010,
ERR_TEDS_NOPARSEBIN = -105014,
ERR_TEDS_NOAMP = -105018,
ERR_TEDS_NO_EXCITE_AMPL_NOM = -105020,
ERR_TEDS_NO_EXCITE_AMPL_MIN = -105022,
ERR_TEDS_NO_EXCITE_AMPL_MAX = -105024,
ERR_TEDS_NO_ATTACHED_SERIALNUM = -105025,
ERR_TEDS_NO_SENSOR_IMPED = -105030,
ERR_TEDS_GAUGEFACTOR = -105031,
ERR_TEDS_NO_BRIDGETYPE = -105032,
ERR_TEDS_TC_TYPE = -105034,
ERR_TEDS_PT_TYPE = -105036,
ERR_TEDS_LOWPASSFILTER = -105038,
ERR_TEDS_ZEROSPANSCALING = -105040,
ERR_TEDS_GAINSCALING = -105041,
ERR_TEDS_PARAM = -105050,
```

```
    ERR_TEDS_NOT_SUPPORTED_BY_CONNECTOR = -105051,
    WARN_TEDS_NOT_SUPPORTED_CHANGED_TO_IGNORE = 105052,
    ERR_TEDS_NO_CALTABLE = -105054,
    ERR_TEDS_NO_CALCURVE = -105055,
    ERR_TEDS_CALCURVE_POWER_INVALID = -105056,
    ERR_TEDS_CALTABLE_NOT_ENOUGH_POINTS = -105059,
    ERR_TEDS_CALTABLE_TOO_MUCH_POINTS = -105060,
    ERR_TEDS_BUFFER_TOO_SMALL = -105061,
    ERR_TEDS_WRITE_NOT_ALLOWED = -105062,
    ERR_TEDS_TID_DONT_MATCH = -105063,
    ERR_TEDS_SCALINGTYPE_NOT_SUPPORTED = -105100,
    ERR_TEDS_SCALING_PARAMETER_INVALID = -105101,
    ERR_TEDS_UNIT_CONVERSION_NOT_SUPPORTED = -105102,
    ERR_SPEEDMODE_CHANGING_NOT_POSSIBLE_SWITCH_OFF = -105201,
    ERR_SPEEDMODE_TOO_MANY_CONNECTORS_HIGH = -105202,
    ERR_SPEEDMODE_MIXED_MODE_NOT_POSSIBLE = -105204,
    WARN_SPEEDMODE_CHANGED_TO_HIGH = 105205,
    WARN_HIGHSPEEDMODE_NOT_SUPPORTED = 105206,
    WARN_SCALING_PARAMS_DONT_SUPPORT_HIGHSPEEDMODE = 105207,
    ERR_SPEEDMODE_HIGH_NOT_SUPPORTED = -105208,
    WARN_SENSOR_PARAMS_DONT_SUPPORT_HIGHSPEEDMODE = 105209,
    ERR_CONNECTORSOURCE = -105210,
    ERR_PHYSUNIT_NOT_FOUND = -106000,
    ERR_PHYSUNIT_NOT_RELATED = -106004,
    ERR_PHYSUNIT_SCALING_FACTOR_ZERO = -106006,
    WARN_SUPPLY_VOLTAGE_SWITCHED_OFF = 119000,
    WARN_WIRING_ALTERED = 119001,
    WARN_SUPPLY_VOLTAGE_CHANGED_TO_MAX_POSSIBLE = 119002,
    WARN_EXCITATION_FREQ_CHANGED = 119003,
    WARN_EXCITATION_AMPL_CHANGED = 119004,
    ERR_SUPPLY_VOLTAGE_BELOW_MINIMUM = -119005,
    WARN_SSI_CLK_FREQ_CHANGED = 119005,
    WARN_SSI_RESOLUTION_CHANGED = 119006,
    WARN_FILTER_CUTOFF_FREQ_CHANGED = 119010,
    WARN_FILTERTYPE_CHANGED = 119011,
    WARN_FILTERCHAR_CHANGED = 119012,
    WARN_OUTPUTRATE_CHANGED = 119015,
    ERR_GLITCHFILTER_NOT_SUPPORTED = -119024,
    WARN_GLITCHFILTER_CHANGED = 119025,
    ERRHAL_ERROR = -120000,
    ERRHAL_OPENMSPDRV = -120100,
    ERRHAL_OPENCPLDDRV = -120110,
    ERRHAL_GETAMPTEMP = -120120,
    ERRHAL_GETAMPTEMPCAL = -120121,
    ERRHAL_READ_I2C = -120125,
    ERRHAL_WRITE_I2C = -120126,
    ERRHAL_WRITE_MSP = -120127,
    ERRHAL_ENABLECHN = -120130,
    ERRHAL_SETMEASINPUT = -120134,
    ERRHAL_SETSENSCIRCUIT = -120135,
    ERRHAL_SETCALINPUT = -120140,
    ERRHAL_SETGAINCAL = -120145,
    ERRHAL_SETGAINMEAS = -120150,
    ERRHAL_SET_SENSORSUPPLY = -120155,
    ERRHAL_SETFEEDBACK = -120159,
    ERRHAL_SETEXCIT = -120160,
    ERRHAL_GETSNAMP = -120165,
    WARN_HAL_GETSNAMP = 120166,
    ERRHAL_SETSNAMP = -120170,
    ERRHAL_GETHWRAMP = -120175,
    WARN_HAL_GETHWRAMP = 120176,
```

```
ERRHAL_SETHWRAMP = -120180,
ERRHAL_GETTID = -120185,
ERRHAL_GETTEDSDATHW = -120189,
ERRHAL_GETTEDSDAT = -120190,
ERRHAL_TEDSCHECKSUM = -120191,
ERRHAL_NOTEDS = -120192,
ERRHAL_TEDS_ALLZERO = -120193,
ERRHAL_TEDSCHECKSUMINSERT = -120194,
ERRHAL_SETTEDSDAT = -120195,
ERRHAL_SETTEDSCON = -120196,
ERRHAL_SIZETEDSDAT = -120197,
ERRHAL_SETLEDAMP = -120200,
ERRHAL_SETSYSLEDAMP = -120201,
ERRHAL_WRSETTING = -120205,
WARNHAL_WRSETTING = 120205,
ERRHAL_SETTING_NOT_AVAILABLE = -120206,
ERRHAL_SETADCCLK = -120210,
ERRHAL_GETMAXSRAMP = -120211,
ERRHAL_SET_FM_CONF = -120212,
ERRHAL_SET_SSI_CONF = -120213,
ERRHAL_GETMSPVERAMP = -120215,
ERRHAL_GETMSPSTATAMP = -120216,
ERRHAL_SYNCADC = -120220,
ERRHAL_SETCHNPROP = -120225,
ERRHAL_GETMSPREG = -120230,
ERRHAL_SETMSPREG = -120235,
ERRHAL_SETTSTSIGMEAS = -120240,
ERRHAL_SETTSTSIGCAL = -120245,
ERRHAL_SETSIMUVAL = -120246,
ERRHAL_SETSIMUVALFORMAT = -120248,
ERRHAL_SETSIMUVALTYPE = -120249,
ERRHAL_GETSNAPSHOT = -120250,
ERRHAL_SNAPSHOT_MAXDATA = -120251,
ERRHAL_CALCPHASE = -120252,
ERRHAL_NO_VALID_SNAPSHOTDATA = -120253,
ERRHAL_GETCALDATHW = -120256,
ERRHAL_GETCALDAT = -120257,
ERRHAL_SETCALDAT = -120258,
ERRHAL_SETCSURCAL = -120260,
ERRHAL_GETCSURCAL = -120265,
ERRHAL_LDRCHKSUM = -120266,
ERRHAL_STORESENSSUPPLYOFFS = -120267,
ERRHAL_GETSENSSUPPLYOFFS = -120268,
WARN_HAL_GETCSURCAL = 120269,
ERRHAL_SETMEASVALINVAL = -120270,
WARN_HAL_GETSENSSUPPLYOFFS = 120271,
ERRHAL_SELFPGACHN = -120272,
ERRHAL_REDDATRATE = -120274,
ERRHAL_RATE_NOT_SUP_BY_FPGA = -120276,
ERRHAL_FB_NOT_AVAILABLE = -120277,
ERRHAL_FB_UNKNOWN_PROPERTY = -120278,
ERRHAL_FPGATEST_NOT_SUPPORTED = -120279,
WARN_HAL_GETPOTIWIPERFROMMSP = 120280,
ERRHAL_GETPOTIWIPERFROMMSP = -120281,
ERRHAL_POTIWIPERCOMPARE = -120282,
ERRHAL_SETIEPEHP = -120283,
ERRHAL_FB_ARBSIGGEN_BUFFER_BUSY = -120284,
ERRHAL_GETHWIDENTPF = -120300,
WARN_HAL_GETHWIDENTPF = 120301,
ERRHAL_SETHWIDENTPF = -120305,
ERRHAL_SETADDHWREVPF = -120306,
```

```
ERRHAL_SETADDHWIDENTPF = -120307,
ERRHAL_GETHWRPF = -120310,
ERRHAL_GETADDHWRPF = -120311,
ERRHAL_GETADDHWIDENTPF = -120312,
WARN_HAL_GETHWRPF = 120313,
WARN_HAL_GETADDHWRPF = 120314,
ERRHAL_SETHWRPF = -120315,
WARN_HAL_GETADDHWIDENTPF = 120316,
ERRHAL_GETPLLVOLTAGE = -120317,
WARN_HAL_GETPLLVOLTAGE = 120318,
ERRHAL_SETPLLVOLTAGE = -120319,
ERRHAL_SETMACADDR = -120320,
ERRHAL_GETMACADDR = -120325,
ERRHAL_SETLEDPF = -120330,
ERRHAL_GETPFTEMP = -120335,
ERRHAL_GETMODVOL = -120340,
ERRHAL_GETMODCUR = -120345,
ERRHAL_GETEXTCUR = -120350,
ERRHAL_GETMSPAPPLDATEPF = -120358,
ERRHAL_GETMSPLOADERDATEPF = -120359,
ERRHAL_GETMSPVERPF = -120360,
ERRHAL_GETMSPSTATPF = -120361,
ERRHAL_GETLIMCURR = -120362,
WARN_HAL_GETMSPVERPF = 120363,
ERRHAL_SETTSTSIGGEN = -120365,
ERRHAL_STRTDATSTRM = -120367,
ERRHAL_STOPDATSTRM = -120369,
ERRHAL_IISDUMDAT = -120371,
ERRHAL_ENIIS = -120373,
ERRHAL_RSTIIS = -120375,
ERRHAL_CHKFLHETRY = -120377,
ERRHAL_GETMAXSRPF = -120379,
ERRHAL_OPENSNAPSHOTDRV = -120405,
ERRHAL_CLOSESNAPSHOTDRV = -120406,
ERRHAL_ENCXUPPER = -120408,
ERRHAL_SIGNUMBER = -120410,
ERRHAL_FOSDELAY = -120412,
ERRHAL_SETSCALPARS = -120414,
ERRHAL_COMPENSATION_OFFSET = -120415,
ERRHAL_DACINPUTTYPE = -120416,
ERRHAL_UNKNOWNSYNCCODE = -120417,
ERRHAL_ADDFUNCMATH = -120418,
ERRHAL_UNKNOWN_MATH_TYPE = -120420,
ERRHAL_PEAK_OUTRATE = -120422,
ERRHAL_CX_SYNC_TEST = -120424,
ERRHAL_SETZERO_NO_VALID_MEASVAL = -120426,
ERRHAL_SET_FILTER_FIR = -120428,
ERRHAL_PARAM_IS_ZERO = -120429,
ERRHAL_FIR_TAB_CNT = -120430,
ERRHAL_FIR_TAB_CNT_SUM = -120431,
ERRHAL_SETDEMODFACTORS = -120432,
ERRHAL_SETFIRDEMODFILTER = -120433,
ERRHAL_SETPOLYCOEFFSTOFPGA = -120434,
ERRHAL_FPGAOPENDRV = -120500,
ERRHAL_FPGACLOSEDRV = -120501,
ERRHAL_FPGABUSTST = -120502,
ERRHAL_FPGAREAD = -120503,
ERRHAL_FPGAWRITE = -120504,
ERRHAL_FPGAINITI2C = -120505,
ERRHAL_FPGARSTI2C = -120506,
ERRHAL_FPGARSTHZR = -120507,
```

```
ERRHAL_FPGAMMAP = -120509,
ERRHAL_FPGAMUNMAP = -120510,
ERRHAL_FPGAGETTIME = -120511,
ERRHAL_FPGARESET = -120512,
ERRHAL_FPGAIOCTL = -120513,
ERRHAL_MSPGETTMP = -120700,
ERRHAL_MSPDISOVRCURRPROT = -120701,
ERRHAL_MSPGETALLSWITCHES = -120702,
ERRHAL_MSPSETALLSWITCHES = -120703,
ERRHAL_MSPMEAS = -120705,
ERRHAL_MSPRDCTRREG = -120710,
ERRHAL_MSPWRCTRREG = -120715,
ERRHAL_MSPWRREGBIT = -120720,
ERRHAL_MSPSETSHUNT = -120721,
ERRHAL_MSPSETVARIANT = -120722,
ERRHAL_MSPSETWIRING = -120723,
ERRHAL_MSPWRINFO = -120725,
ERRHAL_MSPRDINFO = -120730,
ERRHAL_MSPSETPOTI = -120740,
ERRHAL_MSPGETPOTI = -120745,
ERRHAL_MSPGETSLNUM = -120750,
ERRHAL_MSPGETVERSION = -120755,
ERRHAL_MSPGETSTAT = -120756,
ERRHAL_MSPGETLIMITEDSENSORSUPPLY = -120757,
ERRHAL_MSPGETERR = -120760,
ERRHAL_MSPDOOWRST = -120761,
ERRHAL_MSPSETOWMODE = -120762,
ERRHAL_TEDSCOM_NOT_POSSIBLE = -120763,
ERRHAL_MSPGETFOWNUM = -120765,
ERRHAL_MSPGETOWDAT = -120775,
ERRHAL_MSPSETOWDAT = -120780,
ERRHAL_MSPBEEP = -120796,
ERRHAL_MSPPRGBLKV = -120800,
ERRHAL_MSPPRGBLK = -120805,
ERRHAL_MSPRDBLK = -120810,
ERRHAL_MSPPRERDBLK = -120811,
ERRHAL_MSPRDRFID = -120815,
ERRHAL_MSPWRRFID = -120816,
ERRHAL_MSPSETLED = -120817,
ERRHAL_MSPRD_NO_RFID = -120818,
ERRHAL_MSPRD_ERR_RFID = -120819,
ERRHAL_MSP_EEPROM_SIZE = -120820,
ERRHAL_MSPWR_EEPROM = -120821,
ERRHAL_MSPRD_EEPROM = -120822,
ERRHAL_MSP_SET_ADC_LPM = -120823,
ERRHAL_MSP_SET_VARIANT = -120824,
ERRHAL_SPISETDACPLL = -120900,
ERRHAL_SPIOPENDRV = -120905,
ERRHAL_SPISETSS = -120910,
ERRHAL_MAX11245_REG_ACCESS = -120950,
ERRHAL_MAX11245_CMD_REG_BUSY = -120951,
ERRHAL_MAX11245_UNABLE_TO_POWER_DOWN = -120952,
ERR_ADJUST_VALUE_OUT_OF_RANGE = -130490,
ERR_ADJUSTMENT_TIMEOUT = -130491,
ERR_ADJUSTMENT_CANCELLED = -130492,
ERR_ADJUST_ZERO_VALUE_OUT_OF_RANGE = -130493,
ERR_ADJUST_CAL_VALUE_OUT_OF_RANGE = -130494,
ERR_HAL_CALCAL = -130500,
ERR_HAL_CALMEAS = -130505,
ERR_HAL_CORRECTPHASE = -130510,
ERR_HAL_CALC_PT_COEFFS = -130513,
```


現在位置: Supported devices -> Hbm.Api.QuantumX Namespace -> Hbm.Api.QuantumX.Enums Namespace

```
    ERR_HAL_ACTIVATESET = -130515,
    WARN_HAL_ACTIVATESET = 130515,
    WARN_SET_ZERO_INCL_ANALOGOUT = 130517,
    WARN_SET_ZERO = 130518,
    WARN_SIG_NOT_AVAILABLE = 130519,
    WARN_ADJUST_RMS_PERIOD = 130522,
    ERRHAL_LOOP = -130524,
    ERRHAL_CORRTEMPERATURE = -130526,
    ERRHAL_COLDJUNCTION = -130528,
    WARN_COLDJUNCTION_INITJUST_MISSING = 130529,
    ERRHAL_AUTOADJUST = -130530,
    ERRHAL_EXTERNAL_ADAPTOR = -130532,
    ERRHAL_WRONG_CHANTYPE = -130560,
    ERRHAL_NOT_SUPPORTED = -130600,
    ERR_CONNECTOR_NOT_ENABLED = -130610,
    WARN_6_WIRING_POSSIBLE = 130620,
    ERR_WIRING_2_BROKEN = -130622,
    ERR_WIRING_3_BROKEN = -130623,
    ERRHAL_INTERPOLENABLE = -130700,
    ERRHAL_INTERPOLINIT = -130701,
    ERRHAL_INTERPOLSETDATARATE = -130703,
    ERRHAL_INTERPOLSETSKALINGOVERFLOW = -130704,
    ERRHAL_INTERPOLSETSCALING = -130705,
    ERRHAL_SCALDACNULL = -130706,
    ERRHAL_SCALDACGAIN = -130707,
    ERRHAL_SCALDACNULL_MX410 = -130708,
    WARNHAL_SCALDACNULL_MX410 = 130708,
    ERRHAL_SCALDACGAIN_MX410 = -130709,
    WARNHAL_SCALDACGAIN_MX410 = 130709,
    WARN_CONFIGURE_FIREWIRE_FUNCTIONBLOCK = 132000,
    ERR_CREATE_FIREWIRE_FUNCTIONBLOCK = -132001,
    ERR_RT_IOCTL = -140020,
    ERR_RT_COPY_TO_USER = -140024,
    ERR_RT_COPY_FROM_USER = -140025,
    ERR_TEDS_PROPERTY_NOT_FOUND = -160002,
    ERR_TEDS_DATA_CORRUPT = -160010,
    ERR_TEDS_NO_DISCSIGCONFIG = -160020,
    ERR_TEDS_NO_ELECSIGTYPE = -160021,
    ERR_TEDS_NO_MAPMETH = -160022,
    ERR_TEDS_CALCURVE_DOMAIN_INVALID = -160023,
    ERR_TEDS_CALTABLE_DOMAIN_INVALID = -160024,
    ERR_TYPE_NOT_SUPPORTED = -165000,
    ERR_SELECTION_NOT_SUPPORTED = -165001,
    ERR_MATH_TYPE_NOT_SUPPORTED = -165002,
    ERR_ONLY_SINGLE_CONNECTOR_POSSIBLE = -165003,
    ERR_CAN_DECODER_MODESTARTBIT_511 = -178000,
    ERR_CAN_DECODER_STARTBIT_511 = -178001,
    ERR_CAN_MODE_NOT_SUPPORTED = -178010,
    ERR_CAN_FDFRAME_ONLY_CLASSICAL_SUPPORTED = -178011,
    WARN_CAN_FDFRAME_ONLY_CLASSICAL_SUPPORTED = 178011,
    ERR_CAN_FDFRAME_ONLY_CLASSICAL_AVAILABLE = -178012,
    WARN_CAN_FDFRAME_ONLY_CLASSICAL_AVAILABLE = 178012,
    ERR_CAN_ONLY_ENDLESS_RETRANSMISSION_SUPPORTED = -178013,
    ERR_CAN_RETRANSMISSION_OUT_OF_RANGE = -178014,
    ERR_CAN_TRANSMIT_BYTE_NOT_VALID_HEX_VALUE = -178020,
    ERR_ETHERCAT_PARAMETERS = -179800,
    ERR_PROFINET_PARAMETERS = -179850,
    WARN_PROFINET_PARAMETERS = 179850,
    ERR_PROFINET_PARAM_TIMEOUT = -179851,
    ERR_ADDCONN_MODE_NOT_SUPPORTED = -179900,
    WARN_ANALOG_OUT_GUARD_TIMEOUT = 179901,
```

```
WARN_ANALOG_OUT_ONTIMEOUT_VOLTAGE_MAX_POSSIBLE = 179902,
WARN_ANALOG_OUT_ONTIMEOUT_VOLTAGE_MIN_POSSIBLE = 179903,
WARN_ANALOG_OUT_GUARDTIME_MIN_POSSIBLE = 179904,
WARN_ANALOG_OUT_GUARDTIME_MAX_POSSIBLE = 179905,
ERR_ANALOG_OUT_MODE_NOT_SUPPORTED = -179910,
WARN_ANALOG_OUT = 179911,
WARN_ANALOG_OUT_SCALING_NOT_SUPPORTED_DISABLED = 179914,
ERR_ANALOGOUT_NOT_AVAILABLE_IF_ANALOGIN_POLYNOMIAL_SCALING = -179915,
ERR_DIGITALIO_MODE_NOT_OUTPUT = -179919,
ERR_DIGITALIO_MODE_NOT_SUPPORTED = -179920,
ERR_DIGITALIO_TRIGGER_LEVEL_TOO_LOW = -179921,
ERR_DIGITALIO_TRIGGER_LEVEL_TOO_HIGH = -179922,
WARN_SET_DIGITAL_IO_IS_INPUT = 179930,
WARN_SET_DIGITAL_IO_IS_CONNECTED = 179931,
ERR_MASK_RESULT_LONGER_THAN_64_BITS = -179937,
ERR_MASK_SYNTAX = -179938,
ERR_DELTA_MUST_BE_GREATER_THAN_ZERO = -179939,
ERR_DIVISOR_MUST_BE_GREATER_THAN_ZERO = -179940,
WARN_DIVISOR_SET_TO_ZERO = 179940,
ERR_CAN_LISTEN_ONLY = -179941,
WARN_CAN_TRANSMIT_LISTEN_ONLY = 179943,
WARN_TIMER_SET_TO_NEAREST_POSSIBLE = 179944,
WARN_TIMER_SET_TO_MINIMUM_POSSIBLE = 179945,
ERR_CAN_CCP_DBC_DATA_INVALID = -179950,
ERR_CAN_CCP_ERROR_CREATING_LOGGER = -179951,
ERR_CAN_CCP_NOT_INITIALIZED = -179952,
ERR_MATHTYPE_NOT_SUPPORTED = -185100,
WARN_CHECK_ADDFUNC_PARAMETERS = 185101,
WARN_MATH_DISABLED_SIGNALREFERENCE_OFF = 185102,
WARN_PEAK_DISABLED_SIGNALREFERENCE_OFF = 185103,
WARN_PEAK_DISABLED_ANALOGIN_SCALING_NOT_POSSIBLE = 185104,
ERR_PEAK_NOT_POSSIBLE_ANALOGIN_SCALING = -185105,
ERR_TIME_NEGATIVE_OR_ZERO = -185309,
ERR_TIME_NEGATIVE = -185310,
WARN_MATH_RMS_PERIOD_CHANGED_TO_MIN = 185311,
WARN_MATH_ROTVIBE_OUTPUTFILTER_SWITCHED_OFF = 185317,
WARN_MATH_ROTVIBE_OUPUT_HIGHPASS_ALIGNED = 185318,
WARN_MATH_ROTVIBE_FILTER_SWITCHED_OFF = 185319,
WARN_MATH_ROTVIBE_HIGHPASS_ALIGNED = 185320,
WARN_MATH_SCALINGFACTOR_ZERO = 185322,
ERR_MATH_PULSESPERROUND_LESS_THAN_1 = -185323,
ERR_MATH_INTEGRATOR_NOT_ENABLED = -185324,
ERR_MATH_UNIT_NOT_ENABLED = -185325,
WARN_MATH_UNIT_NOT_ENABLED = 185325,
ERR_MATH_PID_MODE_NOT_SUPPORTED = -185326,
ERR_SEGMENT_COUNT_TOO_SMALL = -185327,
ERR_SEGMENT_COUNT_TOO_BIG = -185328,
ERR_MATH_PARAM_REFUSED = -185396,
ERR_PEAK_PARAM_REFUSED = -185397,
ERR_GATE_PARAM_REFUSED = -185398,
ERR_TESTSIGNAL_MODE_NOT_SUPPORTED = -185400,
ERR_TESTSIGNAL_NOT_SUPPORTED = -185401,
ERR_TESTSIGNAL_AMPLITUDETYPE_NOT_SUPPORTED = -185402,
ERR_TESTSIGNAL_AMPLITUDE_LEVEL_MIN_MAX = -185403,
ERR_TESTSIGNAL_AMPLITUDE_PEAKPEAK_DELTA = -185404,
ERR_TESTSIGNAL_ZERO_DIVISION = -185405,
ERR_TESTSIGNAL_FREQUENCY = -185407,
ERR_TESTSIGNAL_WRONG_INDEX = -185410,
WARN_TESTSIGNAL_DUTYCYCLE_SET_TO_MIN = 185412,
WARN_TESTSIGNAL_DUTYCYCLE_SET_TO_MAX = 185413,
WARN_TESTSIGNAL_PHASE_SET_TO_MIN = 185414,
```

```
WARN_TESTSIGNAL_PHASE_SET_TO_MAX = 185415,
ERR_TESTSIGNAL_VOLTAGE_OUT_OF_RANGE = -185416,
ERR_TESTSIGNAL_ARBITRATION_SOURCETYPE_NOT_SUPPORTED = -185420,
ERR_TESTSIGNAL_ARBITRATION_RUNMODE_NOT_SUPPORTED = -185421,
ERR_TESTSIGNAL_ARBITRATION_REPETITIONRATE_NOT_SUPPORTED = -185422,
ERR_TESTSIGNAL_ARBITRATION_BURSTMODE_NOT_SUPPORTED = -185423,
ERR_TESTSIGNAL_NOT_ACTIVE_OR_NOT_CONSTANT = -185430,
ERR_SIGNAL_GENERATOR_MODE_NOT_SUPPORTED = -185440,
ERR_SIGNAL_GENERATOR_NOT_ENABLED = -185441,
ERR_TESTSIGNAL_PARAM_REFUSED = -185499,
WARN_MODULE_NOT_READY_FOR_FIELDBUS_ACQ = 1515710,
ERR_ECAT_FUNCTIONBLOCK_SET_PDO_STATUS = -1515708,
ERR_ECAT_FUNCTIONBLOCK_SET_PDO_INDEX = -1515707,
ERR_ECAT_FUNCTIONBLOCK_CREATE_PROBLEM = -1515706,
ERR_ECAT_RTSIGNALE_NOT_VALID = -1515705,
WARN_ECAT_DEMUX_SIGNALE_NOT_VALID = 1515704,
WARN_ECAT_MODULE_PARAM_NOT_READY = 1515703,
ERR_ECAT_MODULE_STATUS_NOT_READABLE = -1515702,
WARN_ECAT_CORRECTED_FW_SIGNALE_LIST_SIZE = 1515701,
ERR_ECAT_COUNT_OF_FW_SIGNALS_LIMIT_REACHED = -1515700,
ERR_NTP_SERVER_NOT_REACHED = -1515600,
ERR_FW_NO_NEC_CHANNEL = -1018005,
ERR_FW_NO_NEC_DOWNLOAD = -1018006,
ERR_FW_NO_MSP_CHANNEL = -1018000,
WARN_EXCITATIONCALIBRATION_DATA_MISSING = 1017998,
WARN_CALCALIBRATION_DATA_MISSING = 1017997,
ERR_MSP_DATA_CORRUPT = -1017996,
ERR_INITJUST_FROM_MSP = -1017995,
ERR_NO_VALID_INITJUST = -1017993,
ERR_STORING_INITJUST = -1017992,
ERR_GETTING_INITJUST = -1017991,
ERR_PLLVOL_OUTOFRANGE = -1017990,
ERR_PLLVOL_CORRUPT = -1017989,
ERR_PLLVOL_FROM_MSP = -1017988,
ERR_DATA_RATE_DOMAIN_NOT_SUPPORTED = -1016100,
ERR_SYS_FILENAME = -1016050,
ERR_SYS_FILE_TOO_LARGE = -1016049,
ERR_SYS_BAD_DELIVERY = -1016045,
ERR_SYS_FRAMEWORK_START = -1016035,
ERR_SYS_OPEN_CLOSE_WD = -1016030,
ERR_SYS_INVALID_TIME = -1016025,
ERR_SYS_TOO_MANY_EVENTS = -1016003,
ERR_SYS_CREATING_OBJECT = -1016002,
ERR_SPI_OPEN_DRIVER = -1015050,
ERR_SPI_WAKEUP_DAC = -1015049,
ERR_SPI_WRITE_DRIVER_DIGITS = -1015048,
ERR_SPI_WRITE_DRIVER_COMMAND = -1015047,
ERR_SPI_VOLT_OUTOF_RANGE = -1015040,
ERR_SPI_PLLDAC = -1015039,
ERR_SYNC_OPEN_DRIVER = -1015033,
ERR_SYNC_ANY_TSD_PROBLEM = -1015032,
ERR_SYNC_NOT_ALL_MODULES_READY = -1015031,
ERR_SYNC_TIMESOURCE_DATA_NOT_VALID = -1015030,
ERR_SYNC_ACTIVATE_ETHERCAT = -1015029,
ERR_SYNC_ACTIVATE_IRIG = -1015028,
ERR_SYNC_ACTIVATE_PTP = -1015027,
ERR_SYNC_ACTIVATE_NTP = -1015026,
ERR_SYNC_ACTIVATE_AUTO = -1015025,
ERR_PARAM_IP_SUBNETMASK_SYNTAX = -1015024,
ERR_PARAM_IP_ADDRESS_SYNTAX = -1015023,
ERR_PARAM_SYNC_TIMESOURCE_NOT_SUPPORTED = -1015022,
```

```
ERR_SYNC_CYCLEDIFF_TOO_LARGE = -1015021,
ERR_SYNC_NOT_IN_SYNC = -1015009,
ERR_ADJ_MEASRANGE_INVALID = -1014994,
ERR_ADJ_MEASRANGE_OVERRUN = -1014993,
ERR_ADJ_WRONG_CONNECTOR = -1014992,
ERR_CONN_WRONG_CONNECTOR = -1014801,
ERR_ANALOGOUT_NOCONN = -1014603,
ERR_ANALOGOUT_SIGOVFL = -1014602,
ERR_ANALOGOUT_OCCF = -1014601,
ERR_ANALOGOUT_OVFL = -1014600,
ERR_SIGNALREFERENCE_DECODERINDEX_NOT_SUPPORTED = -1014515,
ERR_SIGNALREFERENCE_DECODERINDEX_LOCAL_NOT_SUPPORTED = -1014514,
ERR_SIGNALREFERENCE_SIGNALGENERATORINDEX_NOT_SUPPORTED = -1014513,
ERR_SIGNALREFERENCE_SIGNALGENERATORINDEX_LOCAL_NOT_SUPPORTED = -1014512,
ERR_SIGNALREFERENCE_DIGITALIOINDEX_LOCAL_NOT_SUPPORTED = -1014511,
ERR_SIGNALREFERENCE_PEAKINDEX_LOCAL_NOT_SUPPORTED = -1014510,
ERR_SIGNALREFERENCE_MATHINDEX_LOCAL_NOT_SUPPORTED = -1014509,
ERR_SIGNALREFERENCE_ALARMINDEX_LOCAL_NOT_SUPPORTED = -1014508,
ERR_SIGNALREFERENCE_CONNECTOR_LOCAL_NOT_SUPPORTED = -1014507,
ERR_SIGNALREFERENCE_SIGNALINDEX_LOCAL_NOT_SUPPORTED = -1014506,
ERR_SIGNALREFERENCE_CANNOT_CREATE = -1014505,
ERR_SIGNALREFERENCE_UNKNOWN = -1014504,
WARN_SIGNALREFERENCE_UNKNOWN = 1014504,
ERR_PARAM_LEVELS_IDENTICAL = -1014503,
ERR_SIGNALREFERENCE_CONNECTOR_NOT_SUPPORTED = -1014502,
WARN_MAX_ISO_SIGNALINDEX_REACHED = 1014501,
ERR_SIGNALREFERENCE_SIGNALINDEX_NOT_SUPPORTED = -1014500,
ERR_INDEX_NOT_SUPPORTED = -1014499,
ERR_SIGNALREFERENCE_SIGNALTYPE_NOT_SUPPORTED_OR_SYNTAX_ERROR = -1014498,
WARN_SIGNALREFERENCE_EMPTY = 1014497,
ERR_MODULEREFERENCE_SYNTAX = -1014496,
ERR_SIGNALREFERENCE_PEAKINDEX_NOT_SUPPORTED = -1014495,
ERR_SIGNALREFERENCE_MATHINDEX_NOT_SUPPORTED = -1014494,
ERR_SIGNALREFERENCE_ALARMINDEX_NOT_SUPPORTED = -1014493,
ERR_MODULEREFERENCE_NOT_LOCAL = -1014492,
ERR_MODULEREFERENCE_NOT_EXTERNAL = -1014491,
ERR_SIGNALREFERENCE_SYNTAX = -1014490,
ERR_PHYSUNIT_NOT_AVAILABLE = -1014489,
ERR_SIGNALREFERENCE_ASYNCSIGNAL_NOT_ALLOWED_OR_SYNTAX_ERROR = -1014488,
ERR_SIGNALREFERENCE_DIGITALIOINDEX_NOT_SUPPORTED = -1014487,
ERR_IOCTL_DEVICE = -1014414,
ERR_DEVICE_NOT_OPEN = -1014413,
ERR_CLOSE_DEVICE = -1014412,
ERR_DEVICE_ALREADY_OPEN = -1014411,
ERR_OPEN_DEVICE = -1014410,
ERR_PINFO_NULL = -1014406,
ERR_POINTER_NULL = -1014405,
ERR_PARAMSETTINGS_NULL = -1014404,
ERR_PARAM_CALCSYNCSIGNAL = -1014403,
ERR_PPLATFORM_NULL = -1014402,
ERR_PAMP_NULL = -1014401,
ERR_PANALOG_NULL = -1014400,
ERR_TEDS_DATA = -1014374,
ERR_NO_SIGNAL_GENERATOR_UNIT_AVAILABLE = -1014348,
ERR_NO_GATE_UNIT_AVAILABLE = -1014347,
ERR_NO_PEAK_UNIT_AVAILABLE = -1014346,
ERR_NO_MATH_UNIT_AVAILABLE = -1014345,
ERR_NO_ALARM_UNIT_AVAILABLE = -1014344,
ERR_NO_DIGITAL_IO_AVAILABLE = -1014343,
ERR_NO_CAN_CONNECTORS_AVAILABLE = -1014342,
ERR_NO_ANALOG_OUT_CONNECTORS_AVALIBALE = -1014341,
```

```
    ERR_NO_ANALOG_IN_CONNECTORS_AVAILABLE = -1014340,
    ERR_HANDLING_EXTERNAL_ADAPTOR = -1014334,
    ERR_MODE_NOT_SUPPORTED_BY_SENSORTYPE = -1014333,
    WARN_MODE_NOT_SUPPORTED_BY_SENSORTYPE = 1014333,
    ERR_NOT_AVAILABLE_IF_EXTERNAL_ADAPTOR_PRESENT = -1014332,
    WARN_NOT_AVAILABLE_IF_EXTERNAL_ADAPTOR_PRESENT = 1014332,
    ERR_EXECUTE_CONTROL = -1014330,
    WARN_EXECUTE_CONTROL = 1014330,
    ERR_VALUE_NOT_SUPPORTED = -1014324,
    WARN_VALUE_NOT_SUPPORTED = 1014324,
    ERR_AVAILABLE_ONLY_WITH_EXTERNAL_ADAPTOR = -1014323,
    WARN_AVAILABLE_ONLY_WITH_EXTERNAL_ADAPTOR = 1014323,
    ERR_CONTROL_NOT_SUPPORTED_BY_SENSORTYPE = -1014322,
    ERR_CONTROL_NOT_SUPPORTED_AT_CONNECTOR = -1014321,
    ERR_CONTROL_NOT_SUPPORTED = -1014320,
    WARN_CONTROL_NOT_SUPPORTED = 1014320,
    ERR_MODE_NOT_SUPPORTED = -1014319,
    WARN_MODE_NOT_SUPPORTED = 1014319,
    ERR_KEY_NOT_FOUND = -1014318,
    ERR_NO_SPACE_LEFT = -1014317,
    ERR_ASSEMBLY_SIZE_NOT_SUPPORTED = -1014316,
    WARN_CTRL_PEAKMODE_SIGNALCTRL = 1014310,
    WARN_CTRL_RUNMODE_SIGNALCTRL = 1014309,
    ERR_CTRL_UNABLE_TO_SET_SIGNALGENERATOR = -1014308,
    ERR_CTRL_UNABLE_TO_CONTROL_ALARM_UNIT = -1014307,
    ERR_CTRL_UNABLE_TO_CONTROL_PEAK_UNIT = -1014306,
    ERR_CTRL_UNABLE_TO_SET_ZERO = -1014305,
    ERR_CTRL_UNABLE_TO_CONTROL_MATH_UNIT = -1014304,
    ERR_CTRL_UNABLE_TO_CONTROL_SIGNAL_GENERATOR = -1014303,
    ERR_CTRL_UNSPECIFIED = -1014302,
    ERR_WRONG_SECURITY_CODE = -1014301,
    ERR_CTRL_WRONG_PAR = -1014300,
    ERR_CALPOINT_TYPE_PROTECTED = -1014299,
    ERR_CAN_RAW_RECEIVER_ENABLE_DISABLE = -1014285,
    ERR_CAN_RESOURCE_OVERLOAD = -1014284,
    ERR_TRANSMISSIONTYPE_SOURCECHANGE_NO_SUPPORT_64_BIT = -1014282,
    ERR_CAN_WRONG_INDEX = -1014281,
    ERR_CAN_VALUE_ON_ERROR_TYPE_NOT_SUPPORTED = -1014280,
    ERR_CAN_VALUE_ON_ERROR_TYPE_NOT_SUPPORTED_BY_DATAFORMAT = -1014279,
    ERR_CAN_SET_BITRATE = -1014274,
    WARN_CAN_BITRATE_CHANGED = 1014273,
    WARN_CAN_BITRATE_SAMPLEPOINTRATIO_CHANGED = 1014272,
    ERR_CAN_ANY_DECODER_TIMEOUT = -1014271,
    WARN_CAN_ANY_DECODER_TIMEOUT = 1014271,
    ERR_CAN_ANY_DECODER_LOSS_OF_SIGNAL = -1014270,
    WARN_CAN_ANY_DECODER_LOSS_OF_SIGNAL = 1014270,
    ERR_CAN_MAPPING_MASSAGE_INACTIVE = -1014269,
    ERR_CAN_MAPPING_SOURCES_OVERLAP = -1014268,
    WARN_CAN_MAPPING_NO_SOURCE_DEFINED = 1014266,
    ERR_ONLY_ONE_MAPPINGSOURCE_POSSIBLE = -1014265,
    WARN_SIGNALLENGTH_CHANGED = 1014261,
    ERR_CAN_CCP = -1014259,
    ERR_CAN_WRITE_CANSIGNAL = -1014258,
    ERR_CAN_TRANSMIT_SIGNALREFERENCE_INVALID = -1014257,
    ERR_CAN_TRANSMIT_MODULEREFERENCE_NOT_LOCAL = -1014256,
    ERR_CAN_TRANSMIT_MSG = -1014255,
    WARN_CAN_TRANSMISSIONTYPE_CHANGED = 1014254,
    ERR_CAN_TRANSMISSIONTYPE_NOT_SUPPORTED = -1014253,
    ERR_CAN_CONNECTOR_NOT_SUPPORTED = -1014252,
    ERR_CAN_TRANSMIT_CHANNEL = -1014250,
    ERR_CAN_TRANSMIT_USERDEF_MSG = -1014245,
```

```
ERR_CAN_TRANSMIT_BYTEORDER_NOT_SUPPORTED = -1014244,
ERR_CAN_TRANSMIT_TABLE = -1014243,
ERR_CAN_TRANSMIT_BYTEVALUE = -1014242,
ERR_CAN_TRANSMIT_BYTECOUNT = -1014241,
ERR_CAN_TRANSMITTER_USERDEF_MSG_SET_TABLE = -1014240,
ERR_CAN_DECODER_MODELENGTH_64 = -1014238,
ERR_CAN_DECODER_MODESTARTBIT_63 = -1014237,
ERR_CAN_DECODER_LENGTH_64 = -1014236,
ERR_CAN_DECODER_STARTBIT_63 = -1014235,
ERR_CAN_COMBINATION_IDENTIFIER_FRAMEFORMAT = -1014234,
ERR_CAN_CHANNELNO = -1014233,
ERR_CAN_DECODER_MODE_STARTBIT_LENGTH = -1014232,
ERR_CAN_COMBINATION_STARTBIT_LENGTH = -1014231,
ERR_CAN_DECODER_FORMAT_DOUBLE = -1014230,
ERR_CAN_DECODER_FORMAT_FLOAT = -1014229,
ERR_CAN_DECODER_FORMAT_32 = -1014228,
ERR_CAN_DECODER_RBUFNO_INVALID = -1014227,
ERR_CAN_DECODER_RBUFNO_USED_TWICE = -1014226,
ERR_CAN_DECODER_GET_TABLE = -1014225,
ERR_CAN_DECODER_SET_TABLE = -1014224,
ERR_CAN_DECODER_TABLE_FULL = -1014223,
ERR_CAN_DECODER_DISABLE = -1014222,
ERR_CAN_DECODER_ENABLE = -1014221,
ERR_CAN_DECODER_TABLE_NOT_VALID = -1014220,
ERR_CAN_DECODER_PARAM_CAN_SIGNAL = -1014219,
ERR_CAN_NO_RESPONSE = -1014218,
WARN_CAN_RESET_REQUEST = 1014217,
ERR_CAN_TRANSMITTER_OVERRUN = -1014216,
ERR_CAN_BUS_ERROR_TRANSMITTER = -1014215,
ERR_CAN_BUS_WARNING_TRANSMITTER = -1014214,
WARN_CAN_BUS_WARNING_TRANSMITTER = 1014214,
ERR_CAN_RECEIVER_OVERRUN = -1014213,
ERR_CAN_BUS_OFF = -1014212,
ERR_CAN_BUS_ERROR_RECEIVER = -1014211,
ERR_CAN_BUS_WARNING_RECEIVER = -1014210,
WARN_CAN_BUS_WARNING_RECEIVER = 1014210,
ERR_CAN_INIT_CANSIGNAL_PARAMS = -1014209,
ERR_CAN_GET_BITRATE = -1014208,
ERR_CAN_GET_STATE = -1014207,
ERR_CAN_INVALID_HANDLE = -1014206,
ERR_CAN_INDEX = -1014205,
ERR_CAN_IOCTL = -1014204,
ERR_CAN_BITRATE_NOT_SUPPORTED = -1014203,
ERR_CAN_DISABLE = -1014202,
ERR_CAN_ENABLE = -1014201,
ERR_CAN_OPENDRV = -1014200,
ERR_FBFS_ACCESS_LED = -1014191,
ERR_FBFS_ACCESS_CAN = -1014190,
ERR_DAQ_SET_CARRIERFREQ_MODE = -1014143,
ERR_DAQ_SET_FREEZED_FLAG = -1014142,
ERR_DAQ_SET_RTMEASMODE = -1014141,
ERR_DAQ_TRIGTIME = -1014140,
ERR_DAQ_TRIGGERUNKNOWN = -1014139,
ERR_DAQ_ISOSIGNALLIST_VERSION = -1014138,
ERR_DAQ_AMP_NULL = -1014137,
ERR_DAQ_RTDRIVER_NULL = -1014133,
ERR_DAQ_WRONG_CHAN_TYPE = -1014131,
WARN_SCALING_ANALOGOUT_GAIN_IF_ANALOGIN_LINTABLE = 1014128,
ERR_DAQ_GET_DEMOD_INFO = -1014127,
ERR_DAQ_SET_ERRORBUF = -1014125,
ERR_DAQ_DRV_SET_PT_COEFFS = -1014124,
```

```
    ERR_DAQ_DRV_SET_COLDJUNCTION = -1014123,
    ERR_DAQ_DRV_SET_INITJUST = -1014122,
    ERR_SCALING_POLYNOM_NOT_SUPPORTED_IF_ANALOGOUT_ENABLED = -1014121,
    ERR_SCALING_POLYNOM_TOO_MUCH_COEFFS = -1014120,
    ERR_SCALING_POLYNOM_NOT_ENOUGH_COEFFS = -1014119,
    ERR_SCALING_POLYNOM_TOO_MUCH_SEGMENTS = -1014118,
    ERR_SCALING_LINTABLE_TOO_MUCH_POINTS = -1014117,
    ERR_SCALING_POLYNOM_NOT_ENOUGH_SEGMENTS = -1014116,
    ERR_ACCESS_RTDRIVER = -1014115,
    ERR_SCALING_LINTABLE_NOT_ENOUGH_POINTS = -1014114,
    ERR_DAQ_SCALING_INTERNAL = -1014113,
    ERR_DAQ_SCALETYPE = -1014112,
    ERR_WRONG_CHANNEL_INDEX = -1014111,
    ERR_WRONG_SIG_INDEX = -1014110,
    ERR_WRONG_CONNECTOR = -1014109,
    ERR_DAQ_INITJUST_NOTFOUND = -1014108,
    ERR_DAQ_INITJUST_POINTNR = -1014107,
    ERR_DAQ_INITJUST_WRONGPAR = -1014106,
    ERR_DAQ_SIGID_LEN = -1014105,
    ERR_DAQ_SIGID_UNKNOWN = -1014102,
    ERR_DAQ_FILTER = -1014100,
    ERR_DAQ_SIGNAL_NOT_AVAILABLE = -1014099,
    ERR_DAQ_ADD_FBTYPE = -1014098,
    ERR_DAQ_ADD_FB_INSTANCE = -1014097,
    ERR_DAQ_SET_FB_INPUT = -1014096,
    ERR_DAQ_OUTRATE = -1014095,
    ERR_DAQ_SET_BUFFER_PARS = -1014094,
    ERR_DAQ_SET_ZEROVAL = -1014093,
    ERR_DAQ_FIFO_PARAMS = -1014092,
    ERR_DAQ_FIFOREADER_CAL_PARAMS = -1014091,
    ERR_DAQ_SCALING_PARAMS = -1014090,
    ERR_DAQ_SET_ISO_MODULE_ID = -1014089,
    ERR_DAQ_SET_ISO_TRANSMIT = -1014088,
    ERR_DAQ_GET_ISO_TRANSMIT = -1014087,
    ERR_DAQ_OVERRUN = -1014085,
    ERR_DAQ_UNDEFINED = -1014082,
    ERR_DAQ_FIFO_SYNC = -1014070,
    ERR_DAQ_OVERLOAD = -1014060,
    ERR_DAQ_PROCESS = -1014061,
    ERR_DAQ_OCCF = -1014055,
    ERR_DAQ_LIMSE = -1014050,
    ERR_DAQ_NOSIGNALSUBSCRIBED = -1014030,
    ERR_DAQ_BUFFERSIZE = -1014025,
    ERR_DAQ_AMPOFF = -1014023,
    ERR_DAQ_OPENRTDRV = -1014010,
    ERR_DAQ_SUBSCR_RUNNING = -1014006,
    ERR_DAQ_SUBSCR_UUID = -1014005,
    ERR_DAQ_SUBSCR_STATE = -1014004,
    ERR_DAQ_SUBSCRITION = -1014002,
    ERR_DATA_FORMAT_NOT_SUPPORTED = -1010000,
    ERR_SRV_STAT_COMPOSER = -1004305,
    ERR_MTHD_SYS_NORESTARTFKT = -1004260,
    ERR_MTHD_SYS_WRONGCHANTYPE = -1004242,
    ERR_MTHD_SYS_WRONGCHAN = -1004240,
    ERR_MTHD_SYS_INVALIDFWFILE = -1004208,
    ERR_MTHD_SYS_FILENOTFOUND = -1004201,
    ERR_OPENFILE = -1004200,
    ERR_WRITE_FILE = -1004199,
    ERR_READ_FILE = -1004198,
    ERR_PARSE_XML_SETTING_REFUSED = -1004160,
    ERR_PAR_WRONG_TYPE = -1004153,
```

```
WARN_PAR_NODEFSET = 1004148,
ERR_PAR_VIEWID_INUSE = -1004146,
ERR_PAR_READING_VAL = -1004144,
WARN_PAR_EMPTY_VAL = 1004143,
ERR_PAR_VIEWINVALID = -1004142,
ERR_PAR_WRONGXPATHSTATE = -1004140,
ERR_PAR_XPATHINVALID = -1004138,
ERR_PAR_NOROOT = -1004136,
WARN_PAR_PARSING = 1004134,
ERR_PAR_PARSING = -1004135,
ERR_PAR_INVALIDADDR = -1004118,
WARN_PAR_TAGUNKOWN = 1004112,
ERR_PAR_TAGUNKOWN = -1004111,
ERR_PAR_OUTOFRANGE = -1004110,
WARN_PAR_OUTOFRANGE = 1004110,
ERR_PAR_INVALIDATTR = -1004109,
WARN_PAR_READONLY = 1004108,
ERR_PAR_CHOICEINVALID = -1004107,
ERR_PAR_CNTXTLOST = -1004106,
ERR_PAR_CHOICEVALUESYNTAXINVALID = -1004105,
ERR_PAR_RESTOREFAILED = -1004104,
ERR_PAR_XMLINVALID = -1004103,
ERR_TXTELCONTAINER_TOO_SMALL = -1003001,
ERR_PARSE_PROCESSING_DATA = -1002002,
ERR_PARSE_INVALID_DATA = -1002001,
ERR_UTIL_FILE_RESTORE = -1001140,
ERR_UTIL_INVALID_FILEHANDLE = -1001136,
ERR_UTIL_STRCONVERT = -1001132,
ERR_UTIL_NO_PATTERN_MATCH = -1001131,
ERR_EVFLG_TIMEOUT = -1001130,
ERR_EVFLG_PTHREADS = -1001128,
ERR_UTIL_TIMER_INVALID = -1001125,
WARN_UTIL_TIMER_WASRUNNING = 1001025,
ERR_UTIL_LANGUAGE = -1001110,
ERR_COMPOSER_TAG_NOT_DEFINED = -1001002,
ERR_NET_NETWORKADAPTER = -1000680,
ERR_UDP_BINDTODEVICE = -1000658,
ERR_UDP_BIND = -1000656,
ERR_UDP_SETBROADCAST = -1000654,
ERR_UDP_SETREUSEADR = -1000652,
ERR_UDP_OPENSOCK = -1000650,
ERR_TCP_BIND = -1000644,
ERR_TCP_LISTEN = -1000642,
ERR_TCP_CHECKIP = -1000638,
ERR_TCP_GETADR = -1000636,
ERR_TCP_GETPORT = -1000635,
ERR_TCP_SETNODELAY = -1000632,
ERR_TCP_SETNONBLOCK = -1000630,
ERR_TCP_OPENSOCK = -1000628,
ERR_TCP_CONNECT = -1000626,
ERR_TCP_ACCEPT = -1000624,
ERR_TCP_CONNECT_FAILED = -1000620,
ERR_TCP_IPADR_ALREADY_CONNECTED = -1000615,
ERR_TCP_ADR_UNKNOWN = -1000612,
ERR_TCP_SOCKET_EINTR = -1000604,
ERR_TCP_SOCKET_HUP = -1000603,
ERR_TCP_SOCKET_DESCRIPTOR = -1000602,
ERR_CON_REC = -1000524,
ERR_CON_SENDING = -1000522,
ERR_CON_INVALID_TELTYPE = -1000510,
ERR_CON_ADRRESINI_FAILED = -1000504,
```

```
  ERR_CON_CONHNDLR_UUIDUNKNOWN = -1000500,
  ERR_TRANS_READINGHDR = -1000430,
  ERR_TRANS_NOTFOUND = -1000420,
  ERR_TRANS_CANCELED = -1000419,
  ERR_TRANS_SEND_REQ_TIMEOUT = -1000416,
  ERR_TRANS_ACK_TIMEOUT = -1000417,
  ERR_TRANS_REQ_TIMEOUT = -1000418,
  ERR_TRANS_INVALID_TELFORMAT = -1000414,
  ERR_TRANS_REQUESTOR_WRONGPROTTYPE = -1000404,
  ERR_TRANS_RESPONDER_WRONGPROTTYPE = -1000405,
  ERR_TRANS_NOT_FOR_US = -1000406,
  ERR_NO_MSP_CHKSUM = -1000301,
  ERR_SRV_CONTROL = -1000300,
  ERR_SRV_NODATA = -1000275,
  ERR_SRV_NORESPONSE_YET = -1000271,
  ERR_SRV_REQRMTHD_NOREQTEL = -1000265,
  ERR_SRV_PARAMERROR = -1000255,
  ERR_SRV_MTHDOUT_UNKNOWNMTHD = -1000203,
  ERR_SRV_MTHDOUT_WRONGANSW = -1000202,
  ERR_SRV_MTHDOUT_TOOMUCH = -1000201,
  ERR_SRV_UNKNOWNSRV = -1000107,
  ERR_SRV_UNKNOWNMTHD = -1000106,
  ERR_INDEX_OUT_OF_RANGE = -1000000
}
```

Visual Basic

```
Public enum FrameworkError
  ERR_NO_SUCCESS = -1
  ERR_SUCCESS = 0
  ERR_NOT_SUPPORTED = -110
  WARN_UNSPECIFIED = 1000
  WARN_VALUE_CHANGED = 9
  INTERNAL_ACTIVATE_PROFINET = 1
  INTERNAL_ACTIVATE_PROFINET_DONE = 2
  ERR_FIELDBUS_INIT_DRIVER = -1000
  ERR_FIELDBUS_ACCESS_OD = -1001
  ERR_FIELDBUS_READ_ODXML = -1002
  ERR_FIELDBUS_START_OD = -1005
  ERR_FIELDBUS_STOP_OD = -1006
  ERR_FIELDBUS = -1010
  ERR_FIELDBUS_STATECHANGE = -1106
  ERR_FIELDBUS_INVALIDTIME = -1107
  ERR_FIELDBUS_UNAVAILABLPAR = -1109
  ERR_INVALID_REVISION_STRING = -9000
  ERR_MODULETYPE_STRING_EMPTY = -9001
  ERR_MODULELABEL_STRING_EMPTY = -9002
  ERR_INVALID_MAC_ADDRESS_STRING = -9003
  ERR_SCALING_NOT_SUPPORTED = -100101
  WARN_SCALING_CHANGED_TO_STANDARD_FOR_SENSORTYPE = 100102
  ERR_SCALING_NOT_SUPPORTED_BY_SENSORTYPE = -100103
  ERR_SCALING_INVALID_TYPE = -100104
  ERR_SCALING_STRAINGAGE_NOT_SUPPORTED = -100105
  ERR_RESTORING_FACTORY_SCALING = -100106
  ERR_SCALING_DIV_BY_ZERO = -100110
  ERR_FILTERTYPE_NOT_SUPPORTED = -100201
  ERR_FILTERCHARACT_NOT_SUPPORTED = -100202
  ERR_FILTERFREQ_NOT_SUPPORTED = -100203
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_2ND_SIGNAL = -100204
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_2ND_SIGNAL = 100204
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_CLASSIC = -100205
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_CLASSIC = 100205
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_MODULE = -100206
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_MODULE = 100206
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = -100207
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = 100207
  ERR_IIR_FILTER_NOT_SUPPORTED_BY_MODULE = -100208
  WARN_IIR_FILTER_NOT_SUPPORTED_BY_MODULE = 100208
  ERR_IIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = -100209
  WARN_IIR_FILTER_NOT_SUPPORTED_BY_SENSOR_TYPE_OR_MEAS_RANGE = 100209
  WARN_FILTERCHAR_CHANGED_TO_IIR_BESSEL_FIR_NO_HIGHSPEED = 100210
  WARN_FILTERCHAR_CHANGED_TO_IIR_BUTTERWORTH_FIR_NO_HIGHSPEED = 100211
  ERR_FIR_FILTER_NOT_SUPPORTED_BY_HIGHSPEED = -100212
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_HIGHSPEED = 100212
  WARN_FILTER_SWITCHED_OFF_FIR_NO_HIGHSPEED = 100213
  WARN_FILTER_CHARACT_NOT_SUPPORTED_CHANGED_TO_OFF = 100214
  WARN_FILTER_CHARACT_NOT_SUPPORTED_BY_2ND_SIGNAL_CHANGED_TO_OFF = 100215
  WARN_FILTER_CHARACT_NOT_SUPPORTED_BY_HIGHSPEED_CHANGED_TO_OFF = 100216
  WARN_FIR_FILTER_NOT_SUPPORTED_BY_CLASSIC_CHANGED_TO_OFF = 100217
  WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_IIR_BESSEL = 100218
  WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_IIR_BUTTERWORTH = 100219
  WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_FIR_LINEARPHASE = 100220
  WARN_FILTER_OFF_NOT_SUPPORTED_CHANGED_TO_MAX_FIR_BUTTERWORTH = 100221
  ERR_CONNMODE_NOT_SUPPORTED = -100450
  ERR_SENSORTYPE_NOT_SUPPORTED = -100506
  ERR_OUTPUTRATE_VS_FILTER_NOT_SUPPORTED = -100510
  ERR_OUTPUTRATE_NOT_SUPPORTED = -100511
```

```
ERR_PHYSICAL_UNIT_NOT_NATIVE = -100520
WARN_PHYSICAL_UNIT_NOT_NATIVE = 100520
WARN_PHYSICAL_UNIT_CHANGED_TO_NATIVE = 100521
ERR_VOLTAGERANGE_NOT_SUPP = -100700
ERR_CURRENTRANGE_NOT_SUPP = -100710
ERR_WIRING_NOT_SUPPORTED = -100802
ERR_WIRING_NOT_SUPPORTED_CHECK_HW_REVISION = -100803
ERR_EXCITFREQ_NOT_SUPPORTED = -100804
ERR_EXCITAMPL_NOT_SUPPORTED = -100806
ERR_IMPEDANCE_NOT_SUPPORTED = -100807
ERR_PT_RESISTANCE_NOT_SUPPORTED = -100850
ERR_TC_TYPE_NOT_SUPPORTED = -100860
ERR_RESISTOR_NOT_SUPPORTED = -100904
ERR_SSI_CODING_NOT_SUPPORTED = -101002
ERR_PRESSURE_RANGE_NOT_SUPPORTED = -101010
ERR_PRESSURE_MAX_AVAILABLE = -101011
ERR_PRESSURE_TYPE_ONLY_ABSOLUTE_SUPPORTED = -101012
ERR_PRESSURE_TYPE_ONLY_RELATIVE_SUPPORTED = -101013
ERR_PRESSURE_TYPE_NOT_SUPPORTED = -101014
ERR_PRESSURE_SENSOR_NOT_AVAILABLE = -101015
ERR_PRESSURE_MIN_AVAILABLE = -101016
ERR_FREQUENCYRANGE_NOT_SUPPORTED = -101020
WARN_REQUESTED_MAX_COUNTER_EXCEEDS_POSSIBLE = 101010
ERR_SENSORSUPPLY_NOT_SUPPORTED = -101100
ERR_SET_SENSOR_SUPPLY_NOT_ALLOWED_IF_USED_BY_PARAMETER_SET = -101102
ERR_SETZERO_INHIBITED = -101110
ERR_SETZERO_NOT_SUPPORTED = -101111
ERR_ANALOG_OUT_SETZERO = -101202
ERR_PARAMSETTINGS_IND_COUNTER = -102017
ERR_PARAMSETTINGS_IND_FREQUENCY = -102018
ERR_PARAMSETTINGS_BRIDGE = -102020
ERR_PARAMSETTINGS_VOLTAGE = -102021
ERR_PARAMSETTINGS_CURRENT = -102022
ERR_PARAMSETTINGS_THERMOCOUPLE = -102023
ERR_PARAMSETTINGS_PT_ELEMENT = -102024
ERR_PARAMSETTINGS_POTI = -102025
ERR_PARAMSETTINGS_SSI = -102026
ERR_PARAMSETTINGS_COUNTER = -102027
ERR_PARAMSETTINGS_FREQUENCY = -102028
ERR_PARAMSETTINGS_INDBRIDGE = -102029
ERR_PARAMSETTINGS_LVDT = -102030
ERR_PARAMSETTINGS_PRESSURE = -102031
ERR_PARAMSETTINGS_IEPE = -102032
ERR_PARAMSETTINGS_PWM = -102033
ERR_PARAMSETTINGS_PULSEWIDTH = -102034
ERR_PARAMSETTINGS_PERIOD = -102035
ERR_PARAMSETTINGS_RESISTOR = -102036
ERR_IRIG_FORMAT_NOT_SUPPORTED = -102037
WARN_CONNECTOR_SET_TO_VOLTAGE_BECAUSE_ANOTHER_SET_TO_IRIG = 102038
WARN_IRIG_SENSOR_CAN_ONLY_BE_ACTIVE_ONCE = 102039
ERR_PARAMSETTINGS_IND_CRANK_COUNTER = -102040
ERR_PARAMSETTINGS_IND_CRANK_FREQUENCY = -102041
ERR_GAP_POSITION_SYNTAX = -102042
ERR_GAP_POSITION_ZERO_NOT_ALLOWED = -102043
ERR_GAP_POSITION_REPETITION_NOT_ALLOWED = -102044
ERR_GAP_POSITION_VALUE_GREATER_THAN_NUMBER_OF_TEETH = -102045
ERR_INDEX_POSITION_VALUE_GREATER_THAN_NUMBER_OF_TEETH = -102046
ERR_TOO_MUCH_GAP_POSITIONS_DEFINED = -102047
ERR_PULSES_PER_ROUND_OUT_OF_RANGE = -102048
WARN_PULSES_PER_ROUND_OUT_OF_RANGE_SET_TO_ZERO = 102048
ERR_INDEX_SHIFT_OUT_OF_RANGE = -102049
```

```
WARN_INDEX_SHIFT_OUT_OF_RANGE_SET_TO_ZERO = 102049
WARN_INDEX_DIVISOR_OUT_OF_RANGE_SET_TO_MINIMUM = 102050
WARN_INDEX_DIVISOR_OUT_OF_RANGE_SET_TO_MAXIMUM = 102051
ERR_PARAMSETTINGS_CRANK_COUNTER = -102052
ERR_PARAMSETTINGS_CRANK_FREQUENCY = -102053
ERR_PARAM_ZERO_NOT_VALID = -103000
ERR_PREVIOUS_SETTING_RESTORED = -104010
ERR_PARAMETERIZATION_REFUSED = -104011
ERR_SCALING_LINTABLE_WRONG_NUMBER_OF_POINTS = -104099
WARN_SCALING_NR_OF_SEGMENTS_CHANGED = 104100
WARN_SCALING_NR_OF_COEFFS_CHANGED = 104101
ERR_SPACE_IN_STRING_NOT_ALLOWED = -104103
ERR_STRING_SIZE_TOO_LONG = -104104
ERR_SCM_HV_NOT_SUPPORTED_BY_VOLTAGE_RANGE = -104105
ERR_VOLTAGE_EXCEEDS_MEAS_RANGE = -104106
ERR_HIGHVOLTAGE_ADAPTOR_NOT_SUPPORTED = -104107
ERR_VOLTAGE_RANGE_OR_HIGHVOLTAGE_ADAPTOR_NOT_SUPPORTED = -104108
WARN_EXTERNAL_HIGHVOLTAGE_ADAPTOR_NOT_CONNECTED = 104109
ERR_EXTERNAL_HIGHVOLTAGE_ADAPTOR_NOT_CONNECTED = -104109
WARN_EXTERNAL_QUARTER_BRIDGE_ADAPTOR_NOT_CONNECTED = 104110
ERR_EXTERNAL_QUARTER_BRIDGE_ADAPTOR_NOT_CONNECTED = -104110
ERR_EXTERNAL_ADAPTOR_COEFFS_NOT_AVAILABLE = -104111
WARN_OUTPUTUNIT_INCOMPATIBLE_TO_SENSORUNIT = 104112
WARN_OUTPUTUNIT_DISABLED_BECAUSE_PHYS_UNIT_USER_DEF = 104113
WARN_OUTPUTUNIT_IS_USER_DEFINED = 104114
ERR_AUTOCAL_REPTIME_NOT_SUPPORTED = -104200
ERR_AUTOCAL_NOT_SUPPORTED_SENSOR = -104201
ERR_AUTOCAL_DISABLED = -104203
ERR_AUTOCAL_SENSOR_NOT_CONNECTED = -104204
ERR_AUTOCAL_NOT_SUPPORTED_MODULE = -104205
ERR_ADJUSTMENT_SENSOR_NOT_SUPPORTED = -104206
WARN_ADJUSTMENT_SENSOR_NOT_SUPPORTED = 104206
WARN_ADJUSTMENT_REPTIME_TOO_SHORT = 104207
WARN_ADJUSTMENT_REPTIME_TOO_LONG = 104208
WARN_ADJUSTMENT_NOT_DEFAULT = 104209
ERR_AUTOADJUST_NOT_SUPPORTED_SENSOR = -104211
ERR_AUTOCAL_CYCLE_IS_RUNNING_REPEAT_LATER = -104212
ERR_TEDS_NO_SENSOR_TEMPLATE = -105000
ERR_TEDS_NO_DATA_OR_INVALID = -105001
ERR_TEDS_NO_MIN_PHYS_VAL = -105002
ERR_TEDS_NO_MAX_PHYS_VAL = -105003
ERR_TEDS_NO_MIN_ELEC_VAL = -105004
ERR_TEDS_NO_MAX_ELEC_VAL = -105005
ERR_TEDS_NO_MDEF_UCC = -105007
ERR_TEDS_CALCURVE_COEFFS_INVALID = -105008
ERR_TEDS_NODATA = -105010
ERR_TEDS_NOPARSEBIN = -105014
ERR_TEDS_NOAMP = -105018
ERR_TEDS_NO_EXCITE_AMPL_NOM = -105020
ERR_TEDS_NO_EXCITE_AMPL_MIN = -105022
ERR_TEDS_NO_EXCITE_AMPL_MAX = -105024
ERR_TEDS_NO_ATTACHED_SERIALNUM = -105025
ERR_TEDS_NO_SENSOR_IMPED = -105030
ERR_TEDS_GAUGEFACTOR = -105031
ERR_TEDS_NO_BRIDGETYPE = -105032
ERR_TEDS_TC_TYPE = -105034
ERR_TEDS_PT_TYPE = -105036
ERR_TEDS_LOWPASSFILTER = -105038
ERR_TEDS_ZEROSPANSCALING = -105040
ERR_TEDS_GAINSCALING = -105041
ERR_TEDS_PARAM = -105050
```

```
ERR_TEDS_NOT_SUPPORTED_BY_CONNECTOR = -105051
WARN_TEDS_NOT_SUPPORTED_CHANGED_TO_IGNORE = 105052
ERR_TEDS_NO_CALTABLE = -105054
ERR_TEDS_NO_CALCURVE = -105055
ERR_TEDS_CALCURVE_POWER_INVALID = -105056
ERR_TEDS_CALTABLE_NOT_ENOUGH_POINTS = -105059
ERR_TEDS_CALTABLE_TOO_MUCH_POINTS = -105060
ERR_TEDS_BUFFER_TOO_SMALL = -105061
ERR_TEDS_WRITE_NOT_ALLOWED = -105062
ERR_TEDS_TID_DONT_MATCH = -105063
ERR_TEDS_SCALINGTYPE_NOT_SUPPORTED = -105100
ERR_TEDS_SCALING_PARAMETER_INVALID = -105101
ERR_TEDS_UNIT_CONVERSION_NOT_SUPPORTED = -105102
ERR_SPEEDMODE_CHANGING_NOT_POSSIBLE_SWITCH_OFF = -105201
ERR_SPEEDMODE_TOO_MANY_CONNECTORS_HIGH = -105202
ERR_SPEEDMODE_MIXED_MODE_NOT_POSSIBLE = -105204
WARN_SPEEDMODE_CHANGED_TO_HIGH = 105205
WARN_HIGHSPEEDMODE_NOT_SUPPORTED = 105206
WARN_SCALING_PARAMS_DONT_SUPPORT_HIGHSPEEDMODE = 105207
ERR_SPEEDMODE_HIGH_NOT_SUPPORTED = -105208
WARN_SENSOR_PARAMS_DONT_SUPPORT_HIGHSPEEDMODE = 105209
ERR_CONNECTORSOURCE = -105210
ERR_PHYSUNIT_NOT_FOUND = -106000
ERR_PHYSUNIT_NOT_RELATED = -106004
ERR_PHYSUNIT_SCALING_FACTOR_ZERO = -106006
WARN_SUPPLY_VOLTAGE_SWITCHED_OFF = 119000
WARN_WIRING_ALTERED = 119001
WARN_SUPPLY_VOLTAGE_CHANGED_TO_MAX_POSSIBLE = 119002
WARN_EXCITATION_FREQ_CHANGED = 119003
WARN_EXCITATION_AMPL_CHANGED = 119004
ERR_SUPPLY_VOLTAGE_BELOW_MINIMUM = -119005
WARN_SSI_CLK_FREQ_CHANGED = 119005
WARN_SSI_RESOLUTION_CHANGED = 119006
WARN_FILTER_CUTOFF_FREQ_CHANGED = 119010
WARN_FILTERTYPE_CHANGED = 119011
WARN_FILTERCHAR_CHANGED = 119012
WARN_OUTPUTRATE_CHANGED = 119015
ERR_GLITCHFILTER_NOT_SUPPORTED = -119024
WARN_GLITCHFILTER_CHANGED = 119025
ERRHAL_ERROR = -120000
ERRHAL_OPENMSPDRV = -120100
ERRHAL_OPENCPLDDRV = -120110
ERRHAL_GETAMPTEMP = -120120
ERRHAL_GETAMPTEMPCAL = -120121
ERRHAL_READ_I2C = -120125
ERRHAL_WRITE_I2C = -120126
ERRHAL_WRITE_MSP = -120127
ERRHAL_ENABLECHN = -120130
ERRHAL_SETMEASINPUT = -120134
ERRHAL_SETSENSCIRCUIT = -120135
ERRHAL_SETCALINPUT = -120140
ERRHAL_SETGAINCAL = -120145
ERRHAL_SETGAINMEAS = -120150
ERRHAL_SET_SENSORSUPPLY = -120155
ERRHAL_SETFEEDBACK = -120159
ERRHAL_SETEXCIT = -120160
ERRHAL_GETSNAMP = -120165
WARN_HAL_GETSNAMP = 120166
ERRHAL_SETSNAMP = -120170
ERRHAL_GETHWRAMP = -120175
WARN_HAL_GETHWRAMP = 120176
```

```
ERRHAL_SETHWRAMP = -120180
ERRHAL_GETTID = -120185
ERRHAL_GETTEDSDATHW = -120189
ERRHAL_GETTEDSDAT = -120190
ERRHAL_TEDSCHECKSUM = -120191
ERRHAL_NOTEDS = -120192
ERRHAL_TEDS_ALLZERO = -120193
ERRHAL_TEDSCHECKSUMINSERT = -120194
ERRHAL_SETTEDSDAT = -120195
ERRHAL_SETTEDSCON = -120196
ERRHAL_SIZETEDSDAT = -120197
ERRHAL_SETLEDAMP = -120200
ERRHAL_SETSYSLEDAMP = -120201
ERRHAL_WRSETTING = -120205
WARNHAL_WRSETTING = 120205
ERRHAL_SETTING_NOT_AVAILABLE = -120206
ERRHAL_SETADCCLK = -120210
ERRHAL_GETMAXSRAMP = -120211
ERRHAL_SET_FM_CONF = -120212
ERRHAL_SET_SSI_CONF = -120213
ERRHAL_GETMSPVERAMP = -120215
ERRHAL_GETMSPSTATAMP = -120216
ERRHAL_SYNCADC = -120220
ERRHAL_SETCHNPROP = -120225
ERRHAL_GETMSPREG = -120230
ERRHAL_SETMSPREG = -120235
ERRHAL_SETTSTSIGMEAS = -120240
ERRHAL_SETTSTSIGCAL = -120245
ERRHAL_SETSIMUVAL = -120246
ERRHAL_SETSIMUVALFORMAT = -120248
ERRHAL_SETSIMUVALTYPE = -120249
ERRHAL_GETSNAPSHOT = -120250
ERRHAL_SNAPSHOT_MAXDATA = -120251
ERRHAL_CALCPHASE = -120252
ERRHAL_NO_VALID_SNAPSHOTDATA = -120253
ERRHAL_GETCALDATHW = -120256
ERRHAL_GETCALDAT = -120257
ERRHAL_SETCALDAT = -120258
ERRHAL_SETCSURCAL = -120260
ERRHAL_GETCSURCAL = -120265
ERRHAL_LDRCHKSUM = -120266
ERRHAL_STORESENSSUPPLYOFFS = -120267
ERRHAL_GETSENSSUPPLYOFFS = -120268
WARN_HAL_GETCSURCAL = 120269
ERRHAL_SETMEASVALINVAL = -120270
WARN_HAL_GETSENSSUPPLYOFFS = 120271
ERRHAL_SELFPGACHN = -120272
ERRHAL_REDDATRATE = -120274
ERRHAL_RATE_NOT_SUP_BY_FPGA = -120276
ERRHAL_FB_NOT_AVAILABLE = -120277
ERRHAL_FB_UNKNOWN_PROPERTY = -120278
ERRHAL_FPGATEST_NOT_SUPPORTED = -120279
WARN_HAL_GETPOTIWIPERFROMMSP = 120280
ERRHAL_GETPOTIWIPERFROMMSP = -120281
ERRHAL_POTIWIPERCOMPARE = -120282
ERRHAL_SETIEPEHP = -120283
ERRHAL_FB_ARBSIGGEN_BUFFER_BUSY = -120284
ERRHAL_GETHWIDENTPF = -120300
WARN_HAL_GETHWIDENTPF = 120301
ERRHAL_SETHWIDENTPF = -120305
ERRHAL_SETADDHWREVPF = -120306
```

```
ERRHAL_SETADDHWIDENTPF = -120307
ERRHAL_GETHWRPF = -120310
ERRHAL_GETADDHWRPF = -120311
ERRHAL_GETADDHWIDENTPF = -120312
WARN_HAL_GETHWRPF = 120313
WARN_HAL_GETADDHWRPF = 120314
ERRHAL_SETHWRPF = -120315
WARN_HAL_GETADDHWIDENTPF = 120316
ERRHAL_GETPLLVOLTAGE = -120317
WARN_HAL_GETPLLVOLTAGE = 120318
ERRHAL_SETPLLVOLTAGE = -120319
ERRHAL_SETMACADDR = -120320
ERRHAL_GETMACADDR = -120325
ERRHAL_SETLEDPF = -120330
ERRHAL_GETPFTEMP = -120335
ERRHAL_GETMODVOL = -120340
ERRHAL_GETMODCUR = -120345
ERRHAL_GETEXTCUR = -120350
ERRHAL_GETMSPAPPLDATEPF = -120358
ERRHAL_GETMSPLOADERDATEPF = -120359
ERRHAL_GETMSPVERPF = -120360
ERRHAL_GETMSPSTATPF = -120361
ERRHAL_GETLIMCURR = -120362
WARN_HAL_GETMSPVERPF = 120363
ERRHAL_SETTSTSIGGEN = -120365
ERRHAL_STRTDATSTRM = -120367
ERRHAL_STOPDATSTRM = -120369
ERRHAL_IISDUMDAT = -120371
ERRHAL_ENIIS = -120373
ERRHAL_RSTIIS = -120375
ERRHAL_CHKFLHETRY = -120377
ERRHAL_GETMAXSRPF = -120379
ERRHAL_OPENSNAPSHOTDRV = -120405
ERRHAL_CLOSESNAPSHOTDRV = -120406
ERRHAL_ENCXUPPER = -120408
ERRHAL_SIGNUMBER = -120410
ERRHAL_FOSDELAY = -120412
ERRHAL_SETSCALPARS = -120414
ERRHAL_COMPENSATION_OFFSET = -120415
ERRHAL_DACINPUTTYPE = -120416
ERRHAL_UNKNOWNSYNCCODE = -120417
ERRHAL_ADDFUNCMATH = -120418
ERRHAL_UNKNOWN_MATH_TYPE = -120420
ERRHAL_PEAK_OUTRATE = -120422
ERRHAL_CX_SYNC_TEST = -120424
ERRHAL_SETZERO_NO_VALID_MEASVAL = -120426
ERRHAL_SET_FILTER_FIR = -120428
ERRHAL_PARAM_IS_ZERO = -120429
ERRHAL_FIR_TAB_CNT = -120430
ERRHAL_FIR_TAB_CNT_SUM = -120431
ERRHAL_SETDEMODFACTORS = -120432
ERRHAL_SETFIRDEMODFILTER = -120433
ERRHAL_SETPOLYCOEFFSTOFPGA = -120434
ERRHAL_FPGAOPENDRV = -120500
ERRHAL_FPGACLOSEDRV = -120501
ERRHAL_FPGABUSTST = -120502
ERRHAL_FPGAREAD = -120503
ERRHAL_FPGAWRITE = -120504
ERRHAL_FPGAINITI2C = -120505
ERRHAL_FPGARSTI2C = -120506
ERRHAL_FPGARSTHZR = -120507
```

```
ERRHAL_FPGAMMAP = -120509
ERRHAL_FPGAMUNMAP = -120510
ERRHAL_FPGAGETTIME = -120511
ERRHAL_FPGARESET = -120512
ERRHAL_FPGAIOCTL = -120513
ERRHAL_MSPGETTMP = -120700
ERRHAL_MSPDISOVRCURRPROT = -120701
ERRHAL_MSPGETALLSWITCHES = -120702
ERRHAL_MSPSETALLSWITCHES = -120703
ERRHAL_MSPMEAS = -120705
ERRHAL_MSPRDCTRREG = -120710
ERRHAL_MSPWRCTRREG = -120715
ERRHAL_MSPWRREGBIT = -120720
ERRHAL_MSPSETSHUNT = -120721
ERRHAL_MSPSETVARIANT = -120722
ERRHAL_MSPSETWIRING = -120723
ERRHAL_MSPWRINFO = -120725
ERRHAL_MSPRDINFO = -120730
ERRHAL_MSPSETPOTI = -120740
ERRHAL_MSPGETPOTI = -120745
ERRHAL_MSPGETSLNUM = -120750
ERRHAL_MSPGETVERSION = -120755
ERRHAL_MSPGETSTAT = -120756
ERRHAL_MSPGETLIMITEDSENSORSUPPLY = -120757
ERRHAL_MSPGETERR = -120760
ERRHAL_MSPDOOWRST = -120761
ERRHAL_MSPSETOWMODE = -120762
ERRHAL_TEDSCOM_NOT_POSSIBLE = -120763
ERRHAL_MSPGETFOWNUM = -120765
ERRHAL_MSPGETOWDAT = -120775
ERRHAL_MSPSETOWDAT = -120780
ERRHAL_MSPBEEP = -120796
ERRHAL_MSPPRGBLKV = -120800
ERRHAL_MSPPRGBLK = -120805
ERRHAL_MSPRDBLK = -120810
ERRHAL_MSPPRERDBLK = -120811
ERRHAL_MSPRDRFID = -120815
ERRHAL_MSPWRRFID = -120816
ERRHAL_MSPSETLED = -120817
ERRHAL_MSPRD_NO_RFID = -120818
ERRHAL_MSPRD_ERR_RFID = -120819
ERRHAL_MSP_EEPROM_SIZE = -120820
ERRHAL_MSPWR_EEPROM = -120821
ERRHAL_MSPRD_EEPROM = -120822
ERRHAL_MSP_SET_ADC_LPM = -120823
ERRHAL_MSP_SET_VARIANT = -120824
ERRHAL_SPISETDACPLL = -120900
ERRHAL_SPIOPENDRV = -120905
ERRHAL_SPISETSS = -120910
ERRHAL_MAX11245_REG_ACCESS = -120950
ERRHAL_MAX11245_CMD_REG_BUSY = -120951
ERRHAL_MAX11245_UNABLE_TO_POWER_DOWN = -120952
ERR_ADJUST_VALUE_OUT_OF_RANGE = -130490
ERR_ADJUSTMENT_TIMEOUT = -130491
ERR_ADJUSTMENT_CANCELLED = -130492
ERR_ADJUST_ZERO_VALUE_OUT_OF_RANGE = -130493
ERR_ADJUST_CAL_VALUE_OUT_OF_RANGE = -130494
ERR_HAL_CALCAL = -130500
ERR_HAL_CALMEAS = -130505
ERR_HAL_CORRECTPHASE = -130510
ERR_HAL_CALC_PT_COEFFS = -130513
```

```
ERR_HAL_ACTIVATESET = -130515
WARN_HAL_ACTIVATESET = 130515
WARN_SET_ZERO_INCL_ANALOGOUT = 130517
WARN_SET_ZERO = 130518
WARN_SIG_NOT_AVAILABLE = 130519
WARN_ADJUST_RMS_PERIOD = 130522
ERRHAL_LOOP = -130524
ERRHAL_CORRTEMPERATURE = -130526
ERRHAL_COLDJUNCTION = -130528
WARN_COLDJUNCTION_INITJUST_MISSING = 130529
ERRHAL_AUTOADJUST = -130530
ERRHAL_EXTERNAL_ADAPTOR = -130532
ERRHAL_WRONG_CHANTYPE = -130560
ERRHAL_NOT_SUPPORTED = -130600
ERR_CONNECTOR_NOT_ENABLED = -130610
WARN_6_WIRING_POSSIBLE = 130620
ERR_WIRING_2_BROKEN = -130622
ERR_WIRING_3_BROKEN = -130623
ERRHAL_INTERPOLENABLE = -130700
ERRHAL_INTERPOLINIT = -130701
ERRHAL_INTERPOLSETDATARATE = -130703
ERRHAL_INTERPOLSETSKALINGOVERFLOW = -130704
ERRHAL_INTERPOLSETSCALING = -130705
ERRHAL_SCALDACNULL = -130706
ERRHAL_SCALDACGAIN = -130707
ERRHAL_SCALDACNULL_MX410 = -130708
WARNHAL_SCALDACNULL_MX410 = 130708
ERRHAL_SCALDACGAIN_MX410 = -130709
WARNHAL_SCALDACGAIN_MX410 = 130709
WARN_CONFIGURE_FIREWIRE_FUNCTIONBLOCK = 132000
ERR_CREATE_FIREWIRE_FUNCTIONBLOCK = -132001
ERR_RT_IOCTL = -140020
ERR_RT_COPY_TO_USER = -140024
ERR_RT_COPY_FROM_USER = -140025
ERR_TEDS_PROPERTY_NOT_FOUND = -160002
ERR_TEDS_DATA_CORRUPT = -160010
ERR_TEDS_NO_DISCSIGCONFIG = -160020
ERR_TEDS_NO_ELECSIGTYPE = -160021
ERR_TEDS_NO_MAPMETH = -160022
ERR_TEDS_CALCURVE_DOMAIN_INVALID = -160023
ERR_TEDS_CALTABLE_DOMAIN_INVALID = -160024
ERR_TYPE_NOT_SUPPORTED = -165000
ERR_SELECTION_NOT_SUPPORTED = -165001
ERR_MATH_TYPE_NOT_SUPPORTED = -165002
ERR_ONLY_SINGLE_CONNECTOR_POSSIBLE = -165003
ERR_CAN_DECODER_MODESTARTBIT_511 = -178000
ERR_CAN_DECODER_STARTBIT_511 = -178001
ERR_CAN_MODE_NOT_SUPPORTED = -178010
ERR_CAN_FDFRAME_ONLY_CLASSICAL_SUPPORTED = -178011
WARN_CAN_FDFRAME_ONLY_CLASSICAL_SUPPORTED = 178011
ERR_CAN_FDFRAME_ONLY_CLASSICAL_AVAILABLE = -178012
WARN_CAN_FDFRAME_ONLY_CLASSICAL_AVAILABLE = 178012
ERR_CAN_ONLY_ENDLESS_RETRANSMISSION_SUPPORTED = -178013
ERR_CAN_RETRANSMISSION_OUT_OF_RANGE = -178014
ERR_CAN_TRANSMIT_BYTE_NOT_VALID_HEX_VALUE = -178020
ERR_ETHERCAT_PARAMETERS = -179800
ERR_PROFINET_PARAMETERS = -179850
WARN_PROFINET_PARAMETERS = 179850
ERR_PROFINET_PARAM_TIMEOUT = -179851
ERR_ADDCONN_MODE_NOT_SUPPORTED = -179900
WARN_ANALOG_OUT_GUARD_TIMEOUT = 179901
```

```
WARN_ANALOG_OUT_ONTIMEOUT_VOLTAGE_MAX_POSSIBLE = 179902
WARN_ANALOG_OUT_ONTIMEOUT_VOLTAGE_MIN_POSSIBLE = 179903
WARN_ANALOG_OUT_GUARDTIME_MIN_POSSIBLE = 179904
WARN_ANALOG_OUT_GUARDTIME_MAX_POSSIBLE = 179905
ERR_ANALOG_OUT_MODE_NOT_SUPPORTED = -179910
WARN_ANALOG_OUT = 179911
WARN_ANALOG_OUT_SCALING_NOT_SUPPORTED_DISABLED = 179914
ERR_ANALOGOUT_NOT_AVAILABLE_IF_ANALOGIN_POLYNOMIAL_SCALING = -179915
ERR_DIGITALIO_MODE_NOT_OUTPUT = -179919
ERR_DIGITALIO_MODE_NOT_SUPPORTED = -179920
ERR_DIGITALIO_TRIGGER_LEVEL_TOO_LOW = -179921
ERR_DIGITALIO_TRIGGER_LEVEL_TOO_HIGH = -179922
WARN_SET_DIGITAL_IO_IS_INPUT = 179930
WARN_SET_DIGITAL_IO_IS_CONNECTED = 179931
ERR_MASK_RESULT_LONGER_THAN_64_BITS = -179937
ERR_MASK_SYNTAX = -179938
ERR_DELTA_MUST_BE_GREATER_THAN_ZERO = -179939
ERR_DIVISOR_MUST_BE_GREATER_THAN_ZERO = -179940
WARN_DIVISOR_SET_TO_ZERO = 179940
ERR_CAN_LISTEN_ONLY = -179941
WARN_CAN_TRANSMIT_LISTEN_ONLY = 179943
WARN_TIMER_SET_TO_NEAREST_POSSIBLE = 179944
WARN_TIMER_SET_TO_MINIMUM_POSSIBLE = 179945
ERR_CAN_CCP_DBC_DATA_INVALID = -179950
ERR_CAN_CCP_ERROR_CREATING_LOGGER = -179951
ERR_CAN_CCP_NOT_INITIALIZED = -179952
ERR_MATHTYPE_NOT_SUPPORTED = -185100
WARN_CHECK_ADDFUNC_PARAMETERS = 185101
WARN_MATH_DISABLED_SIGNALREFERENCE_OFF = 185102
WARN_PEAK_DISABLED_SIGNALREFERENCE_OFF = 185103
WARN_PEAK_DISABLED_ANALOGIN_SCALING_NOT_POSSIBLE = 185104
ERR_PEAK_NOT_POSSIBLE_ANALOGIN_SCALING = -185105
ERR_TIME_NEGATIVE_OR_ZERO = -185309
ERR_TIME_NEGATIVE = -185310
WARN_MATH_RMS_PERIOD_CHANGED_TO_MIN = 185311
WARN_MATH_ROTVIBE_OUTPUTFILTER_SWITCHED_OFF = 185317
WARN_MATH_ROTVIBE_OUPUT_HIGHPASS_ALIGNED = 185318
WARN_MATH_ROTVIBE_FILTER_SWITCHED_OFF = 185319
WARN_MATH_ROTVIBE_HIGHPASS_ALIGNED = 185320
WARN_MATH_SCALINGFACTOR_ZERO = 185322
ERR_MATH_PULSESPERROUND_LESS_THAN_1 = -185323
ERR_MATH_INTEGRATOR_NOT_ENABLED = -185324
ERR_MATH_UNIT_NOT_ENABLED = -185325
WARN_MATH_UNIT_NOT_ENABLED = 185325
ERR_MATH_PID_MODE_NOT_SUPPORTED = -185326
ERR_SEGMENT_COUNT_TOO_SMALL = -185327
ERR_SEGMENT_COUNT_TOO_BIG = -185328
ERR_MATH_PARAM_REFUSED = -185396
ERR_PEAK_PARAM_REFUSED = -185397
ERR_GATE_PARAM_REFUSED = -185398
ERR_TESTSIGNAL_MODE_NOT_SUPPORTED = -185400
ERR_TESTSIGNAL_NOT_SUPPORTED = -185401
ERR_TESTSIGNAL_AMPLITUDETYPE_NOT_SUPPORTED = -185402
ERR_TESTSIGNAL_AMPLITUDE_LEVEL_MIN_MAX = -185403
ERR_TESTSIGNAL_AMPLITUDE_PEAKPEAK_DELTA = -185404
ERR_TESTSIGNAL_ZERO_DIVISION = -185405
ERR_TESTSIGNAL_FREQUENCY = -185407
ERR_TESTSIGNAL_WRONG_INDEX = -185410
WARN_TESTSIGNAL_DUTYCYCLE_SET_TO_MIN = 185412
WARN_TESTSIGNAL_DUTYCYCLE_SET_TO_MAX = 185413
WARN_TESTSIGNAL_PHASE_SET_TO_MIN = 185414
```

```
WARN_TESTSIGNAL_PHASE_SET_TO_MAX = 185415
ERR_TESTSIGNAL_VOLTAGE_OUT_OF_RANGE = -185416
ERR_TESTSIGNAL_ARBITRATION_SOURCETYPE_NOT_SUPPORTED = -185420
ERR_TESTSIGNAL_ARBITRATION_RUNMODE_NOT_SUPPORTED = -185421
ERR_TESTSIGNAL_ARBITRATION_REPETITIONRATE_NOT_SUPPORTED = -185422
ERR_TESTSIGNAL_ARBITRATION_BURSTMODE_NOT_SUPPORTED = -185423
ERR_TESTSIGNAL_NOT_ACTIVE_OR_NOT_CONSTANT = -185430
ERR_SIGNAL_GENERATOR_MODE_NOT_SUPPORTED = -185440
ERR_SIGNAL_GENERATOR_NOT_ENABLED = -185441
ERR_TESTSIGNAL_PARAM_REFUSED = -185499
WARN_MODULE_NOT_READY_FOR_FIELDBUS_ACQ = 1515710
ERR_ECAT_FUNCTIONBLOCK_SET_PDO_STATUS = -1515708
ERR_ECAT_FUNCTIONBLOCK_SET_PDO_INDEX = -1515707
ERR_ECAT_FUNCTIONBLOCK_CREATE_PROBLEM = -1515706
ERR_ECAT_RTSIGNALE_NOT_VALID = -1515705
WARN_ECAT_DEMUX_SIGNALE_NOT_VALID = 1515704
WARN_ECAT_MODULE_PARAM_NOT_READY = 1515703
ERR_ECAT_MODULE_STATUS_NOT_READABLE = -1515702
WARN_ECAT_CORRECTED_FW_SIGNALE_LIST_SIZE = 1515701
ERR_ECAT_COUNT_OF_FW_SIGNALS_LIMIT_REACHED = -1515700
ERR_NTP_SERVER_NOT_REACHED = -1515600
ERR_FW_NO_NEC_CHANNEL = -1018005
ERR_FW_NO_NEC_DOWNLOAD = -1018006
ERR_FW_NO_MSP_CHANNEL = -1018000
WARN_EXCITATIONCALIBRATION_DATA_MISSING = 1017998
WARN_CALCALIBRATION_DATA_MISSING = 1017997
ERR_MSP_DATA_CORRUPT = -1017996
ERR_INITJUST_FROM_MSP = -1017995
ERR_NO_VALID_INITJUST = -1017993
ERR_STORING_INITJUST = -1017992
ERR_GETTING_INITJUST = -1017991
ERR_PLLVOL_OUTOFRANGE = -1017990
ERR_PLLVOL_CORRUPT = -1017989
ERR_PLLVOL_FROM_MSP = -1017988
ERR_DATA_RATE_DOMAIN_NOT_SUPPORTED = -1016100
ERR_SYS_FILENAME = -1016050
ERR_SYS_FILE_TOO_LARGE = -1016049
ERR_SYS_BAD_DELIVERY = -1016045
ERR_SYS_FRAMEWORK_START = -1016035
ERR_SYS_OPEN_CLOSE_WD = -1016030
ERR_SYS_INVALID_TIME = -1016025
ERR_SYS_TOO_MANY_EVENTS = -1016003
ERR_SYS_CREATING_OBJECT = -1016002
ERR_SPI_OPEN_DRIVER = -1015050
ERR_SPI_WAKEUP_DAC = -1015049
ERR_SPI_WRITE_DRIVER_DIGITS = -1015048
ERR_SPI_WRITE_DRIVER_COMMAND = -1015047
ERR_SPI_VOLT_OUTOF_RANGE = -1015040
ERR_SPI_PLLDAC = -1015039
ERR_SYNC_OPEN_DRIVER = -1015033
ERR_SYNC_ANY_TSD_PROBLEM = -1015032
ERR_SYNC_NOT_ALL_MODULES_READY = -1015031
ERR_SYNC_TIMESOURCE_DATA_NOT_VALID = -1015030
ERR_SYNC_ACTIVATE_ETHERCAT = -1015029
ERR_SYNC_ACTIVATE_IRIG = -1015028
ERR_SYNC_ACTIVATE_PTP = -1015027
ERR_SYNC_ACTIVATE_NTP = -1015026
ERR_SYNC_ACTIVATE_AUTO = -1015025
ERR_PARAM_IP_SUBNETMASK_SYNTAX = -1015024
ERR_PARAM_IP_ADDRESS_SYNTAX = -1015023
ERR_PARAM_SYNC_TIMESOURCE_NOT_SUPPORTED = -1015022
```

```
ERR_SYNC_CYCLEDIFF_TOO_LARGE = -1015021
ERR_SYNC_NOT_IN_SYNC = -1015009
ERR_ADJ_MEASRANGE_INVALID = -1014994
ERR_ADJ_MEASRANGE_OVERRUN = -1014993
ERR_ADJ_WRONG_CONNECTOR = -1014992
ERR_CONN_WRONG_CONNECTOR = -1014801
ERR_ANALOGOUT_NOCONN = -1014603
ERR_ANALOGOUT_SIGOVFL = -1014602
ERR_ANALOGOUT_OCCF = -1014601
ERR_ANALOGOUT_OVFL = -1014600
ERR_SIGNALREFERENCE_DECODERINDEX_NOT_SUPPORTED = -1014515
ERR_SIGNALREFERENCE_DECODERINDEX_LOCAL_NOT_SUPPORTED = -1014514
ERR_SIGNALREFERENCE_SIGNALGENERATORINDEX_NOT_SUPPORTED = -1014513
ERR_SIGNALREFERENCE_SIGNALGENERATORINDEX_LOCAL_NOT_SUPPORTED = -1014512
ERR_SIGNALREFERENCE_DIGITALIOINDEX_LOCAL_NOT_SUPPORTED = -1014511
ERR_SIGNALREFERENCE_PEAKINDEX_LOCAL_NOT_SUPPORTED = -1014510
ERR_SIGNALREFERENCE_MATHINDEX_LOCAL_NOT_SUPPORTED = -1014509
ERR_SIGNALREFERENCE_ALARMINDEX_LOCAL_NOT_SUPPORTED = -1014508
ERR_SIGNALREFERENCE_CONNECTOR_LOCAL_NOT_SUPPORTED = -1014507
ERR_SIGNALREFERENCE_SIGNALINDEX_LOCAL_NOT_SUPPORTED = -1014506
ERR_SIGNALREFERENCE_CANNOT_CREATE = -1014505
ERR_SIGNALREFERENCE_UNKNOWN = -1014504
WARN_SIGNALREFERENCE_UNKNOWN = 1014504
ERR_PARAM_LEVELS_IDENTICAL = -1014503
ERR_SIGNALREFERENCE_CONNECTOR_NOT_SUPPORTED = -1014502
WARN_MAX_ISO_SIGNALINDEX_REACHED = 1014501
ERR_SIGNALREFERENCE_SIGNALINDEX_NOT_SUPPORTED = -1014500
ERR_INDEX_NOT_SUPPORTED = -1014499
ERR_SIGNALREFERENCE_SIGNALTYPE_NOT_SUPPORTED_OR_SYNTAX_ERROR = -1014498
WARN_SIGNALREFERENCE_EMPTY = 1014497
ERR_MODULEREFERENCE_SYNTAX = -1014496
ERR_SIGNALREFERENCE_PEAKINDEX_NOT_SUPPORTED = -1014495
ERR_SIGNALREFERENCE_MATHINDEX_NOT_SUPPORTED = -1014494
ERR_SIGNALREFERENCE_ALARMINDEX_NOT_SUPPORTED = -1014493
ERR_MODULEREFERENCE_NOT_LOCAL = -1014492
ERR_MODULEREFERENCE_NOT_EXTERNAL = -1014491
ERR_SIGNALREFERENCE_SYNTAX = -1014490
ERR_PHYSUNIT_NOT_AVAILABLE = -1014489
ERR_SIGNALREFERENCE_ASYNCSIGNAL_NOT_ALLOWED_OR_SYNTAX_ERROR = -1014488
ERR_SIGNALREFERENCE_DIGITALIOINDEX_NOT_SUPPORTED = -1014487
ERR_IOCTL_DEVICE = -1014414
ERR_DEVICE_NOT_OPEN = -1014413
ERR_CLOSE_DEVICE = -1014412
ERR_DEVICE_ALREADY_OPEN = -1014411
ERR_OPEN_DEVICE = -1014410
ERR_PINFO_NULL = -1014406
ERR_POINTER_NULL = -1014405
ERR_PARAMSETTINGS_NULL = -1014404
ERR_PARAM_CALCSYNCSIGNAL = -1014403
ERR_PPLATFORM_NULL = -1014402
ERR_PAMP_NULL = -1014401
ERR_PANALOG_NULL = -1014400
ERR_TEDS_DATA = -1014374
ERR_NO_SIGNAL_GENERATOR_UNIT_AVAILABLE = -1014348
ERR_NO_GATE_UNIT_AVAILABLE = -1014347
ERR_NO_PEAK_UNIT_AVAILABLE = -1014346
ERR_NO_MATH_UNIT_AVAILABLE = -1014345
ERR_NO_ALARM_UNIT_AVAILABLE = -1014344
ERR_NO_DIGITAL_IO_AVAILABLE = -1014343
ERR_NO_CAN_CONNECTORS_AVAILABLE = -1014342
ERR_NO_ANALOG_OUT_CONNECTORS_AVALIBALE = -1014341
```


現在位置: Supported devices -> Hbm.Api.QuantumX Namespace -> Hbm.Api.QuantumX.Enums Namespace

```
ERR_NO_ANALOG_IN_CONNECTORS_AVAILABLE = -1014340
ERR_HANDLING_EXTERNAL_ADAPTOR = -1014334
ERR_MODE_NOT_SUPPORTED_BY_SENSORTYPE = -1014333
WARN_MODE_NOT_SUPPORTED_BY_SENSORTYPE = 1014333
ERR_NOT_AVAILABLE_IF_EXTERNAL_ADAPTOR_PRESENT = -1014332
WARN_NOT_AVAILABLE_IF_EXTERNAL_ADAPTOR_PRESENT = 1014332
ERR_EXECUTE_CONTROL = -1014330
WARN_EXECUTE_CONTROL = 1014330
ERR_VALUE_NOT_SUPPORTED = -1014324
WARN_VALUE_NOT_SUPPORTED = 1014324
ERR_AVAILABLE_ONLY_WITH_EXTERNAL_ADAPTOR = -1014323
WARN_AVAILABLE_ONLY_WITH_EXTERNAL_ADAPTOR = 1014323
ERR_CONTROL_NOT_SUPPORTED_BY_SENSORTYPE = -1014322
ERR_CONTROL_NOT_SUPPORTED_AT_CONNECTOR = -1014321
ERR_CONTROL_NOT_SUPPORTED = -1014320
WARN_CONTROL_NOT_SUPPORTED = 1014320
ERR_MODE_NOT_SUPPORTED = -1014319
WARN_MODE_NOT_SUPPORTED = 1014319
ERR_KEY_NOT_FOUND = -1014318
ERR_NO_SPACE_LEFT = -1014317
ERR_ASSEMBLY_SIZE_NOT_SUPPORTED = -1014316
WARN_CTRL_PEAKMODE_SIGNALCTRL = 1014310
WARN_CTRL_RUNMODE_SIGNALCTRL = 1014309
ERR_CTRL_UNABLE_TO_SET_SIGNALGENERATOR = -1014308
ERR_CTRL_UNABLE_TO_CONTROL_ALARM_UNIT = -1014307
ERR_CTRL_UNABLE_TO_CONTROL_PEAK_UNIT = -1014306
ERR_CTRL_UNABLE_TO_SET_ZERO = -1014305
ERR_CTRL_UNABLE_TO_CONTROL_MATH_UNIT = -1014304
ERR_CTRL_UNABLE_TO_CONTROL_SIGNAL_GENERATOR = -1014303
ERR_CTRL_UNSPECIFIED = -1014302
ERR_WRONG_SECURITY_CODE = -1014301
ERR_CTRL_WRONG_PAR = -1014300
ERR_CALPOINT_TYPE_PROTECTED = -1014299
ERR_CAN_RAW_RECEIVER_ENABLE_DISABLE = -1014285
ERR_CAN_RESOURCE_OVERLOAD = -1014284
ERR_TRANSMISSIONTYPE_SOURCECHANGE_NO_SUPPORT_64_BIT = -1014282
ERR_CAN_WRONG_INDEX = -1014281
ERR_CAN_VALUE_ON_ERROR_TYPE_NOT_SUPPORTED = -1014280
ERR_CAN_VALUE_ON_ERROR_TYPE_NOT_SUPPORTED_BY_DATAFORMAT = -1014279
ERR_CAN_SET_BITRATE = -1014274
WARN_CAN_BITRATE_CHANGED = 1014273
WARN_CAN_BITRATE_SAMPLEPOINTRATIO_CHANGED = 1014272
ERR_CAN_ANY_DECODER_TIMEOUT = -1014271
WARN_CAN_ANY_DECODER_TIMEOUT = 1014271
ERR_CAN_ANY_DECODER_LOSS_OF_SIGNAL = -1014270
WARN_CAN_ANY_DECODER_LOSS_OF_SIGNAL = 1014270
ERR_CAN_MAPPING_MASSAGE_INACTIVE = -1014269
ERR_CAN_MAPPING_SOURCES_OVERLAP = -1014268
WARN_CAN_MAPPING_NO_SOURCE_DEFINED = 1014266
ERR_ONLY_ONE_MAPPINGSOURCE_POSSIBLE = -1014265
WARN_SIGNALLENGTH_CHANGED = 1014261
ERR_CAN_CCP = -1014259
ERR_CAN_WRITE_CANSIGNAL = -1014258
ERR_CAN_TRANSMIT_SIGNALREFERENCE_INVALID = -1014257
ERR_CAN_TRANSMIT_MODULEREFERENCE_NOT_LOCAL = -1014256
ERR_CAN_TRANSMIT_MSG = -1014255
WARN_CAN_TRANSMISSIONTYPE_CHANGED = 1014254
ERR_CAN_TRANSMISSIONTYPE_NOT_SUPPORTED = -1014253
ERR_CAN_CONNECTOR_NOT_SUPPORTED = -1014252
ERR_CAN_TRANSMIT_CHANNEL = -1014250
ERR_CAN_TRANSMIT_USERDEF_MSG = -1014245
```

```
ERR_CAN_TRANSMIT_BYTEORDER_NOT_SUPPORTED = -1014244
ERR_CAN_TRANSMIT_TABLE = -1014243
ERR_CAN_TRANSMIT_BYTEVALUE = -1014242
ERR_CAN_TRANSMIT_BYTECOUNT = -1014241
ERR_CAN_TRANSMITTER_USERDEF_MSG_SET_TABLE = -1014240
ERR_CAN_DECODER_MODELENGTH_64 = -1014238
ERR_CAN_DECODER_MODESTARTBIT_63 = -1014237
ERR_CAN_DECODER_LENGTH_64 = -1014236
ERR_CAN_DECODER_STARTBIT_63 = -1014235
ERR_CAN_COMBINATION_IDENTIFIER_FRAMEFORMAT = -1014234
ERR_CAN_CHANNELNO = -1014233
ERR_CAN_DECODER_MODE_STARTBIT_LENGTH = -1014232
ERR_CAN_COMBINATION_STARTBIT_LENGTH = -1014231
ERR_CAN_DECODER_FORMAT_DOUBLE = -1014230
ERR_CAN_DECODER_FORMAT_FLOAT = -1014229
ERR_CAN_DECODER_FORMAT_32 = -1014228
ERR_CAN_DECODER_RBUFNO_INVALID = -1014227
ERR_CAN_DECODER_RBUFNO_USED_TWICE = -1014226
ERR_CAN_DECODER_GET_TABLE = -1014225
ERR_CAN_DECODER_SET_TABLE = -1014224
ERR_CAN_DECODER_TABLE_FULL = -1014223
ERR_CAN_DECODER_DISABLE = -1014222
ERR_CAN_DECODER_ENABLE = -1014221
ERR_CAN_DECODER_TABLE_NOT_VALID = -1014220
ERR_CAN_DECODER_PARAM_CAN_SIGNAL = -1014219
ERR_CAN_NO_RESPONSE = -1014218
WARN_CAN_RESET_REQUEST = 1014217
ERR_CAN_TRANSMITTER_OVERRUN = -1014216
ERR_CAN_BUS_ERROR_TRANSMITTER = -1014215
ERR_CAN_BUS_WARNING_TRANSMITTER = -1014214
WARN_CAN_BUS_WARNING_TRANSMITTER = 1014214
ERR_CAN_RECEIVER_OVERRUN = -1014213
ERR_CAN_BUS_OFF = -1014212
ERR_CAN_BUS_ERROR_RECEIVER = -1014211
ERR_CAN_BUS_WARNING_RECEIVER = -1014210
WARN_CAN_BUS_WARNING_RECEIVER = 1014210
ERR_CAN_INIT_CANSIGNAL_PARAMS = -1014209
ERR_CAN_GET_BITRATE = -1014208
ERR_CAN_GET_STATE = -1014207
ERR_CAN_INVALID_HANDLE = -1014206
ERR_CAN_INDEX = -1014205
ERR_CAN_IOCTL = -1014204
ERR_CAN_BITRATE_NOT_SUPPORTED = -1014203
ERR_CAN_DISABLE = -1014202
ERR_CAN_ENABLE = -1014201
ERR_CAN_OPENDRV = -1014200
ERR_FBFS_ACCESS_LED = -1014191
ERR_FBFS_ACCESS_CAN = -1014190
ERR_DAQ_SET_CARRIERFREQ_MODE = -1014143
ERR_DAQ_SET_FREEZED_FLAG = -1014142
ERR_DAQ_SET_RTMEASMODE = -1014141
ERR_DAQ_TRIGTIME = -1014140
ERR_DAQ_TRIGGERUNKNOWN = -1014139
ERR_DAQ_ISOSIGNALLIST_VERSION = -1014138
ERR_DAQ_AMP_NULL = -1014137
ERR_DAQ_RTDRIVER_NULL = -1014133
ERR_DAQ_WRONG_CHAN_TYPE = -1014131
WARN_SCALING_ANALOGOUT_GAIN_IF_ANALOGIN_LINTABLE = 1014128
ERR_DAQ_GET_DEMOD_INFO = -1014127
ERR_DAQ_SET_ERRORBUF = -1014125
ERR_DAQ_DRV_SET_PT_COEFFS = -1014124
```

```
ERR_DAQ_DRV_SET_COLDJUNCTION = -1014123
ERR_DAQ_DRV_SET_INITJUST = -1014122
ERR_SCALING_POLYNOM_NOT_SUPPORTED_IF_ANALOGOUT_ENABLED = -1014121
ERR_SCALING_POLYNOM_TOO_MUCH_COEFFS = -1014120
ERR_SCALING_POLYNOM_NOT_ENOUGH_COEFFS = -1014119
ERR_SCALING_POLYNOM_TOO_MUCH_SEGMENTS = -1014118
ERR_SCALING_LINTABLE_TOO_MUCH_POINTS = -1014117
ERR_SCALING_POLYNOM_NOT_ENOUGH_SEGMENTS = -1014116
ERR_ACCESS_RTDRIVER = -1014115
ERR_SCALING_LINTABLE_NOT_ENOUGH_POINTS = -1014114
ERR_DAQ_SCALING_INTERNAL = -1014113
ERR_DAQ_SCALETYPE = -1014112
ERR_WRONG_CHANNEL_INDEX = -1014111
ERR_WRONG_SIG_INDEX = -1014110
ERR_WRONG_CONNECTOR = -1014109
ERR_DAQ_INITJUST_NOTFOUND = -1014108
ERR_DAQ_INITJUST_POINTNR = -1014107
ERR_DAQ_INITJUST_WRONGPAR = -1014106
ERR_DAQ_SIGID_LEN = -1014105
ERR_DAQ_SIGID_UNKNOWN = -1014102
ERR_DAQ_FILTER = -1014100
ERR_DAQ_SIGNAL_NOT_AVAILABLE = -1014099
ERR_DAQ_ADD_FBTYPE = -1014098
ERR_DAQ_ADD_FB_INSTANCE = -1014097
ERR_DAQ_SET_FB_INPUT = -1014096
ERR_DAQ_OUTRATE = -1014095
ERR_DAQ_SET_BUFFER_PARS = -1014094
ERR_DAQ_SET_ZEROVAL = -1014093
ERR_DAQ_FIFO_PARAMS = -1014092
ERR_DAQ_FIFOREADER_CAL_PARAMS = -1014091
ERR_DAQ_SCALING_PARAMS = -1014090
ERR_DAQ_SET_ISO_MODULE_ID = -1014089
ERR_DAQ_SET_ISO_TRANSMIT = -1014088
ERR_DAQ_GET_ISO_TRANSMIT = -1014087
ERR_DAQ_OVERRUN = -1014085
ERR_DAQ_UNDEFINED = -1014082
ERR_DAQ_FIFO_SYNC = -1014070
ERR_DAQ_OVERLOAD = -1014060
ERR_DAQ_PROCESS = -1014061
ERR_DAQ_OCCF = -1014055
ERR_DAQ_LIMSE = -1014050
ERR_DAQ_NOSIGNALSUBSCRIBED = -1014030
ERR_DAQ_BUFFERSIZE = -1014025
ERR_DAQ_AMPOFF = -1014023
ERR_DAQ_OPENRTDRV = -1014010
ERR_DAQ_SUBSCR_RUNNING = -1014006
ERR_DAQ_SUBSCR_UUID = -1014005
ERR_DAQ_SUBSCR_STATE = -1014004
ERR_DAQ_SUBSCRITION = -1014002
ERR_DATA_FORMAT_NOT_SUPPORTED = -1010000
ERR_SRV_STAT_COMPOSER = -1004305
ERR_MTHD_SYS_NORESTARTFKT = -1004260
ERR_MTHD_SYS_WRONGCHANTYPE = -1004242
ERR_MTHD_SYS_WRONGCHAN = -1004240
ERR_MTHD_SYS_INVALIDFWFILE = -1004208
ERR_MTHD_SYS_FILENOTFOUND = -1004201
ERR_OPENFILE = -1004200
ERR_WRITE_FILE = -1004199
ERR_READ_FILE = -1004198
ERR_PARSE_XML_SETTING_REFUSED = -1004160
ERR_PAR_WRONG_TYPE = -1004153
```

```
WARN_PAR_NODEFSET = 1004148
ERR_PAR_VIEWID_INUSE = -1004146
ERR_PAR_READING_VAL = -1004144
WARN_PAR_EMPTY_VAL = 1004143
ERR_PAR_VIEWINVALID = -1004142
ERR_PAR_WRONGXPATHSTATE = -1004140
ERR_PAR_XPATHINVALID = -1004138
ERR_PAR_NOROOT = -1004136
WARN_PAR_PARSING = 1004134
ERR_PAR_PARSING = -1004135
ERR_PAR_INVALIDADDR = -1004118
WARN_PAR_TAGUNKOWN = 1004112
ERR_PAR_TAGUNKOWN = -1004111
ERR_PAR_OUTOFRANGE = -1004110
WARN_PAR_OUTOFRANGE = 1004110
ERR_PAR_INVALIDATTR = -1004109
WARN_PAR_READONLY = 1004108
ERR_PAR_CHOICEINVALID = -1004107
ERR_PAR_CNTXTLOST = -1004106
ERR_PAR_CHOICEVALUESYNTAXINVALID = -1004105
ERR_PAR_RESTOREFAILED = -1004104
ERR_PAR_XMLINVALID = -1004103
ERR_TXTELCONTAINER_TOO_SMALL = -1003001
ERR_PARSE_PROCESSING_DATA = -1002002
ERR_PARSE_INVALID_DATA = -1002001
ERR_UTIL_FILE_RESTORE = -1001140
ERR_UTIL_INVALID_FILEHANDLE = -1001136
ERR_UTIL_STRCONVERT = -1001132
ERR_UTIL_NO_PATTERN_MATCH = -1001131
ERR_EVFLG_TIMEOUT = -1001130
ERR_EVFLG_PTHREADS = -1001128
ERR_UTIL_TIMER_INVALID = -1001125
WARN_UTIL_TIMER_WASRUNNING = 1001025
ERR_UTIL_LANGUAGE = -1001110
ERR_COMPOSER_TAG_NOT_DEFINED = -1001002
ERR_NET_NETWORKADAPTER = -1000680
ERR_UDP_BINDTODEVICE = -1000658
ERR_UDP_BIND = -1000656
ERR_UDP_SETBROADCAST = -1000654
ERR_UDP_SETREUSEADR = -1000652
ERR_UDP_OPENSOCK = -1000650
ERR_TCP_BIND = -1000644
ERR_TCP_LISTEN = -1000642
ERR_TCP_CHECKIP = -1000638
ERR_TCP_GETADR = -1000636
ERR_TCP_GETPORT = -1000635
ERR_TCP_SETNODELAY = -1000632
ERR_TCP_SETNONBLOCK = -1000630
ERR_TCP_OPENSOCK = -1000628
ERR_TCP_CONNECT = -1000626
ERR_TCP_ACCEPT = -1000624
ERR_TCP_CONNECT_FAILED = -1000620
ERR_TCP_IPADR_ALREADY_CONNECTED = -1000615
ERR_TCP_ADR_UNKNOWN = -1000612
ERR_TCP_SOCKET_EINTR = -1000604
ERR_TCP_SOCKET_HUP = -1000603
ERR_TCP_SOCKET_DESCRIPTOR = -1000602
ERR_CON_REC = -1000524
ERR_CON_SENDING = -1000522
ERR_CON_INVALID_TELTYPE = -1000510
ERR_CON_ADRRESINI_FAILED = -1000504
```


現在位置: Supported devices -> Hbm.Api.QuantumX Namespace -> Hbm.Api.QuantumX.Enums Namespace

```
   ERR_CON_CONHNDLR_UUIDUNKNOWN = -1000500
   ERR_TRANS_READINGHDR = -1000430
   ERR_TRANS_NOTFOUND = -1000420
   ERR_TRANS_CANCELED = -1000419
   ERR_TRANS_SEND_REQ_TIMEOUT = -1000416
   ERR_TRANS_ACK_TIMEOUT = -1000417
   ERR_TRANS_REQ_TIMEOUT = -1000418
   ERR_TRANS_INVALID_TELFORMAT = -1000414
   ERR_TRANS_REQUESTOR_WRONGPROTTYPE = -1000404
   ERR_TRANS_RESPONDER_WRONGPROTTYPE = -1000405
   ERR_TRANS_NOT_FOR_US = -1000406
   ERR_NO_MSP_CHKSUM = -1000301
   ERR_SRV_CONTROL = -1000300
   ERR_SRV_NODATA = -1000275
   ERR_SRV_NORESPONSE_YET = -1000271
   ERR_SRV_REQRMTHD_NOREQTEL = -1000265
   ERR_SRV_PARAMERROR = -1000255
   ERR_SRV_MTHDOUT_UNKNOWNMTHD = -1000203
   ERR_SRV_MTHDOUT_WRONGANSW = -1000202
   ERR_SRV_MTHDOUT_TOOMUCH = -1000201
   ERR_SRV_UNKNOWNSRV = -1000107
   ERR_SRV_UNKNOWNMTHD = -1000106
   ERR_INDEX_OUT_OF_RANGE = -1000000
End Enum
```

ファイル

FrameworkError.cs

メンバー

メンバー	説明
INTERNAL_ACTIVATE_PROFINET	内部プロセス(PROFINETパラメータ化)用に予約済み。
ERR_FIELDBUS_INIT_DRIVER	CX27:(歴史的な理由のみ、他のすべてのフィールドバスエラー:TextParAddConn.cppの使用を検討してください!)。
ERR_INVALID_REVISION_STRING	EEPROM構成。
ERR_SCALING_NOT_SUPPORTED	Analog In ++++++++++++++++++++++++++++++ ++++++++++++++++++++++++++++++ #define HNDL_ANALOGIN_ERR_MIN -119999 #define HNDL_ANALOGIN_ERR_MAX -100000 #define HNDL_ANALOGIN_WARN_MIN 100000 #define HNDL_ANALOGIN_WARN_MAX 119999
ERR_SENSORTYPE_NOT_SUPPORTED	基本センサ。
ERR_VOLTAGERANGE_NOT_SUPP	電圧入力。
ERR_CURRENTRANGE_NOT_SUPP	電流入力。
ERR_WIRING_NOT_SUPPORTED	ブリッジセンサ。
ERR_PT_RESISTANCE_NOT_SUPPORTED	PTセンサ。
ERR_TC_TYPE_NOT_SUPPORTED	熱電対。
ERR_RESISTOR_NOT_SUPPORTED	抵抗。
ERR_SSI_CODING_NOT_SUPPORTED	SSIコーディング。
ERR_PRESSURE_RANGE_NOT_SUPPORTED	圧力センサ。
ERR_FREQUENCYRANGE_NOT_SUPPORTED	周波数、入力。
ERR_SENSORSUPPLY_NOT_SUPPORTED	サポートしていないセンサ電源。
ERR_SETZERO_INHIBITED	ゼロバランスは許可されていません/禁止です。
ERR_ANALOG_OUT_SETZERO	アナログ出力。
ERR_PARAMSETTINGS_IND_COUNTER	アナログ入力パラメータ。
ERR_TEDS_NO_SENSOR_TEMPLATE	TEDSの説明にセンサが見つかりません。
ERR_TEDS_NO_DATA_OR_INVALID	TEDS評価時に発生したエラー。
ERR_TEDS_NO_MIN_PHYS_VAL	物理単位での最小測定値はありません。
ERR_TEDS_NO_MAX_PHYS_VAL	物理単位での最大測定値はありません。
ERR_TEDS_NO_MIN_ELEC_VAL	電気単位での最小測定値はありません。
ERR_TEDS_NO_MAX_ELEC_VAL	電気単位での最大測定値はありません。
ERR_TEDS_NO_MDEF_UCC	HBM UCC が定義されていません。
ERR_TEDS_CALCURVE_COEFFS_INVALID	校正情報がありません。
ERR_TEDS_NODATA	利用可能なTEDSバイナリブロックがありません。
ERR_TEDS_NOPARSEBIN	TEDSバイナリブロックの解析に失敗しました。
ERR_TEDS_NOAMP	チャンネルのHALアンプが見つかりませんでした。
ERR_TEDS_NO_EXCITE_AMPL_NOM	ブリッジ供給電圧の公称振幅はありません。
ERR_TEDS_NO_EXCITE_AMPL_MIN	ブリッジ供給電圧の最小振幅はありません。
ERR_TEDS_NO_EXCITE_AMPL_MAX	ブリッジ供給電圧の最大振幅はありません。
ERR_TEDS_NO_SENSOR_IMPED	インピーダンスはありません。
ERR_TEDS_GAUGEFACTOR	Kファクタはありません。
ERR_TEDS_NO_BRIDGETYPE	ブリッジ回路タイプはありません。
ERR_TEDS_PARAM	TEDSパラメータ構成時に発生する一般的なエラー。
ERR_TEDS_NOT_SUPPORTED_BY_CONNECTOR	TEDSデータはコネクタでサポートされていません。
ERR_TEDS_NO_CALTABLE	計算できません。
ERR_TEDS_CALCURVE_POWER_INVALID	ドメインの累乗値が無効です。
ERR_TEDS_CALTABLE_NOT_ENOUGH_POINTS	校正ポイントが足りません(HBMは計算可能な部分として最小値と最大値を計算していないため、少なくとも2ポイントが必要です)。
ERR_TEDS_CALTABLE_TOO_MUCH_POINTS	サポートポイントが多すぎます。
ERR_TEDS_BUFFER_TOO_SMALL	TEDSデータ用のバッファが小さすぎます。
ERR_TEDS_WRITE_NOT_ALLOWED	TEDS(RFID)は書き込めません。

メンバー	説明
ERR_TEDS_TID_DONT_MATCH	TEDS TIDが一致しません。
ERR_TEDS_SCALINGTYPE_NOT_SUPPORTED	スケーリングタイプはサポートされていません。
ERR_SPEEDMODE_CHANGING_NOT_POSSIBLE_SWITCH_OFF	スピードモード。
ERR_PHYSUNIT_NOT_FOUND	物理単位が見つかりません。
ERR_PHYSUNIT_NOT_RELATED	単位は変換できません。
ERR_PHYSUNIT_SCALING_FACTOR_ZERO	ファクタが間違っています(すなわちスケーリング)。
WARN_SUPPLY_VOLTAGE_SWITCHED_OFF	パラメータ化。
WARN_SSI_CLK_FREQ_CHANGED	SSIセンサ。
WARN_FILTER_CUTOFF_FREQ_CHANGED	フィルタ。
ERRHAL_ERROR	一般的なHALエラー。
ERRHAL_OPENMSPDRV	予約済み、4.10までのファームウェアバージョンで使用され、4.11.1以降廃止されました:ERRHAL_HARDWARE_COMMUNICATION_PARAMETERS_INCOMPATIBLE = -120090、MSPを制御するためのドライバを開くことができませんでした。
ERRHAL_OPENCPLDDRV	CPLDを制御するためのドライバを開くことができませんでした。
ERRHAL_GETAMPTEMP	温度の読み取り(チャネルから)に失敗。
ERRHAL_GETAMPTEMPCAL	温度オフセットの決定に失敗。
ERRHAL_ENABLECHN	チャネルのオン時に発生したエラー。
ERRHAL_SETMEASINPUT	測定入力の設定エラー。
ERRHAL_SETSENSCIRCUIT	測定信号の設定エラー。
ERRHAL_SETCALINPUT	校正信号の設定時に発生したエラー。
ERRHAL_SETGAINCAL	ゲイン設定エラー(校正チャネル)。
ERRHAL_SETGAINMEAS	ゲイン設定エラー(測定チャネル)。
ERRHAL_SET_SENSORSUPPLY	センサ電源の設定エラー。
ERRHAL_SETFEEDBACK	TFパワーアンプのフィードバック設定エラー。
ERRHAL_SETEXCIT	ブリッジ電源電圧の設定エラー。
ERRHAL_GETSNAMP	シリアル番号(アンプ)の読み取りエラー。
WARN_HAL_GETSNAMP	シリアル番号(アンプ)の読み取り時に1バイト修正しました。
ERRHAL_SETSNAMP	シリアル番号(アンプ)の書き込みエラー。
ERRHAL_GETHWRAMP	ハードウェアリビジョン(アンプ)の読み取りエラー。
WARN_HAL_GETHWRAMP	ハードウェアリビジョン(アンプ)の読み取り時に1バイト修正しました。
ERRHAL_SETHWRAMP	ハードウェアリビジョン(アンプ)の書き込みエラー。
ERRHAL_GETTID	TIDの判別時に発生したエラー。
ERRHAL_GETTEDSDATHW	ハードウェアからTEDSデータの読み取り時に発生したエラー。
ERRHAL_GETTEDSDAT	TEDSデータの読み取り時に発生したエラー。
ERRHAL_TEDSCHECKSUM	TEDSのチェックサムエラー。
ERRHAL_NOTEDS	利用可能なTEDSはありません。
ERRHAL_TEDS_ALLZERO	TEDSは利用可能ですが、内容はゼロのみです。
ERRHAL_TEDSCHECKSUMINSERT	TEDSバイナリブロックにチェックサムを挿入中に発生したエラー。
ERRHAL_SETTEDSDAT	TEDSデータの書き込みエラー。
ERRHAL_SETTEDSCON	TEDSインタフェース(1 Wire/0 Wire)の設定エラー。
ERRHAL_SIZETEDSDAT	データがTEDSチップに収まりません。
ERRHAL_SETLEDAMP	LED(アンプ)の設定エラー。
ERRHAL_SETSYSLEDAMP	システムLED(MX1609)の設定時に発生したエラー。
ERRHAL_WRSETTING	スイッチの設定時に発生したエラー。
ERRHAL_SETADCCLK	サンプルレートの設定時に発生したエラー。

メンバー	説明
ERRHAL_GETMAXSRAMP	(全チャンネルの)最高サンプリングレートを決定時に発生したエラー。
ERRHAL_SET_FM_CONF	FM Config設定時に発生したエラー。
ERRHAL_SET_SSI_CONF	SSI Config設定時に発生したエラー。
ERRHAL_GETMSPVERAMP	MSPバージョン番号(アンプ)の決定時に発生したエラー。
ERRHAL_GETMSPSTATAMP	MSPステータス(アンプ)の読み取りエラー。
ERRHAL_SYNCADC	ADCの同期中に発生したエラー。
ERRHAL_SETCHNPROP	チャネルプロパティの設定時に発生したエラー。
ERRHAL_GETMSPREG	MSPレジスタの読み取り時に発生したエラー。
ERRHAL_SETMSPREG	MSPレジスタの設定時に発生したエラー。
ERRHAL_SETTSTSIGMEAS	テスト信号(測定チャネル)の設定エラー。
ERRHAL_SETTSTSIGCAL	テスト信号(演算チャネル)の設定エラー。
ERRHAL_SETSIMUVAL	シミュレーション値の設定中に発生したエラー。
ERRHAL_SETSIMUVALFORMAT	サポートしていないシミュレーション値の形式。
ERRHAL_SETSIMUVALTYPE	サポートしていないシミュレーション値の種類。
ERRHAL_GETSNAPSHOT	スナップショット時のエラー(測定/演算値のペア)。
ERRHAL_SNAPSHOT_MAXDATA	スナップショットデータの最大数を超えています。
ERRHAL_CALCPHASE	位相補正の計算時に発生したエラー。
ERRHAL_NO_VALID_SNAPSHOTDATA	有効なスナップショットデータがありません。
ERRHAL_GETCALDATHW	MSPフラッシュから元の校正データの読み取り時に発生したエラー。
ERRHAL_GETCALDAT	属性から元の校正データの読み取り時に発生したエラー。
ERRHAL_SETCALDAT	元の校正データからMSPフラッシュへの書き込み時に発生したエラー。
ERRHAL_SETCSURCAL	元の校正データのチェックサム読み取り時に発生したエラー。
ERRHAL_GETCSURCAL	元の校正データのチェックサム書き込み時に発生したエラー。
ERRHAL_LDRCHKSUM	ローダのチェックサムが正しくありません。
ERRHAL_STORESENSSUPPLYOFFS	可変センサ電源のオフセットをMSPフラッシュへの書き込み時に発生したエラー。
ERRHAL_GETSENSSUPPLYOFFS	MSPフラッシュからの可変センサ電源のオフセットの読み取り時に発生したエラー。
WARN_HAL_GETCSURCAL	元の校正データのチェックサムを読み取り時に修正が行われました。
ERRHAL_SETMEASVALINVAL	測定値を無効としてマークするときに発生したエラー。
WARN_HAL_GETSENSSUPPLYOFFS	MSPフラッシュから可変センサ電源のオフセット読み取り時に、バイトが修正されました。
ERRHAL_SELFPGACHN	FPGAレジスタ構造(MX460)内のコネクタの選択時に発生したエラー。
ERRHAL_REDDATRATE	データレートを下げるための除数が大きすぎます。
ERRHAL_RATE_NOT_SUP_BY_FPGA	出力データレートはFPGAでサポートされていません。
ERRHAL_FB_NOT_AVAILABLE	ファンクションブロックは使用できません。
ERRHAL_FPGATEST_NOT_SUPPORTED	要求した(FPGA)テスト信号はサポートされていません。
WARN_HAL_GETPOTIWIPERFROMMSP	MSPに保存されている(MX403)ポテンショメータ値の読み取り時に修正が行われました。
ERRHAL_GETPOTIWIPERFROMMSP	MSPに保存されている(MX403)ポテンショメータのワイパに有効な値がありません。
ERRHAL_POTIWIPERCOMPARE	ポテンショメータのワイパ値の比較(MX403)に失敗しました(MSPとポテンショメータ本体)。

メンバー	説明
ERRHAL_SETIEPEHP	IEPEブランチでのハイパスの切り替え。
ERRHAL_GETHWIDENTPF	プラットフォーム識別子の読み取り時に発生したエラー。
WARN_HAL_GETHWIDENTPF	プラットフォーム識別子の読み出し時に、1バイト修正する必要がありました。
ERRHAL_SETHWIDENTPF	プラットフォーム識別子の書き込み時に発生したエラー。
ERRHAL_SETADDHWREVPF	追加のハードウェアリビジョンの書き込み時に発生したエラー。
ERRHAL_SETADDHWIDENTPF	追加のプラットフォーム識別子の書き込み時に発生したエラー。
ERRHAL_GETHWRPF	ハードウェアリビジョン(プラットフォーム)の読み取り時に発生したエラー。
ERRHAL_GETADDHWRPF	追加のハードウェアリビジョン(プラットフォーム)の読み取り時に発生したエラー。
ERRHAL_GETADDHWIDENTPF	追加の識別子(プラットフォーム)の読み取り時に発生したエラー。
WARN_HAL_GETHWRPF	ハードウェアリビジョン(プラットフォーム)の読み取り時に1バイト修正されました。
WARN_HAL_GETADDHWRPF	追加のハードウェアリビジョン(プラットフォーム)の読み取り時に1バイト修正されました。
ERRHAL_SETHWRPF	ハードウェアリビジョン(プラットフォーム)の書き込み時に発生したエラー。
WARN_HAL_GETADDHWIDENTPF	追加の識別子(プラットフォーム)の読み取り時に1バイト修正されました。
ERRHAL_GETPLLVOLTAGE	MSPに保存されているPLL電圧(プラットフォーム)の読み取りエラー。
WARN_HAL_GETPLLVOLTAGE	MSPに格納されているPLL電圧(プラットフォーム)の読み取り時に1バイト修正されました。
ERRHAL_SETPLLVOLTAGE	PLL電圧をMSP(プラットフォーム)保存時に発生したエラー。
ERRHAL_SETMACADDR	MACアドレス(プラットフォーム)の書き込みエラー。
ERRHAL_GETMACADDR	MACアドレス(プラットフォーム)の読み取りエラー。
ERRHAL_SETLEDPF	LED(Pf)の設定エラー。
ERRHAL_GETPFTEMP	温度の読み取り(プラットフォームから)に失敗しました。
ERRHAL_GETMODVOL	モジュール電圧の決定時に発生したエラー。
ERRHAL_GETMODCUR	モジュール電流の決定時に発生したエラー。
ERRHAL_GETEXTCUR	入力(供給)する電流の決定時に発生したエラー。
ERRHAL_GETMSPAPPLDATEPF	MSPアプリケーションの日付(プラットフォーム)の判別時に発生したエラー。
ERRHAL_GETMSPLOADERDATEPF	MSPローダの日付(プラットフォーム)を決定できませんでした。
ERRHAL_GETMSPVERPF	MSPバージョン番号(プラットフォーム)の決定時に発生したエラー。
ERRHAL_GETMSPSTATPF	MSPステータス(プラットフォーム)の読み取り時に発生したエラー。
ERRHAL_GETLIMCURR	可変センサ電源の電流制限内にあるチャネル決定時に発生したエラー。
WARN_HAL_GETMSPVERPF	MSPバージョン番号(プラットフォーム)決定時に1バイト修正されました。
ERRHAL_SETTSTSIGGEN	MSPバージョン番号(プラットフォーム)の決定時に発生したエラー。
ERRHAL_STRTDATSTRM	データストリーム開始時に発生したエラー。

メンバー	説明
ERRHAL_STOPDATSTRM	データストリーム停止時に発生したエラー。
ERRHAL_IISDUMDAT	IISダミーデータの設定時に発生したエラー。
ERRHAL_ENIIS	IISインタフェースのアクティブ化時に発生したエラー。
ERRHAL_RSTIIS	IISインタフェースのリセット時に発生したエラー。
ERRHAL_CHKFLHETRY	Flashエントリの妥当性チェック時に発生したエラー。
ERRHAL_GETMAXSRPF	全チャネルの最高サンプリング レート決定時に発生したエラー。
ERRHAL_OPENSNAPSHOTDRV	スナップショットドライバを開くときに発生したエラー。
ERRHAL_CLOSESNAPSHOTDRV	スナップショットドライバを閉じるときに発生したエラー。
ERRHAL_ENCXUPPER	CX上部取り付け基板の電源を切り替え時に発生したエラー。
ERRHAL_SIGNUMBER	信号番号が正しくありません。
ERRHAL_FOSDELAY	FOS遅延のパラメータ化エラー(不明なサンプルレートまたは不正なFOSゲート番号)。
ERRHAL_SETSCALPARS	スケーリング用パラメータ(MX410、MX460)が許容範囲外です。
ERRHAL_COMPENSATION_OFFSET	許容範囲外の測定トランスデューサと計算可能なトランスデューサの相互影響を補償するためのオフセット。
ERRHAL_DACINPUTTYPE	DACの信号ソースが不明です。
ERRHAL_UNKNOWNSYNCCODE	FPGA(Sync Code)でフィルタの出力レートを設定できません。
ERRHAL_ADDFUNCMATH	HBM数学関数のパラメータ化のエラー。
ERRHAL_UNKNOWN_MATH_TYPE	不明な数学単位。
ERRHAL_PEAK_OUTRATE	MX410のピーク値単位の出力レートが正しくありません。
ERRHAL_CX_SYNC_TEST	同期インタフェース(CX)のテストに失敗しました。
ERRHAL_SETZERO_NO_VALID_MEASVAL	有効な測定値がまだ利用できないため、ゼロ調整(測定単位)に失敗しました。
ERRHAL_SET_FILTER_FIR	FIRフィルタのパラメータ化時に発生したエラー。
ERRHAL_PARAM_IS_ZERO	パラメータをnullにできません。
ERRHAL_FIR_TAB_CNT	FIRタブが多すぎるか少なすぎます(部分フィルタ)。
ERRHAL_FIR_TAB_CNT_SUM	許可されているタブの合計数を超えています。
ERRHAL_SETDEMODFACTORS	復調係数の設定時に発生したエラー。
ERRHAL_SETFIRDEMODFILTER	FIR復調フィルタのタップ設定時に発生したエラー。
ERRHAL_SETPOLYCOEFFSTOFPGA	FPGAへの多項式係数の設定時に発生したエラー。
ERRHAL_FPGAOPENDRV	ドライバを開くときに発生したエラー。
ERRHAL_FPGACLOSEDRV	ドライバを閉じるときに発生したエラー。
ERRHAL_FPGABUSTST	データ/アドレスのバステスト(FPGA)エラー。
ERRHAL_FPGAREAD	ドライバからの読み取りエラー(ioctl:read)。
ERRHAL_FPGAWRITE	ドライバへの書き込みエラー(ioctl:write)
ERRHAL_FPGAINITI2C	I2Cインタフェースの初期化時に発生したエラー。
ERRHAL_FPGARSTI2C	I2Cインタフェースのリセット時に発生したエラー。
ERRHAL_FPGARSTHZR	HZRのリセット時に発生したエラー。
ERRHAL_MSPWRINFO	MSP Info Flashの書き込み時に発生したエラー。
ERRHAL_MSPRDINFO	MSP Info Flashからの読み取り時に発生したエラー。
ERRHAL_MSPGETERR	エラーレジスタ(MSPから)の読み取りエラー。
ERRHAL_MSPDOOWRST	OWのリセット時に発生したエラー(MSP 経由)。
ERRHAL_MSPSETOWMODE	MSPでTEDS接続タイプの設定時に発生したエラー。

メンバー	説明
ERRHAL_TEDSCOM_NOT_POSSIBLE	(少なくとも)1つのコネクタがオーバーフローしているため、TEDS通信はできません。
ERRHAL_MSPGETFOWNUM	1Wireバス上の最初のデバイスのシリアル番号の判別時に発生したエラー。
ERRHAL_MSPGETOWDAT	1Wireデバイスからの読み取り時に発生したエラー。
ERRHAL_MSPSETOWDAT	1Wireデバイスへの書き込み時に発生したエラー。
ERRHAL_MSPBEEP	MSPファームウェアのダウンロードエラー。 ERRHAL_MSPFWDWN = -120785、"Beep"コマンドの送信時に発生したエラー。
ERRHAL_MSPPRGBLKV	フラッシュのプログラミング時に発生したエラー(ブロックごとにリードバック)。
ERRHAL_MSPPRGBLK	フラッシュのプログラミング時に発生したエラー(リードバックせずにブロックごと)。
ERRHAL_MSPRDBLK	フラッシュ(ブロックごと)の読み取り時に発生したエラー。
ERRHAL_MSPPRERDBLK	MSPフラッシュからの読み取り準備に失敗しました。
ERRHAL_MSPRDRFID	RFIDデータの読み取り時に発生したエラー。
ERRHAL_MSPWRRFID	RFIDデータの書き込み時に発生したエラー。
ERRHAL_MSPSETLED	LEDの設定時に発生したエラー。
ERRHAL_MSPRD_NO_RFID	RFIDデータの読み取り時に発生したエラー --> 利用可能なRFIDがありません。
ERRHAL_MSPRD_ERR_RFID	RFIDデータの読み取り時に発生したエラー --> RFIDは存在しますが、ヘッダが正しくありません。
ERRHAL_MSP_EEPROM_SIZE	アドレスと長さの組み合わせがEEEPROMの制限を超えています。
ERRHAL_MSPWR_EEPROM	EEPROMへの書き込み時に発生したエラー。
ERRHAL_MSPRD_EEPROM	EEPROMへの読み取り時に発生したエラー。
ERRHAL_MSP_SET_ADC_LPM	MX403、MX410: コンバータのAD7764を低電力モードにできませんでした。
ERRHAL_MSP_SET_VARIANT	MSPプログラムのバリアントを設定できませんでした。
ERRHAL_SPISETDACPLL	PLL(SPIバス)のDAU設定時に発生したエラー。
ERRHAL_SPIOPENDRV	SPIドライバを開くときにエラーが発生しました。
ERRHAL_SPISETSS	SPIバス上のデバイスを選択時に発生したエラー(スレーブ選択)。
ERRHAL_MAX11245_REG_ACCESS	MX11254のコンバータエラー。
WARN_ADJUST_RMS_PERIOD	予約済み。以前のファームウェアバージョン1.17まで使用していました: ERRHAL_COLDJUNCTION = -130520でしたが、バージョン1.21/2.21では -130528 に移動しました。
ERRHAL_LOOP	予約済み。以前のファームウェア バージョン 4.6.12まで使用していました: WARN_ADJUST_IDX_DIV = 130523。
ERRHAL_CORRTEMPERATURE	MX590で温度ドリフトの編集時に発生したエラー。
ERRHAL_COLDJUNCTION	バージョン1.21/2.21は-130520から移動しました。
ERRHAL_INTERPOLENABLE	補間フィルタのオン(FPGA内)時に発生したエラー。
ERRHAL_INTERPOLINIT	補間フィルタの初期化時に発生したエラー。
ERRHAL_INTERPOLSETDATARATE	補間フィルタのレート設定時に発生したエラー。
ERRHAL_INTERPOLSETSKALINGOVERFLOW	補間フィルタのスケーリングでオーバーフローが発生しました。
ERRHAL_INTERPOLSETSCALING	補間フィルタのスケーリング時に発生したエラー。
ERRHAL_SCALDACNULL	アナログ出力のスケーリング: ゼロオフセットが範囲外。
ERRHAL_SCALDACGAIN	アナログ出力のスケーリング: ゲインが範囲外。

メンバー	説明
ERRHAL_SCALDACNULL_MX410	アナログ出力のスケーリング: ゼロオフセットが範囲外(MX410)。
WARNHAL_SCALDACNULL_MX410	アナログ出力のスケーリング: 調整可能な範囲外のゼロ オフセット(MX410)。
ERRHAL_SCALDACGAIN_MX410	アナログ出力のスケーリング: ゲインが範囲外(MX410)。
WARNHAL_SCALDACGAIN_MX410	アナログ出力のスケーリング: ゲインが調整可能な範囲外(MX410)。
WARN_CONFIGURE_FIREWIRE_FUNCTIONBLOCK	ファンクションブロックドライバ。
ERR_RT_IOCTL	ヒント: 利用可能なテキストはありません。
ERR_RT_COPY_TO_USER	ヒント: 利用可能なテキストはありません。
ERR_RT_COPY_FROM_USER	ヒント: 利用可能なテキストはありません。
ERR_TEDS_PROPERTY_NOT_FOUND	プロパティは見つかりませんでした。
ERR_TEDS_DATA_CORRUPT	予約済み。4.4までのファームウェアバージョンで使用していましたが、4.5.3以降では廃止されました: 以前は: ERR_TEDS_INTERN_MEMORY = -160003、以前は: ERR_TEDS_MODEL_FILE = -160004、以前は: ERR_TEDS_FILE_IDENTIFICATION = -160005、以前は: ERR_TEDS_NO_UNITS = -160006、以前は: ERR_TEDS_FILE_READ = -160007、以前は: ERR_TEDS_TDL_FILE_NOT_FOUND = -160008、以前は: ERR_TEDS_FILE_NOT_OPEN = -160009。バイナリブロックが破損しています。
ERR_TEDS_NO_DISCSIGCONFIG	予約済み。4.4までのファームウェアバージョンで使用していましたが、4.5.3以降は廃止されました: 以前は: ERR_TEDS_NOCREATE_OBJECT = -160011、以前は: ERR_TEDS_TDL_OR_TEMPLATE_NOT_FOUND = -160012、以前は: ERR_TEDS_BITPOS_IS_OUT_OF_RANGE_MEMORY = -160014、以前は: ERR_TEDS_TEMPLATE_NOT_FOUND = -160016。
ERR_TYPE_NOT_SUPPORTED	controls ++ #define HNDL_CONTROL_ERR_MIN -165999 #define HNDL_CONTROL_ERR_MAX -165000 #define HNDL_CONTROL_WARN_MIN 165000 #define HNDL_CONTROL_WARN_MAX 165999
ERR_CAN_DECODER_MODESTARTBIT_511	CAN。
ERR_ETHERCAT_PARAMETERS	EtherCATパラメータエラー。
WARN_ANALOG_OUT_GUARD_TIMEOUT	アナログ出力。
WARN_SET_DIGITAL_IO_IS_INPUT	バージョン1.22/2.22は130526から移動しました。
WARN_SET_DIGITAL_IO_IS_CONNECTED	バージョン1.22/2.22は130527から移動しました。
ERR_MASK_RESULT_LONGER_THAN_64_BITS	CAN。
WARN_TIMER_SET_TO_MINIMUM_POSSIBLE	バージョン4.1は1014264から移動しました。
ERR_MATHTYPE_NOT_SUPPORTED	additional functions ++ #define HNDL_ADDFUNC_ERR_MIN -189999 #define HNDL_ADDFUNC_ERR_MAX -180000 #define HNDL_ADDFUNC_WARN_MIN 180000 #define HNDL_ADDFUNC_WARN_MAX 189999
ERR_TESTSIGNAL_MODE_NOT_SUPPORTED	テスト信号。

メンバー	説明
WARN_MODULE_NOT_READY_FOR_FIELDBUS_ACQ	Error codes known by HBM_QX_Framework.dll / DSS_Frame ++++++++++++++++++++++++++++ ++++++++++++++++++++++++++++++ #define HNDL_FRAME_ERR_MIN -1099999 #define HNDL_FRAME_ERR_MAX -1000000 #define HNDL_FRAME_WARN_MIN 1000000 #define HNDL_FRAME_WARN_MAX 1099999
ERR_ECAT_MODULE_STATUS_NOT_READABLE	最大ISO制限の調整。
WARN_ECAT_CORRECTED_FW_SIGNALE_LIST_SIZE	最大ISO制限の調整。
ERR_ECAT_COUNT_OF_FW_SIGNALS_LIMIT_REACHED	最大ISO制限到達時に発生したエラー。
ERR_NTP_SERVER_NOT_REACHED	NTPサーバ到達時に発生したエラー。
ERR_FW_NO_NEC_CHANNEL	NECチャネル番号が正しくありません。
ERR_FW_NO_NEC_DOWNLOAD	NECのダウンロードに失敗しました!
ERR_FW_NO_MSP_CHANNEL	MSPチャネル番号が正しくありません。
ERR_MSP_DATA_CORRUPT	MSPの調整値が破損しています。
ERR_INITJUST_FROM_MSP	元の調整値はMSPから読み込まれました。
ERR_NO_VALID_INITJUST	有効な元の調整値がありません(MSP)。
ERR_STORING_INITJUST	元の調整値を保存できませんでした。
ERR_GETTING_INITJUST	元の調整値を読み取れませんでした。
ERR_PLLVOL_OUTOFRANGE	MSPが読み取ったPll電圧は有効範囲外です。
ERR_PLLVOL_CORRUPT	MSPが読み取ったPll電圧は破損しています。
ERR_PLLVOL_FROM_MSP	Pll電圧はMSPによってロードされました。
ERR_SYS_FILENAME	ファイル名が長すぎます。
ERR_SYS_BAD_DELIVERY	不正な転送パラメータ。例: "Null Pointer"
ERR_SYS_FRAMEWORK_START	フレームワークが開始されていません。
ERR_SYS_OPEN_CLOSE_WD	ウォッチドッグを開けませんでした。
ERR_SYS_INVALID_TIME	無効な時刻です。
ERR_SPI_OPEN_DRIVER	SPIドライバを開くことができませんでした。
ERR_SPI_WAKEUP_DAC	DACの起動に失敗しました。
ERR_SPI_WRITE_DRIVER_DIGITS	ドライバに設定する数字を書き込めません。
ERR_SPI_WRITE_DRIVER_COMMAND	ドライバにコマンドを書き込めません。
ERR_SPI_VOLT_OUTOF_RANGE	電圧が許容範囲外です。
ERR_SPI_PLLDAC	PLL-DACの設定エラー。
ERR_SYNC_OPEN_DRIVER	同期ドライバを開くとき発生したファイルエラー。
ERR_SYNC_ANY_TSD_PROBLEM	タイムソースのデーモンエラー。
ERR_SYNC_NOT_ALL_MODULES_READY	タイムソースデータが無効です。
ERR_SYNC_TIMESOURCE_DATA_NOT_VALID	タイムソースデータが無効です。
ERR_SYNC_ACTIVATE_ETHERCAT	タイムソースEtherCATアクティブ化の問題。
ERR_SYNC_ACTIVATE_IRIG	タイムソースIRIGアクティブ化の問題。
ERR_SYNC_ACTIVATE_PTP	タイムソースPTPアクティブ化の問題。
ERR_SYNC_ACTIVATE_NTP	タイムソースNTPアクティブ化の問題。
ERR_SYNC_ACTIVATE_AUTO	タイムソースAUTOアクティブ化の問題。
ERR_PARAM_IP_ADDRESS_SYNTAX	IPアドレスの構文エラーです。
ERR_PARAM_SYNC_TIMESOURCE_NOT_SUPPORTED	選択したタイムソースパラメータはサポートされていません。
ERR_SYNC_CYCLEDIFF_TOO_LARGE	マスタとスレーブ間のサイクル数が長時間にわたって異なりすぎています。
ERR_SYNC_NOT_IN_SYNC	デバイスはマスタと同期していません。
ERR_ADJ_MEASRANGE_INVALID	測定範囲は自動調整アルゴリズムでは無効です。
ERR_ADJ_MEASRANGE_OVERRUN	自動調整アルゴリズムの測定値は範囲外です。
ERR_ADJ_WRONG_CONNECTOR	自動調整: コネクタ番号が正しくありません。
ERR_CONN_WRONG_CONNECTOR	コネクタ情報: コネクタ番号が正しくありません。
ERR_ANALOGOUT_NOCONN	信号入力のオーバードライブなどによりアナログ出力が制限されます。
ERR_ANALOGOUT_SIGOVFL	信号入力のオーバードライブなどによりアナログ出力が制限されます。

メンバー	説明
ERR_ANALOGOUT_OCCF	アナログ出力が短絡しました。
ERR_ANALOGOUT_OVFL	アナログ出力はスケーリング等により制限があります。
ERR_SIGNALREFERENCE_UNKNOWN	入力リファレンス 。
ERR_PARAM_LEVELS_IDENTICAL	OnLevelとOffLevelは同一です。
ERR_SIGNALREFERENCE_CONNECTOR_NOT_SUPPORTED	入力リファレンス 。
ERR_PINFO_NULL	pInfoがNULLです(HALが初期化されていません)。
ERR_PARAMSETTINGS_NULL	パラメータ化: 有効なパラメータ化がまだ設定されていないためpParamSettingはNULLです。
ERR_PARAM_CALCSYNCSIGNAL	パラメータ化: SyncSignalParamsの編集時に発生したエラー。
ERR_PPLATFORM_NULL	pPlatformがNULLです(HALが初期化されていません)。
ERR_PAMP_NULL	pAmpがNULLです(HALが初期化されていません)。
ERR_PANALOG_NULL	pAnalogがNULLです(HALが初期化されていません)。
ERR_TEDS_DATA	TEDSデータをエラーなしで読み取れなかった場合に設定されます。TEDSが接続されていない可能性があります。
WARN_CTRL_PEAKMODE_SIGNALCTRL	警告: ピークモードは内部信号によって制御されるため、制御は実行されません。
WARN_CTRL_RUNMODE_SIGNALCTRL	警告: 実行モードは内部信号によって制御されているため、制御は実行されません。
ERR_CTRL_UNABLE_TO_SET_SIGNALGENERATOR	信号発生器を設定できません。
ERR_CTRL_UNABLE_TO_CONTROL_ALARM_UNIT	警報器の制御が行えません。
ERR_CTRL_UNABLE_TO_CONTROL_PEAK_UNIT	ピークユニットの制御が行えません。
ERR_CTRL_UNABLE_TO_SET_ZERO	ゼロ調整が行えません。
ERR_CTRL_UNABLE_TO_CONTROL_MATH_UNIT	数学ユニットの制御が行えません(ファンクションブロックがみつかりません)。
ERR_CTRL_UNABLE_TO_CONTROL_SIGNAL_GENERATOR	信号発生器の制御が行えません(ファンクションブロックがみつかりません)。
ERR_CTRL_UNSPECIFIED	一般的な制御エラー。
ERR_CTRL_WRONG_PAR	制御パラメータエラー。
WARN_SIGNALLENGTH_CHANGED	予約済み。ファームウェアバージョン4.0まで使用していました: WARN_TIMER_SET_TO_MINIMUM_POSSIBLE = 1014264でしたが、バージョン4.1では179945に移動しました。
ERR_CAN_TRANSMIT_MSG	CANの送信パラメータが無効な場合の一般的なエラー。
ERR_CAN_CONNECTOR_NOT_SUPPORTED	コネクタ番号はCANをサポートしていません。
ERR_CAN_TRANSMIT_CHANNEL	CAN送信チャネルがパラメータ化されていないか、アクティブ化されていません。
ERR_CAN_TRANSMIT_USERDEF_MSG	ユーザCANメッセージを直接送信できませんでした。
ERR_CAN_TRANSMIT_BYTEORDER_NOT_SUPPORTED	CAN送信テーブルのバイトオーダーはサポートされていません。
ERR_CAN_TRANSMIT_TABLE	CAN送信テーブルが正しくありません。
ERR_CAN_TRANSMIT_BYTEVALUE	CAN送信テーブルの値がバイト単位に収まりません(255を超えています)。
ERR_CAN_TRANSMIT_BYTECOUNT	CAN送信テーブルの送信バイト数が大きすぎます。
ERR_CAN_TRANSMITTER_USERDEF_MSG_SET_TABLE	新しいCAN送信テーブルをioctl経由で設定できませんでした。
ERR_CAN_DECODER_MODELENGTH_64	モード長 > 64。
ERR_CAN_DECODER_MODESTARTBIT_63	モード開始ビット > 63。

メンバー	説明
ERR_CAN_DECODER_LENGTH_64	長さ > 64。
ERR_CAN_DECODER_STARTBIT_63	開始ビット > 63。
ERR_CAN_COMBINATION_IDENTIFIER_FRAMEFORMAT	CAN IDが大きすぎます。
ERR_CAN_CHANNELNO	CANチャネル番号が利用できません。
ERR_CAN_DECODER_MODE_STARTBIT_LENGTH	CANデコーダテーブルを検証したところ、モード開始ビットとモード長の合計が無効であることが判明しました。
ERR_CAN_COMBINATION_STARTBIT_LENGTH	CANデコーダテーブルを検証したところ、開始ビットと長さの合計が無効であることが判明しました。
ERR_CAN_DECODER_FORMAT_DOUBLE	CANデコーダテーブルの検証したところ、生データは倍精度として扱う必要があり、評価される生データの長さが64に等しくありません。
ERR_CAN_DECODER_FORMAT_FLOAT	CANデコーダテーブルを検証したところ、生データは浮動小数点として扱う必要があり、評価される生データの長さが32に等しくありません。
ERR_CAN_DECODER_FORMAT_32	CANデコーダテーブルを検証したところ、以下の条件が満たされていないことがわかりました。評価される生データの長さが32より大きい場合、生データのデータ型と信号値は同じでなければなりません。
ERR_CAN_DECODER_RBUFNO_INVALID	CANデコーダテーブルを検証したところ、信号リングバッファへのインデックスが有効範囲外であることがわかりました。
ERR_CAN_DECODER_RBUFNO_USED_TWICE	CANデコーダテーブルを検証したところ、信号リングバッファを2回使用する必要があることが判明しました。
ERR_CAN_DECODER_GET_TABLE	現在のCANデコーダテーブルをioctl経由で取得できませんでした。
ERR_CAN_DECODER_SET_TABLE	新しいCANデコーダテーブルをioctl経由で設定できませんでした。
ERR_CAN_DECODER_TABLE_FULL	CANデコーダテーブルがいっぱいです。
ERR_CAN_DECODER_DISABLE	CANデコーダを無効化できませんでした。
ERR_CAN_DECODER_ENABLE	CANデコーダを有効化できませんでした。
ERR_CAN_DECODER_TABLE_NOT_VALID	デコーダ規則が無効なCANデコーダテーブル。
ERR_CAN_DECODER_PARAM_CAN_SIGNAL	CAN信号のパラメータ化時に発生したエラー。
ERR_CAN_NO_RESPONSE	CANコントローラから応答がありません。
WARN_CAN_RESET_REQUEST	CANコントローラがリセットを要求。
ERR_CAN_TRANSMITTER_OVERRUN	CANオーバーラン送信コントローラまたは送信リングバッファ。
ERR_CAN_BUS_ERROR_TRANSMITTER	送信中にCAN Busエラーが発生しました。
ERR_CAN_BUS_WARNING_TRANSMITTER	CAN Bus送信の警告。
ERR_CAN_RECEIVER_OVERRUN	受信コントローラまたは受信リングバッファのオーバーラン。
ERR_CAN_BUS_OFF	がオフです。
ERR_CAN_BUS_ERROR_RECEIVER	CAN Bus受信のエラー。
ERR_CAN_BUS_WARNING_RECEIVER	CAN Bus受信の警告。
ERR_CAN_INIT_CANSIGNAL_PARAMS	CAN信号バッファのパラメータの初期化時に発生したエラー。
ERR_CAN_GET_BITRATE	CANのビットレートをドライバから取得できませんでした。
ERR_CAN_GET_STATE	CANのステータスをドライバから取得できませんでした。
ERR_CAN_INVALID_HANDLE	CANドライバのハンドルが無効、または開いていません。
ERR_CAN_INDEX	CANデバイスのインデックスが存在しません。

メンバー	説明
ERR_CAN_IOCTL	IOCTLに関するCANドライバの一般的なエラー。
ERR_CAN_BITRATE_NOT_SUPPORTED	CANビットレートのパラメータはサポートされていません。
ERR_CAN_DISABLE	CANコントローラを無効化できませんでした。
ERR_CAN_ENABLE	CANコントローラを有効化できませんでした。
ERR_CAN_OPENDRV	CANドライバを開けませんでした。
ERR_FBFS_ACCESS_LED	LEDパラメータのFbfsデータの読み取り/書き込みでエラーが発生しました。
ERR_FBFS_ACCESS_CAN	CANパラメータのFbfsデータの読み取り/書き込みでエラーが発生しました。
ERR_DAQ_SET_FREEZED_FLAG	FreezedFlagの設定時に発生したエラー。
ERR_DAQ_SET_RTMEASMODE	RtMeasModeの設定時に発生したエラー。
ERR_DAQ_TRIGTIME	測定の開始点を過ぎています。
ERR_DAQ_ISOSIGNALLIST_VERSION	IsoSignalListのバージョンが一致しません。
ERR_DAQ_AMP_NULL	pAmpへのポインタが0です。
ERR_DAQ_RTDRIVER_NULL	m_pDriverへのポインタがNULLであるため初期化していません。
ERR_DAQ_WRONG_CHAN_TYPE	チャネルタイプが定義されていません。
ERR_DAQ_GET_DEMOD_INFO	復調情報の読み取り時に発生したエラー。
ERR_DAQ_SET_ERRORBUF	エラーバッファの設定時に発生したエラー。
ERR_DAQ_DRV_SET_PT_COEFFS	リアルタイムドライバでのPt要素係数の設定時に発生したエラー。
ERR_DAQ_DRV_SET_COLDJUNCTION	リアルタイムドライバで熱電対ColdJunctionの設定時に発生したエラー。
ERR_DAQ_DRV_SET_INITJUST	リアルタイムドライバでActJust係数の設定時に発生したエラー。
ERR_SCALING_POLYNOM_TOO_MUCH_COEFFS	スケーリング多項式の係数が多すぎます。
ERR_SCALING_POLYNOM_NOT_ENOUGH_COEFFS	スケーリング多項式の係数が少なすぎます
ERR_SCALING_POLYNOM_TOO_MUCH_SEGMENTS	スケーリング多項式のセグメントが多すぎます。
ERR_SCALING_LINTABLE_TOO_MUCH_POINTS	線形テーブルスケーリングのポイントが多すぎます。
ERR_SCALING_POLYNOM_NOT_ENOUGH_SEGMENTS	スケーリング多項式のセグメントが少なすぎます。
ERR_ACCESS_RTDRIVER	RtAnalogドライバへのアクセス時に発生したエラー。
ERR_SCALING_LINTABLE_NOT_ENOUGH_POINTS	線形テーブルスケーリングのポイントが少なすぎます。
ERR_DAQ_SCALING_INTERNAL	内部スケーリングの設定時に発生したエラー。
ERR_DAQ_SCALETYPE	この信号タイプではスケーリング タイプを使用できません。
ERR_WRONG_CHANNEL_INDEX	チャンネル番号が正しくありません。
ERR_WRONG_SIG_INDEX	信号のインデックスが正しくありません。
ERR_WRONG_CONNECTOR	コネクタ番号が正しくありません。
ERR_DAQ_INITJUST_NOTFOUND	元の調整パラメータが見つかりません。
ERR_DAQ_INITJUST_POINTNR	校正ポイントの番号が現在のアンプ設定と一致しません。
ERR_DAQ_INITJUST_WRONGPAR	クロック調整パラメータが無効です。
ERR_DAQ_SIGID_LEN	信号IDが長すぎます。
ERR_DAQ_SIGID_UNKNOWN	信号IDが不明です。
ERR_DAQ_FILTER	フィルタのパラメータ化時に発生したエラー。
ERR_DAQ_SIGNAL_NOT_AVAILABLE	信号が利用できません。
ERR_DAQ_ADD_FBTYPE	ファクションブロックのタイプ設定時に発生したエラー。
ERR_DAQ_ADD_FB_INSTANCE	ファクションブロックのインスタンス作成時に発生したエラー。
ERR_DAQ_SET_FB_INPUT	ファクションブロックの入力設定
ERR_DAQ_OUTRATE	信号の出力レートを選択できません。

メンバー	説明
ERR_DAQ_SET_BUFFER_PARS	リングバッファのパラメータの設定時に発生したエラー。
ERR_DAQ_SET_ZEROVAL	ゼロ値の設定時に発生したエラー。
ERR_DAQ_FIFO_PARAMS	Fiforreaderパラメータの設定時に発生したエラー。
ERR_DAQ_FIFOREADER_CAL_PARAMS	Fiforreader-Calパラメータの設定時に発生したエラー。
ERR_DAQ_SCALING_PARAMS	スケーリングパラメータの設定時に発生したエラー。
ERR_DAQ_SET_ISO_MODULE_ID	アイソクロナス転送のモジュールID設定時に発生したエラー。
ERR_DAQ_SET_ISO_TRANSMIT	アイソクロナス転送の設定時に発生したエラー。
ERR_DAQ_GET_ISO_TRANSMIT	アイソクロナス転送の読み取り時に発生したエラー。
ERR_DAQ_OVERRUN	バッファオーバーフローが発生しました。データが時間内にバッファから読み取られませんでした。
ERR_DAQ_FIFO_SYNC	FIFO同期エラー。
ERR_DAQ_OVERLOAD	過負荷。
ERR_DAQ_PROCESS	制限内の電源。
ERR_DAQ_OCCF	制限内の電源。
ERR_DAQ_LIMSE	トランスデューサの供給には制限があります。
ERR_DAQ_BUFFERSIZE	サービスに転送される測定データのバッファサイズが小さすぎます。
ERR_DAQ_AMPOFF	ソース信号のアンプチャンネルはオフです。
ERR_DAQ_OPENRTDRV	リアルタイム部分のドライバを開くことができませんでした。
ERR_DAQ_SUBSCR_RUNNING	このクライアントはすでに測定を実行しています。
ERR_DAQ_SUBSCR_UUID	このUUIDにはサブスクリプションが作成されていません。
ERR_DAQ_SUBSCR_STATE	サブスクリプションが間違った状態にあります。常にopenSubscription()、addSignal()、およびfinishSubscription()のシーケンスに従う必要があります。
ERR_DAQ_SUBSCRITION	サブスクリプションできません(すでに実行されている測定が多すぎます)。
ERR_SRV_STAT_COMPOSER	エラー サービスステータスコンポーザに問題が発生しました
ERR_MTHD_SYS_NORESTARTFKT	再起動機能は付属していません。
ERR_MTHD_SYS_WRONGCHANTYPE	チャンネルタイプが正しくありません。
ERR_MTHD_SYS_WRONGCHAN	チャンネル番号が正しくありません。
ERR_MTHD_SYS_FILENOTFOUND	ファイルが見つかりません 。
ERR_PAR_WRONG_TYPE	予期しないパラメータタイプです。
WARN_PAR_NODEFSET	このビューにはデフォルト設定がありません。
ERR_PAR_VIEWID_INUSE	ParameterViewのIDはすでに存在しています。
ERR_PAR_READING_VAL	パラメータ値の読み取り時に発生したエラー。
ERR_PAR_VIEWINVALID	不明なビューです。
ERR_PAR_WRONGXPATHSTATE	コンテキストハンドラのXpath-Stateが正しくありません。
ERR_PAR_XPATHINVALID	Xpathのエラー。
WARN_PAR_PARSING	XMLパラメータの解析時に警告が生成されました。
ERR_PAR_PARSING	XMLパラメータの解析時にエラーが生成されました。
ERR_PAR_INVALIDADDR	マルチプレクサのアドレスが正しくありません。
WARN_PAR_TAGUNKOWN	このコンテキストでタグ名が不明です。
ERR_PAR_TAGUNKOWN	このコンテキストでタグ名が不明です。
ERR_PAR_OUTOFRANGE	パラメータ値が有効範囲外です。
ERR_PAR_INVALIDATTR	属性が正しくありません。
ERR_PAR_CHOICEINVALID	選択したパラメータの値が正しくありません。

メンバー	説明
ERR_PAR_CNTXTLOST	XMLパラメータ化でエラーが発生し、コンテキストが失われました。
ERR_PAR_RESTOREFAILED	パラメータ化にエラーが含まれているため拒否され、元のパラメータを復元できませんでした。
ERR_PAR_XMLINVALID	パラメータ化にエラーが含まれているため拒否されました。
ERR_TXTELCONTAINER_TOO_SMALL	TxContainerが転送されるデータ量に対して小さすぎます。
ERR_PARSE_PROCESSING_DATA	eXpatパーサが解釈できませんでした。
ERR_PARSE_INVALID_DATA	eXpatパーサに無効なデータがあります。
ERR_UTIL_FILE_RESTORE	ファイルをフラッシュから復元できませんでした。
ERR_UTIL_INVALID_FILEHANDLE	FileItemのハンドルが無効です。
ERR_UTIL_STRCONVERT	文字列変換エラー。
ERR_UTIL_NO_PATTERN_MATCH	文字列がパターンと一致しません。
ERR_EVFLG_TIMEOUT	EvenFlagがタイムアウトを返すまで待ちます。
ERR_EVFLG_PTHREADS	非推奨pthreadsのEventFlagを使用してエラーが発生しました。バージョン>= 1.22/2.22ではサポートしていません。互換性の理由から残されています。
ERR_UTIL_TIMER_INVALID	間違ったハンドラを使用してタイマにアクセスしました。
WARN_UTIL_TIMER_WASRUNNING	未登録のタイマが実行されました。
ERR_UTIL_LANGUAGE	目的の言語がモジュールでサポートされていません。
ERR_COMPOSER_TAG_NOT_DEFINED	XMLコンポーザ: TAGが定義されていません。
ERR_NET_NETWORKADAPTER	ネットワークアダプタが見つかりません。
ERR_UDP_BINDTODEVICE	バインドエラー。
ERR_UDP_BIND	バインドエラー。
ERR_UDP_SETBROADCAST	setsockoptブロードキャストでエラー。
eraERR_UDP_SETREUSEADR	setsockoptのreuseaddrエラー。
ERR_UDP_OPENSOCK	socket()エラー。
ERR_TCP_BIND	バインドエラー。
ERR_TCP_LISTEN	バインドエラー。
ERR_TCP_CHECKIP	checkIpAdrrエラー。
ERR_TCP_GETADR	getadrエラー。
ERR_TCP_GETPORT	getPortエラー。
ERR_TCP_SETNODELAY	setsockopt NODELAYエラー。
ERR_TCP_SETNONBLOCK	sockopt nonblockinエラー。
ERR_TCP_OPENSOCK	socket()エラー。
ERR_TCP_CONNECT	TCPソケット接続エラー。
ERR_TCP_ACCEPT	TCPソケットAcceptエラー。
ERR_TCP_CONNECT_FAILED	エラー: TCP参加者への接続に失敗しました。
ERR_TCP_IPADR_ALREADY_CONNECTED	エラー: 接続済みの非推奨IPアドレスに接続しようとしています。バージョン>= 1.22/2.22ではサポートしていません。互換性の理由から残されています。
ERR_TCP_ADR_UNKNOWN	エラー: TCPアドレスは非推奨です。バージョン>= 1.22/2.22ではサポートしていません。互換性の理由から残されています。
ERR_TCP_SOCKET_DESCRIPTOR	ソケットは接続済みです。
ERR_CON_REC	インタフェースの受信エラー。
ERR_CON_SENDING	インタフェースの送信エラー。
ERR_CON_INVALID_TELTYPE	ConnectionHandler: 無効なTelegramタイプです。
ERR_CON_ADRRESINI_FAILED	初期のアドレス解決に失敗しました。
ERR_CON_CONHNDLR_UUIDUNKNOWN	不明なUUIDが渡されました。
ERR_TRANS_READINGHDR	トランザクションエラー RxTelContainerのヘッダ読み取りエラー。
ERR_TRANS_NOTFOUND	トランザクションが見つかりません。

メンバー	説明
ERR_TRANS_SEND_REQ_TIMEOUT	送信リクエストのタイムアウト。セグメント化したリクエストをデバイスに送信中にタイムアウトしました。
ERR_TRANS_ACK_TIMEOUT	トランザクション確認のタイムアウト。デバイスは時間内(3秒)にリクエストを確認できませんでした。
ERR_TRANS_REQ_TIMEOUT	トランザクション応答のタイムアウト。デバイスはリクエストを確認しましたが、時間内に応答の送信を開始しませんでした。
ERR_TRANS_INVALID_TELFORMAT	電文の形式が正しくありません。
ERR_TRANS_REQUESTOR_WRONGPROTTYPE	トランザクション要求が無効なプロトコルを受信しました。
ERR_TRANS_RESPONDER_WRONGPROTTYPE	トランザクション応答が無効なプロトコルを受信しました。
ERR_NO_MSP_CHKSUM	MSPにJustFactsチェックサムが保存されていないため、校正中にControl経由でアクセスされていません。
ERR_SRV_CONTROL	Controlが不明です。
ERR_SRV_NODATA	データがありません 。
ERR_SRV_REQRMTHD_NOREQTEL	送信リクエスト電文はありません。
ERR_SRV_PARAMERROR	転送したパラメータは正しくありません。
ERR_SRV_MTHDOUT_UNKNOWNMTHD	応答に不明なコマンドが含まれています。
ERR_SRV_MTHDOUT_WRONGANSW	受信した応答が送信したコマンドと一致しません。SrvまたはMTHDが正しくありません。
ERR_SRV_MTHDOUT_TOOMUCH	受信データ(送信コマンド)用のデータバッファが小さすぎます。
ERR_SRV_UNKNOWNSRV	clCmdWorkerエラー: 不明なサービスです。
ERR_SRV_UNKNOWNMTHD	clMTHDWorkerエラー: 不明なコマンドです。
ERR_INDEX_OUT_OF_RANGE	無効なインデックスです。

Hbm.Api.QuantumX.Enums.LedBlinking Enumeration

モジュールまたはコネクタLEDを点滅させるLED動作モード。

C++

```
enum LedBlinking : public int {
  Green = 1,
  Orange = 2,
  Red = 3
};
```

C#

```
public enum LedBlinking : int {
  Green = 1,
  Orange = 2,
  Red = 3
}
```

Visual Basic

```
Public Enum LedBlinking
Inherits int
  Green = 1
  Orange = 2
  Red = 3
End Enum
```

ファイル

LedBlinking.cs

メンバー

メンバー	説明
Green	緑色の点滅。
Orange	橙色の点滅。
Red	赤色の点滅。

備考

列挙型の値は、Hbm.Api.QuantumX.Core.FrameworkDllWrapper (HBM_XCTRL_SetLedMode)のHbm.Api.QuantumX.Core.FrameworkDllWrapper.SetLedModeのledModeパラメータ値と一致する必要があります。

点滅をオフにする値は、この列挙体には含まれません(オフは点滅ではないため)。代わりに、定数Hbm.Api.QuantumX.Core.FrameworkDllWrapper.DISABLE_LED_BLINKINGとして定義されます。

Hbm.Api.QuantumX.Enums.LedMode Enumeration

FlashLEDのLEDモード。

C++

```
enum LedMode : public int {
  Normal = 0,
  FlashGreen,
  FlashOrange,
  FlashRed
};
```

C#

```
internal enum LedMode {
  Normal = 0,
  FlashGreen,
  FlashOrange,
  FlashRed
}
```

Visual Basic

```
Friend Enum LedMode
  Normal = 0
  FlashGreen
  FlashOrange
  FlashRed
End Enum
```

ファイル

LedMode.cs

メンバー

メンバー	説明
Normal	LEDモードを通常にリセットします。
Green	指定したLED緑色に点滅させます。
Orange	指定したLED橙色に点滅させます。
Red	指定したLED赤色に点滅させます。

Hbm.Api.QuantumX.Enums.LowpassFilterCharacteristic Enumeration

サポートしているQuantumXのローパスフィルタのフィルタ特性。

C++

```
enum LowpassFilterCharacteristic : public int {
  Bessel  = 1,
  Butterworth = 2,
  LinearPhase = 3,
  ButterworthFir = 4
};
```

C#

```
internal enum LowpassFilterCharacteristic : int {
  Bessel  = 1,
  Butterworth = 2,
  LinearPhase = 3,
  ButterworthFir = 4
}
```

Visual Basic

```
Friend Enum LowpassFilterCharacteristic
Inherits int
  Bessel  = 1
  Butterworth = 2
  LinearPhase = 3
  ButterworthFir = 4
End Enum
```

ファイル

LowpassFilterCharacteristic.cs

メンバー

メンバー	説明
Bessel	ベッセルIIRローパスフィルタ。
Butterworth	バターワースIIRローパスフィルタ。
LinearPhase	線形位相FIRローパスフィルタ。
ButterworthFir	バターワースFIRローパスフィルタ。

備考

列挙型の値は、デバイスのxmlによって返されるフィルタ特性番号と同じです(例: Besselの場合は`<FiltCharact>1</FiltCharact>`)。

Hbm.Api.QuantumX.Enums.ModuleStatusId Enumeration

モジュールのステータスIDを指定します。

C++

```
[Flags]
enum ModuleStatusId : public uint {
  StatusConnected = 0x00000001,
  StatusParamAdjustFinishedAfterBoot = 0x00000002,
  StatusTeds = 0x00000008,
  StatusBitMeasuringValueInvalid = 0x00000010,
  StatusBitMeasuringValueInOverflow = 0x00000020,
  StatusParamBusy = 0x00000040,
  StatusAdjustBusy = 0x00000080,
  StatusAnalogInSimulationActive = 0x00000100,
  StatusAutoAdjustAutoCalibrateInhibited = 0x00000400,
  StatusBitControlResetChargeAmplifier = 0x00000800,
  StatusBitControlExtCalSignal = 0x00001000,
  StatusBitControlShuntCalibration = 0x00002000,
  StatusBitControlShuntCalibration2 = 0x00004000,
  StatusIdBitsConnector = 0x00008000,
  StatusBitTedsPresent = 0x00010000,
  StatusTime = 0x00040000,
  StatusModuleInSync = 0x00080000,
  StatusModuleBusy = 0x00400000
};
```

C#

```
[Flags]
internal enum ModuleStatusId : uint {
  StatusConnected = 0x00000001,
  StatusParamAdjustFinishedAfterBoot = 0x00000002,
  StatusTeds = 0x00000008,
  StatusBitMeasuringValueInvalid = 0x00000010,
  StatusBitMeasuringValueInOverflow = 0x00000020,
  StatusParamBusy = 0x00000040,
  StatusAdjustBusy = 0x00000080,
  StatusAnalogInSimulationActive = 0x00000100,
  StatusAutoAdjustAutoCalibrateInhibited = 0x00000400,
  StatusBitControlResetChargeAmplifier = 0x00000800,
  StatusBitControlExtCalSignal = 0x00001000,
  StatusBitControlShuntCalibration = 0x00002000,
  StatusBitControlShuntCalibration2 = 0x00004000,
  StatusIdBitsConnector = 0x00008000,
  StatusBitTedsPresent = 0x00010000,
  StatusTime = 0x00040000,
  StatusModuleInSync = 0x00080000,
  StatusModuleBusy = 0x00400000
}
```

Visual Basic

```
<Flags>
Friend Enum ModuleStatusId
Inherits uint
  StatusConnected = 0x00000001
  StatusParamAdjustFinishedAfterBoot = 0x00000002
  StatusTeds = 0x00000008
  StatusBitMeasuringValueInvalid = 0x00000010
  StatusBitMeasuringValueInOverflow = 0x00000020
  StatusParamBusy = 0x00000040
  StatusAdjustBusy = 0x00000080
  StatusAnalogInSimulationActive = 0x00000100
  StatusAutoAdjustAutoCalibrateInhibited = 0x00000400
  StatusBitControlResetChargeAmplifier = 0x00000800
  StatusBitControlExtCalSignal = 0x00001000
  StatusBitControlShuntCalibration = 0x00002000
  StatusBitControlShuntCalibration2 = 0x00004000
  StatusIdBitsConnector = 0x00008000
  StatusBitTedsPresent = 0x00010000
  StatusTime = 0x00040000
  StatusModuleInSync = 0x00080000
  StatusModuleBusy = 0x00400000
End Enum
```

ファイル

ModuleStatusId.cs

メンバー

メンバー	説明
StatusConnected	トランスデューサが接続されているかどうかを示す情報(値=1または0、またはSTATUS_ID_BITS_CONNECTORのビット)。
StatusParamAdjustFinishedAfterBoot	起動後にすべてのコネクタがパラメータ化されているかどうかを示す情報(値=1または0、またはSTATUS_ID_BITS_CONNECTORのビット)。
StatusTeds	TEDSチップに関する情報。
StatusBitMeasuringValueInvalid	測定値が有効かどうかを示す情報(値=1または0、またはSTATUS_ID_BITS_CONNECTORのビット)。
StatusBitMeasuringValueInOverflow	測定値がオーバーフローしているかどうかを示す情報(値=1または0、またはSTATUS_ID_BITS_CONNECTORのビット)。
StatusParamBusy	アナログ入力のXMLパラメータ化が進行中かどうかを示す情報(値=1または0、またはSTATUS_ID_BITS_CONNECTORのビット)。
StatusAdjustBusy	センサパラメータを接続または変更した後、調整が進行中かどうかを示す情報(STATUS_ID_BITS_CONNECTORのビット)。
StatusAnalogInSimulationActive	アナログ入力でテスト信号がアクティブかどうかを示す情報(STATUS_ID_BITS_CONNECTORのビット)。
StatusAutoAdjustAutoCalibrateInhibited	アナログ出力でテスト信号がアクティブかどうかを示す情報(STATUS_ID_BITS_CONNECTORのビット)。 自動調整または自動校正が一時的に無効かどうかを示す情報 (STATUS_ID_BITS_CONNECTORのビット)。
StatusBitControlResetChargeAmplifier	コントロールResetChargeAmplifierが有効かどうかを示す情報(STATUS_ID_BITS_CONNECTORのビット)。
StatusBitControlExtCalSignal	コントロールExtCalSignalが有効かどうかを示す情報(STATUS_ID_BITS_CONNECTORのビット)。
StatusBitControlShuntCalibration	コントロールShuntCalibrationが有効かどうかを示す情報(STATUS_ID_BITS_CONNECTORのビット)。

StatusBitControlShuntCalibration2	コントロールShuntCalibrationが有効かどうかを示す情報(STATUS_ID_BITS_CONNECTORのビット)。
StatusIdBitsConnector	すべてのステータスIDの論理和(上記のフラグのようなビット)。
StatusBitTedsPresent	TEDSが存在するかどうかを示す情報(STATUS_ID_BITS_CONNECTORのビット)。
StatusTime	モジュールの現在時刻(値はUInt64)。
StatusModuleInSync	モジュールが同期しているかどうかを示す情報(値=1または0)。
StatusModuleBusy	モジュールがビジーかどうかを示す情報(値=1または0)。

Hbm.Api.QuantumX.Enums.OriginOfNameType Enumeration

チャネル名の起源の種類。

C++

```
enum OriginOfNameType : public int {
  Default = 0,
  User = 1,
  Teds = 2,
  Unknown = 3
};
```

C#

```
public enum OriginOfNameType : int {
  Default = 0,
  User = 1,
  Teds = 2,
  Unknown = 3
}
```

Visual Basic

```
Public Enum OriginOfNameType
Inherits int
  Default = 0
  User = 1
  Teds = 2
  Unknown = 3
End Enum
```

ファイル

OriginOfNameType.cs

メンバー

メンバー	説明
Default	デフォルト名。
User	ユーザによって設定された名前。
Teds	TEDSによって設定された名前。
Unknown	不明。

備考

列挙型の値は、XML内のOriginOfNameのQuantumX固有の値と一致します。APIの共通部分に対応するものがないため、追加の値はありません。

Hbm.Api.QuantumX.Enums.OutputType Enumeration

アナログ出力コネクタの出力の種類。

C++

```
enum OutputType {
  Voltage
};
```

C#

```
public enum OutputType {
  Voltage
}
```

Visual Basic

```
Public Enum OutputType
  Voltage
End Enum
```

ファイル

OutputType.cs

メンバー

メンバー	説明
Voltage	電圧。

Hbm.Api.QuantumX.Enums.QuantumXBridgeType Enumeration

QuantumXのブリッジの種類。
列挙の値は、xmlのBridgeTypeのQuantumX固有の値と一致します (HBM_QX_Framework.hの列挙enBridgeTypeに対応)。
追加の値は、列挙Hbm.Api.SensorDB.Enums.BridgeTypeの値と一致する必要があります。

C++

```
[AdditionalValueType(typeof(int))]
enum QuantumXBridgeType : public int {
  FullBridge = 4,
  HalfBridge = 2,
  QuarterBridge = 1
};
```

C#

```
[AdditionalValueType(typeof(int))]
internal enum QuantumXBridgeType : int {
  FullBridge = 4,
  HalfBridge = 2,
  QuarterBridge = 1
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum QuantumXBridgeType
Inherits int
  FullBridge = 4
  HalfBridge = 2
  QuarterBridge = 1
End Enum
```

ファイル

QuantumXBridgeType.cs

備考

HBM_QX_Framework.h:

```
//! bridge sensor types
enum enBridgeType {
    BRIDGETYPE_QUARTER = 1,
       BRIDGETYPE_HALF = 2,
       BRIDGETYPE_FULL = 4,
       BRIDGETYPE_UNKNOWN = ENUM_UNKNOWN
};
```

Hbm.Api.QuantumX.Enums.QuantumXCarrierFrequencyType Enumeration

QuantumXのキャリア周波数の種類。
列挙の値は、xml内の励起周波数のQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enExcitationFreqに対応)。
追加の値は、列挙Hbm.Api.SensorDB.Enums.CarrierFrequencyTypeの型関連値の一部と一致する必要があります。Hz単位の直接周波数値は、列挙Hbm.Api.QuantumX.Enums.QuantumXCarrierFrequencyValueの一部です。

C++

```
[AdditionalValueType(typeof(int))]
enum QuantumXCarrierFrequencyType : public int {
  Auto = 0,
  Dc = 1,
  AutoAc = 7
};
```

C#

```
[AdditionalValueType(typeof(int))]
internal enum QuantumXCarrierFrequencyType : int {
  Auto = 0,
  Dc = 1,
  AutoAc = 7
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum QuantumXCarrierFrequencyType
Inherits int
  Auto = 0
  Dc = 1
  AutoAc = 7
End Enum
```

ファイル

QuantumXCarrierFrequencyType.cs

備考

HBM_QX_Framework.h:

```
    enum enExcitationFreq
    {
        EXCFREQ_AUTO = 0,       // type relevant
        EXCFREQ_DC = 1,         // type relevant
        EXCFREQ_75HZ = 2,
        EXCFREQ_600HZ = 3,
        EXCFREQ_1200HZ = 4,
        EXCFREQ_4800HZ = 5,
        EXCFREQ_9600HZ = 6,
        EXCFREQ_AUTO_AC = 7,    // type relevant
        EXCFREQ_225HZ = 8,
        EXCFREQ_UNKNOWN = ENUM_UNKNOWN
};
```

Hbm.Api.QuantumX.Enums.QuantumXCarrierFrequencyValue Enumeration

QuantumXのキャリア周波数値。
列挙の値は、xml内の励起周波数のQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enExcitationFreqに対応)。
追加の値は、周波数値を直接定義する列挙Hbm.Api.SensorDB.Enums.CarrierFrequencyTypeの周波数値の一部と一致する必要があります。タイプ関連情報は、列挙Hbm.Api.QuantumX.Enums.QuantumXCarrierFrequencyTypeの一部です。

C++

```
[AdditionalValueType(typeof(decimal))]
enum QuantumXCarrierFrequencyValue : public int {
  Hz75 = 2,
  Hz600 = 3,
  Hz1200 = 4,
  Hz4800 = 5,
  Hz9600 = 6,
  Hz225 = 8
};
```

C#

```
[AdditionalValueType(typeof(decimal))]
internal enum QuantumXCarrierFrequencyValue : int {
  Hz75 = 2,
  Hz600 = 3,
  Hz1200 = 4,
  Hz4800 = 5,
  Hz9600 = 6,
  Hz225 = 8
}
```

Visual Basic

```
<AdditionalValueType(typeof(decimal))>
Friend Enum QuantumXCarrierFrequencyValue
Inherits int
  Hz75 = 2
  Hz600 = 3
  Hz1200 = 4
  Hz4800 = 5
  Hz9600 = 6
  Hz225 = 8
End Enum
```

ファイル

QuantumXCarrierFrequencyValue.cs

備考

HBM_QX_Framework.h:

```
    enum enExcitationFreq
    {
        EXCFREQ_AUTO = 0,
        EXCFREQ_DC = 1,
        EXCFREQ_75HZ = 2,        // type relevant
        EXCFREQ_600HZ = 3,       // type relevant
        EXCFREQ_1200HZ = 4,      // type relevant
        EXCFREQ_4800HZ = 5,      // type relevant
        EXCFREQ_9600HZ = 6,      // type relevant
        EXCFREQ_AUTO_AC = 7,
        EXCFREQ_225HZ = 8,       // type relevant
        EXCFREQ_UNKNOWN = ENUM_UNKNOWN
};
```

Hbm.Api.QuantumX.Enums.QuantumXDataRateDomainType Enumeration

データレートドメインの種類。
列挙の値は、デバイスのシステムビューのxml要素Hbm.Api.QuantumX.Constants.XmlTag.SYSTEM_DATARATEDOMAIN_REQUESTおよびHbm.Api.QuantumX.Constants.XmlTag.SYSTEM_DATARATEDOMAIN_CURRENTSTATEのQuantumX固有の値と一致します。
追加の値は、列挙Hbm.Api.Common.Enums.DataRateDomainTypeの値と一致する必要があります。

C++

```
[AdditionalValueType(typeof(int))]
enum QuantumXDataRateDomainType {
  Classic = 0,
  Decimal = 2,
  Binary = 3
};
```

C#

```
[AdditionalValueType(typeof(int))]
internal enum QuantumXDataRateDomainType {
  Classic = 0,
  Decimal = 2,
  Binary = 3
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum QuantumXDataRateDomainType
  Classic = 0,
  Decimal = 2,
  Binary = 3
End Enum
```

ファイル
QuantumXDataRateDomainType.cs

Hbm.Api.QuantumX.Enums.QuantumXDirectionType Enumeration

QuantumXのデジタルI/Oの方向変換に使用します。
列挙の値は、Hbm.Api.Common.Enums.DirectionTypeの共通値と一致します。追加の値は、xml内の追加コネクタのビュータグHbm.Api.QuantumX.Constants.XmlTag.ADDCONN_DIGITALIO_MODE_DIGITALINまたはHbm.Api.QuantumX.Constants.XmlTag.ADDCONN_DIGITALIO_MODE_DIGITALOUTの文字列コンテンツと一致する必要があります。

C++

```
[AdditionalValueType(typeof(string))]
enum QuantumXDirectionType : public int {
  DigitalIn = DirectionType.In,
  DigitalOut = DirectionType.Out
};
```

C#

```
[AdditionalValueType(typeof(string))]
internal enum QuantumXDirectionType : int {
  DigitalIn = DirectionType.In,
  DigitalOut = DirectionType.Out
}
```

Visual Basic

```
<AdditionalValueType(typeof(string))>
Friend Enum QuantumXDirectionType
Inherits int
  DigitalIn = DirectionType.In
  DigitalOut = DirectionType.Out
End Enum
```

ファイル

QuantumXDirectionType.cs

Hbm.Api.QuantumX.Enums.QuantumXEnableConnectorType Enumeration

QuantumXの特定のコネクタを有効にするために使用します。

C++

```
enum QuantumXEnableConnectorType : public int {
  Default,
  CanBus
};
```

C#

```
public enum QuantumXEnableConnectorType : int {
  Default,
  CanBus
}
```

Visual Basic

```
Public Enum QuantumXEnableConnectorType
Inherits int
  Default
  CanBus
End Enum
```

ファイル

QuantumXEnableConnectorType.cs

メンバー

メンバー	説明
Default	ハードウェアに応じて異なるデフォルトのコネクタ タイプを使用します(MX471の場合はHbm.Api.QuantumX.Connectors.QuantumXCanConnector、その他のQX追加コネクタの場合はHbm.Api.QuantumX.Connectors.QuantumXAnalogOutConnector、それ以外の場合はHbm.Api.QuantumX.Connectors.QuantumXAnalogInConnector)。
CanBus	Hbm.Api.QuantumX.Connectors.QuantumXCanConnectorを使用します(例: MX840の最初のコネクタなど)。

Hbm.Api.QuantumX.Enums.QuantumXPolarityType Enumeration

QuantumXの極性の種類。
列挙の値は、xml内の極性のQuantumX 固有の値と一致します(HBM_QX_Framework.hの列挙enPolarityに対応)。
追加の値は、列挙Hbm.Api.SensorDB.Enums.PolarityTypeの値と一致する必要があります。

C++

```
[AdditionalValueType(typeof(int))]
enum QuantumXPolarityType : public int {
  NotInverted = 0,
  Inverted = 1
};
```

C#

```
[AdditionalValueType(typeof(int))]
internal enum QuantumXPolarityType : int {
  NotInverted = 0,
  Inverted = 1
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum QuantumXPolarityType
Inherits int
  NotInverted = 0
  Inverted = 1
End Enum
```

ファイル

QuantumXPolarityType.cs

備考

HBM_QX_Framework.h:

```
    enum enPolarity
    {
        POLARITY_NOTINVERTED = 0,
        POLARITY_INVERTED    = 1,
        POLARITY_UNKNOWN
};
```

Hbm.Api.QuantumX.Enums.QuantumXPtType Enumeration

QuantumXのPtの種類。
列挙型の値は、xmlのPTの種類のQuantumX 固有の値と一致します(HBM_QX_Framework.hの列挙型enPtTypeに対応)
。
追加の値は、PtTypeに従って公称抵抗と一致する必要があります。

C++

```
[AdditionalValueType(typeof(int))]
enum QuantumXPtType : public int {
  Pt10 = 0,
  Pt100 = 1,
  Pt500 = 2,
  Pt1000 = 3
};
```

C#

```
[AdditionalValueType(typeof(int))]
internal enum QuantumXPtType : int {
  Pt10 = 0,
  Pt100 = 1,
  Pt500 = 2,
  Pt1000 = 3
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum QuantumXPtType
Inherits int
  Pt10 = 0
  Pt100 = 1
  Pt500 = 2
  Pt1000 = 3
End Enum
```

ファイル

QuantumXPtType.cs

備考

HBM_QX_Framework.h:

```
//! PT type
enum enPtType {
    PTTYPE_10 = 0,
       PTTYPE_100 = 1,
       PTTYPE_500 = 2,
       PTTYPE_1000 = 3,
    PTTYPE_UNKNOWN = ENUM_UNKNOWN
};
```

Hbm.Api.QuantumX.Enums.QuantumXSignal2Type Enumeration

QuantumXのSignal2の種類。
列挙の値は、xmlのsignal2の種類のQuantumX固有の値と一致します(HBM_QX_Framework.h の struct stCounter のフィールド S2enableに対応)。
追加の値は、列挙Hbm.Api.SensorDB.Enums.Signal2Typeの値と一致する必要があります。

C++

```
[AdditionalValueType(typeof(int))]
enum QuantumXSignal2Type : public int {
  Off = 0,
  Dynamic = 1,
  Static = 2
};
```

C#

```
[AdditionalValueType(typeof(int))]
internal enum QuantumXSignal2Type : int {
  Off = 0,
  Dynamic = 1,
  Static = 2
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum QuantumXSignal2Type
Inherits int
  Off = 0
  Dynamic = 1
  Static = 2
End Enum
```

ファイル

QuantumXSignal2Type.cs

備考

HBM_QX_Framework.h:

```
//!  Parameter of counting sensor
enum stCounter {
   struct stCounter {
        //! 0: Index channel inactive
        //! 1: Index channel active. Relevant for rotary encoders. With each revolution, the
counter is being reset. See IndexDivisor.
       int IndexEnable;
       //! Index channel divider. Relevant for rotary encoders. The counter is being reset
every IndexDivisor revolution. IndexDivisor=0 has the same effect as IndexDivisor=1.
       unsigned int IndexDivisor;
       //! 0: Signal S2 disabled
       //! 1: Signal S2 enabled. For sensors with A second signal with 90 degree
displacement against S1. Allows to detect the direction. Might increase the accuracy.
       //! 2: Signal S2 is used as static signal to determine direction.
       int S2enable;
       //! 0: Signal F4 disabled
       //! 1: Signal F4 enabled. If S2 signal is enabled, this enables 4-slope evaluation.
Each slope is being counted.
       //! The signal has the fourfold value. Increases the resolution
       int F4enable;
       //! Number of pulses per revolution
       unsigned int PulsesPerRound;
};
```

Hbm.Api.QuantumX.Enums.QuantumXSsiCodingType Enumeration

QuantumXのSSIの種類タイプ。
列挙の値は、xmlのSSIコーディングの種類のQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enSSICodingに対応)。
追加の値は、列挙Hbm.Api.SensorDB.Enums.SsiCodingTypeの値と一致する必要があります。

C++

```
[AdditionalValueType(typeof(int))]
enum QuantumXSsiCodingType : public int {
  Gray = 0,
  Dual = 1
};
```

C#

```
[AdditionalValueType(typeof(int))]
internal enum QuantumXSsiCodingType : int {
  Gray = 0,
  Dual = 1
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum QuantumXSsiCodingType
Inherits int
  Gray = 0
  Dual = 1
End Enum
```

ファイル

QuantumXSsiCodingType.cs

備考

HBM_QX_Framework.h:

```
//! Encoding type for SSI Sensors
enum enSSICoding {
    CODING_GRAY = 0,
       CODING_DUAL = 1,
       CODING_UNKNOWN = ENUM_UNKNOWN
};
```

Hbm.Api.QuantumX.Enums.QuantumXSyncModeType Enumeration

QuantumXの同期モードの変換に使用します。
列挙型の値は、Hbm.Api.Common.Enums.SyncModeTypeの共通値と一致します。追加の値は、xml内のシステムのビューータグHbm.Api.QuantumX.Constants.XmlTag.SYSTEM_SYNCSTATUSの文字列コンテンツと一致する必要があります。

C++

```
[AdditionalValueType(typeof(string))]
enum QuantumXSyncModeType : public int {
  Single = SyncModeType.Standalone,
  Slave = SyncModeType.Slave,
  SyncMaster = SyncModeType.Master
};
```

C#

```
[AdditionalValueType(typeof(string))]
internal enum QuantumXSyncModeType : int {
  Single = SyncModeType.Standalone,
  Slave = SyncModeType.Slave,
  SyncMaster = SyncModeType.Master
}
```

Visual Basic

```
<AdditionalValueType(typeof(string))>
Friend Enum QuantumXSyncModeType
Inherits int
  Single = SyncModeType.Standalone
  Slave = SyncModeType.Slave
  SyncMaster = SyncModeType.Master
End Enum
```

ファイル

QuantumXSyncModeType.cs

Hbm.Api.QuantumX.Enums.QuantumXTedsUsageMode Enumeration

QuantumXのTEDS使用モード。
列挙の値は、xmlのTEDS使用モードのQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enTedsUsageに対応)。
追加の値は、列挙Hbm.Api.SensorDB.Enums.TedsUsageModeの値と一致する必要があります。

C++

```
[AdditionalValueType(typeof(int))]
enum QuantumXTedsUsageMode {
  NotSupported = Int32.MaxValue,
  Ignore = 0,
  UseIfAvailable = 2,
  UseRequired = 3
};
```

C#

```
[AdditionalValueType(typeof(int))]
internal enum QuantumXTedsUsageMode {
  NotSupported = Int32.MaxValue,
  Ignore = 0,
  UseIfAvailable = 2,
  UseRequired = 3
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum QuantumXTedsUsageMode
  NotSupported = Int32.MaxValue
  Ignore = 0
  UseIfAvailable = 2
  UseRequired = 3
End Enum
```

ファイル

QuantumXTedsUsageMode.cs

備考

HBM_QX_Framework.h:

```
//TEDS usage
enum enTedsUsage
{
    TEDSUSAGE_IGNORE = 0,
    TEDSUSAGE_AUTO = 2,
    TEDSUSAGE_REQUIRED = 3,
    TEDSUSAGE_UNKNOWN = ENUM_UNKNOWN //(MAX_INT32)
};
```

Hbm.Api.QuantumX.Enums.QuantumXTestSignalActiveValueType Enumeration

Statuses.QuantumXChannelTestSignalActiveStatusの値パラメータの値を定義します。

C++

```
enum QuantumXTestSignalActiveValueType {
  NotSupported = -1,
  NotActive = 0,
  Active = 1
};
```

C#

```
public enum QuantumXTestSignalActiveValueType {
  NotSupported = -1,
  NotActive = 0,
  Active = 1
}
```

Visual Basic

```
Public Enum QuantumXTestSignalActiveValueType
  NotSupported = -1
  NotActive = 0
  Active = 1
End Enum
```

ファイル

QuantumXTestSignalActiveValueType.cs

メンバー

メンバー	説明
NotSupported	サポートされていません。
NotActive	テスト信号はアクティブではありません。
Active	テスト信号はアクティブです。

Hbm.Api.QuantumX.Enums.QuantumXThermoCoupleType Enumeration

QuantumXの熱電対の種類。
列挙型の値は、xml内のThermocoupleTypeのQuantumX固有の値と一致します(HBM_QX_Framework.h内の列挙型enPtType に対応)。
追加の値は、センサデータベースのデータフィールドに従って熱電対の種類と一致する必要があります。

C++

```
[AdditionalValueType(typeof(string))]
enum QuantumXThermoCoupleType : public int {
  B = 0,
  C = 8,
  E = 1,
  J = 2,
  K = 3,
  N = 4,
  R = 5,
  S = 6,
  T = 7
};
```

C#

```
[AdditionalValueType(typeof(string))]
internal enum QuantumXThermoCoupleType : int {
  B = 0,
  C = 8,
  E = 1,
  J = 2,
  K = 3,
  N = 4,
  R = 5,
  S = 6,
  T = 7
}
```

Visual Basic

```
<AdditionalValueType(typeof(string))>
Friend Enum QuantumXThermoCoupleType
Inherits int
  B = 0
  C = 8
  E = 1
  J = 2
  K = 3
  N = 4
  R = 5
  S = 6
  T = 7
End Enum
```

ファイル
QuantumXThermoCoupleType.cs

備考

HBM_QX_Framework.h:

```
enum enTcType{
    TCTYPE_B = 0,
    TCTYPE_E = 1,
    TCTYPE_J = 2,
    TCTYPE_K = 3,
    TCTYPE_N = 4,
    TCTYPE_R = 5,
    TCTYPE_S = 6,
    TCTYPE_T = 7,
    TCTYPE_C = 8,
    TCTYPE_UNKNOWN = ENUM_UNKNOWN
};
```

Hbm.Api.QuantumX.Enums.TedsValidation Enumeration

ModuleStatusID.STATUS_TEDSの値。列挙の値は、xmlのTEDS使用モードのQuantumX固有の値と一致します(HBM_QX_Framework.hの列挙enTedsUsageに対応)。追加の値は、列挙Common.Enums.TedsValidationValueTypeの値と一致する必要があります。

C++

```
[AdditionalValueType(typeof(int))]
enum TedsValidation {
  TedsNotAvailable = 0,
  TedsValid = 1,
  TedsChecksumError = 2,
  TedsDataEmpty = 3,
  TedsDataNotSupportedByConnector = 4,
  TedsDataParamBusy = 5,
  TedsValidRequestPressureEqualAmp = 6,
  TedsValidRequestPressureBelowAmp = 7,
  TedsValidRequestPressureAboveAmp = 8,
  TedsDataNotUsed = 9,
  TedsUnknown = int.MaxValue
};
```

C#

```
[AdditionalValueType(typeof(int))]
internal enum TedsValidation {
  TedsNotAvailable = 0,
  TedsValid = 1,
  TedsChecksumError = 2,
  TedsDataEmpty = 3,
  TedsDataNotSupportedByConnector = 4,
  TedsDataParamBusy = 5,
  TedsValidRequestPressureEqualAmp = 6,
  TedsValidRequestPressureBelowAmp = 7,
  TedsValidRequestPressureAboveAmp = 8,
  TedsDataNotUsed = 9,
  TedsUnknown = int.MaxValue
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum TedsValidation
  TedsNotAvailable = 0
  TedsValid = 1
  TedsChecksumError = 2
  TedsDataEmpty = 3
  TedsDataNotSupportedByConnector = 4
  TedsDataParamBusy = 5
  TedsValidRequestPressureEqualAmp = 6
  TedsValidRequestPressureBelowAmp = 7
  TedsValidRequestPressureAboveAmp = 8
  TedsDataNotUsed = 9
  TedsUnknown = int.MaxValue
End Enum
```

ファイル
TedsValidation.cs

メンバー

メンバー	説明
TedsUnknown	ENUM_UNKNOWN

Hbm.Api.QuantumX.Enums.XmlViewType Enumeration

デバイスからXMLデータを読み書きするためのビューの種類。

C++

```
enum XmlViewType {
  ComSettings = 1,
  System = 3,
  Connector = 5,
  AdditionalFunction = 6,
  AdditionalConnector = 8
};
```

C#

```
public enum XmlViewType {
  ComSettings = 1,
  System = 3,
  Connector = 5,
  AdditionalFunction = 6,
  AdditionalConnector = 8
}
```

Visual Basic

```
Public Enum XmlViewType
  ComSettings = 1
  System = 3
  Connector = 5
  AdditionalFunction = 6
  AdditionalConnector = 8
End Enum
```

ファイル

XmlViewType.cs

メンバー

メンバー	説明
ComSettings	通信とネットワーク設定ビュー。
System	システムビュー。
Connector	コネクタビュー。
AdditionalFunction	追加機能ビュー。
AdditionalConnector	追加コネクタビュー。

メンバー

列挙の値は、Hbm.Api.QuantumX.Core.FrameworkDllWrapperのXML関連メソッドのビューパラメータと一致する必要があります(例: HBM_XPAR_ReadXML)。

Hbm.Api.QuantumX.Exceptions Namespace

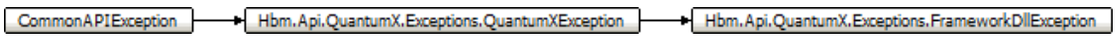
これは名前空間Hbm.Api.QuantumX.Exceptionsです。

クラス

	名前	説明
	FrameworkDllException	この例外は、HBM_QX_Framework.dllがエラーをスローした場合、またはHBM_QX_Framework.dllのメソッド呼び出しがエラー コードを返した場合にスローされます。
	QuantumXException	QuantumX固有エラーの基本クラス。

FrameworkDllException Class

この例外は、HBM_QX_Framework.dllがエラーをスローした場合、またはHBM_QX_Framework.dllのメソッド呼び出しがエラー コードを返した場合にスローされます。



C++

```
[Serializable]
class FrameworkDllException : public QuantumXException;
```

C#

```
[Serializable]
public class FrameworkDllException : QuantumXException;
```

Visual Basic

```
<Serializable>
Public Class FrameworkDllException
Inherits QuantumXException
```

ファイル

FrameworkDllException.cs

FrameworkDllException Members

次の表は、FrameworkDllExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	FrameworkDllException	これは、FrameworkDllExceptionコンストラクタの多重定義の概要です。
	QuantumXException	これは、QuantumXExceptionコンストラクタの多重定義の概要です。

FrameworkDllException Constructor

これは、FrameworkDllExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	FrameworkDllException. FrameworkDllException ()	新しいインスタンスを作成します。
	FrameworkDllException. FrameworkDllException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	FrameworkDllException. FrameworkDllException (string, System. Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

FrameworkDllException.FrameworkDllException Constructor ()

新しいインスタンスを作成します。

C++

```
FrameworkDllException();
```

C#

```
public FrameworkDllException();
```

Visual Basic

```
Public Sub New()
```

FrameworkDllException.FrameworkDllException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
FrameworkDllException(
    string message
);
```

C#

```
public FrameworkDllException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

FrameworkDllException.FrameworkDllException Constructor (string, System.Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
FrameworkDllException(
    string message,
    System.Exception inner
);
```

C#

```
public FrameworkDllException(
    string message,
    System.Exception inner
);
```

Visual Basic

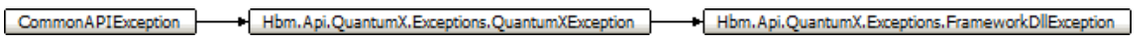
```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

QuantumXException Class

QuantumX固有エラーの基本クラス。



C++

```
[Serializable]
class QuantumXException : public CommonAPIException;
```

C#

```
[Serializable]
public abstract class QuantumXException : CommonAPIException;
```

Visual Basic

```
<Serializable>
Public abstract Class QuantumXException
Inherits CommonAPIException
```

ファイル

QuantumXException.cs

QuantumXException Members

次の表は、QuantumXExceptionによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	QuantumXException	これは、QuantumXExceptionコンストラクタの多重定義の概要です。

QuantumXException Constructor

これは、QuantumXExceptionコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	QuantumXException.QuantumXException ()	新しいインスタンスを作成します。
	QuantumXException.QuantumXException (string)	指定したエラーメッセージを使用して新しいインスタンスを作成します。
	QuantumXException.QuantumXException (string, System.Exception)	指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

QuantumXException.QuantumXException Constructor ()

新しいインスタンスを作成します。

C++

```
QuantumXException();
```

C#

```
public QuantumXException();
```

Visual Basic

```
Public Sub New()
```

QuantumXException.QuantumXException Constructor (string)

指定したエラーメッセージを使用して新しいインスタンスを作成します。

C++

```
QuantumXException(
    string message
);
```

C#

```
public QuantumXException(
    string message
);
```

Visual Basic

```
Public Sub New(
    message As string
)
```

引数

引数	説明
message	エラーメッセージ。

QuantumXException.QuantumXException Constructor (string, System. Exception)

指定したエラーメッセージと内部例外を使用して新しいインスタンスを作成します。

C++

```
QuantumXException(
    string message,
    System.Exception inner
);
```

C#

```
public QuantumXException(
    string message,
    System.Exception inner
);
```

Visual Basic

```
Public Sub New(
    message As string,
    inner As System.Exception
)
```

引数

引数	説明
message	エラーメッセージ。
inner	エラーの内部例外。

Hbm.Api.QuantumX.Signals Namespace

これは名前空間Hbm.Api.QuantumX.Signalsです。

クラス

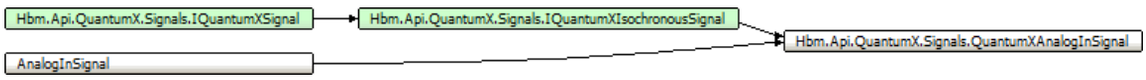
名前	説明
QuantumXAnalogInSignal	物理センサから配信されるQuantumX固有の測定信号を表します。
QuantumXAnalogOutSignal	QuantumX固有のアナログ出力信号を表します。
QuantumXCanInSignal	CAN bus経由でセンサから配信されるQuantumX固有の測定信号を表します。
QuantumXCanRawSignal	CAN bus経由でセンサから送信されるQuantumX固有の測定信号(RAW信号)を表します。
QuantumXDigitalSignal	QuantumX固有のデジタル入力または出力信号を表します。
QuantumXSyncSignal	物理センサから配信されるQuantumX固有の測定信号を表します。

インタフェース

	名前	説明
	IQuantumXIsochronousSignal	すべてのアイソクロナスQuantumX信号で実装する必要があるインタフェース。このインタフェースは、アイソクロナスQuantumX信号を識別するために使用されます。
	IQuantumXSignal	すべてのQuantumX信号で実装する必要があるインタフェース。このインタフェースは、QuantumX信号を識別するために使用されます。

QuantumXAnalogInSignal Class

物理センサから配信されるQuantumX固有の測定信号を表します。



C++

```
class QuantumXAnalogInSignal : public AnalogInSignal, public IQuantumXIsochronousSignal;
```

C#

```
public class QuantumXAnalogInSignal : AnalogInSignal, IQuantumXIsochronousSignal;
```

Visual Basic

```
Public Class QuantumXAnalogInSignal
Inherits AnalogInSignal
Inherits IQuantumXIsochronousSignal
```

ファイル

QuantumXAnalogInSignal.cs

QuantumXAnalogInSignal Members

次の表は、QuantumXAnalogInSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。
	QuantumXAnalogInSignal	これは、QuantumXAnalogInSignalコンストラクタの多重定義の概要です。

Public Properties

	名前	説明
	Description	信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。
	IsDaqAvailable	データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	OriginOfName	チャネル名の起源(読み取り専用)。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

QuantumXAnalogInSignal Constructor

これは、QuantumXAnalogInSignalコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	QuantumXAnalogInSignal. QuantumXAnalogInSignal ()	QuantumXAnalogInSignalクラスの新しいインスタンスを初期化します。
	QuantumXAnalogInSignal. QuantumXAnalogInSignal (QuantumXSyncSignal)	QuantumXAnalogInSignalクラスの新しいインスタンスを初期化します。

QuantumXAnalogInSignal.QuantumXAnalogInSignal Constructor ()

QuantumXAnalogInSignalクラスの新しいインスタンスを初期化します。

C++

```
QuantumXAnalogInSignal();
```

C#

```
public QuantumXAnalogInSignal();
```

Visual Basic

```
Public Sub New()
```

QuantumXAnalogInSignal.QuantumXAnalogInSignal Constructor (QuantumXSyncSignal)

QuantumXAnalogInSignalクラスの新しいインスタンスを初期化します。

C++

```
QuantumXAnalogInSignal(
    QuantumXSyncSignal qxSyncSignal
);
```

C#

```
public QuantumXAnalogInSignal(
    QuantumXSyncSignal qxSyncSignal
);
```

Visual Basic

```
Public Sub New(
    qxSyncSignal As QuantumXSyncSignal
)
```

引数

引数	説明
qxSyncSignal	新しいQuantumXAnalogInSignalを作成するためのプロパティを含むQuantumX同期信号。

QuantumXAnalogInSignal Methods

QuantumXAnalogInSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。

QuantumXAnalogInSignal.GetUniqueID Method

信号が属するデバイス間で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェアモジュール上の信号を識別するためにストリーミングで使用される情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXAnalogInSignal Properties

QuantumXAnalogInSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Description	信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。
	IsDaqAvailable	データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	OriginOfName	チャネル名の起源(読み取り専用)。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

QuantumXAnalogInSignal.Description Property

信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。

C++

```
[ JsonProperty ]
__property string Description;
```

C#

```
[ JsonProperty ]
public string Description;
```

Visual Basic

```
< JsonProperty >
Public Property Description() As string
```

QuantumXAnalogInSignal.IsDaqAvailable Property

データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。

C++

```
[ JsonProperty ]
__property bool IsDaqAvailable;
```

C#

```
[ JsonProperty ]
public bool IsDaqAvailable;
```

Visual Basic

```
< JsonProperty >
Public Property IsDaqAvailable() As bool
```

QuantumXAnalogInSignal.IsIsochronous Property

アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。

C++

```
[ JsonProperty ]
__property bool IsIsochronous;
```

C#

```
[ JsonProperty ]
public bool IsIsochronous;
```

Visual Basic

```
< JsonProperty >
Public Property IsIsochronous() As bool
```

備考

このプロパティは、意味を明確にし、GUIで使用する用語に反映するため、V1.0.0.0と比較して名前が変更されました。

QuantumXAnalogInSignal.OriginOfName Property

チャネル名の起源(読み取り専用)。

C++

```
[ JsonProperty ]
__property OriginOfNameType OriginOfName;
```

C#

```
[ JsonProperty ]
public OriginOfNameType OriginOfName;
```

Visual Basic

```
< JsonProperty >
Public Property OriginOfName() As OriginOfNameType
```

QuantumXAnalogInSignal.Position Property

信号が属するコネクタの位置。

C++

```
[ JsonProperty ]
__property int Position;
```

C#

```
[ JsonProperty ]
public int Position;
```

Visual Basic

```
< JsonProperty >
Public Property Position() As Integer
```

QuantumXAnalogInSignal.SignalReference Property

この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

C++

```
[ JsonProperty ]
__property string SignalReference;
```

C#

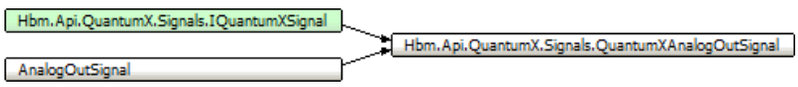
```
[ JsonProperty ]
public string SignalReference;
```

Visual Basic

```
< JsonProperty >
Public Property SignalReference() As string
```

QuantumXAnalogOutSignal Class

QuantumX固有のアナログ出力信号を表します。



C++

```
class QuantumXAnalogOutSignal : public AnalogOutSignal, public IQuantumXSignal;
```

C#

```
public class QuantumXAnalogOutSignal : AnalogOutSignal, IQuantumXSignal;
```

Visual Basic

```
Public Class QuantumXAnalogOutSignal
Inherits AnalogOutSignal
Inherits IQuantumXSignal
```

ファイル

QuantumXAnalogOutSignal.cs

備考

この種の信号を測定に含めることはできません。

QuantumXAnalogOutSignal Members

次の表は、QuantumXAnalogOutSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。
	QuantumXAnalogOutSignal	クラスの新しいインスタンスを作成し、変数を初期化します。

Public Properties

	名前	説明
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンス(読み取り専用)。

QuantumXAnalogOutSignal.QuantumXAnalogOutSignal Constructor

クラスの新しいインスタンスを作成し、変数を初期化します。

C++

```
QuantumXAnalogOutSignal();
```

C#

```
public QuantumXAnalogOutSignal();
```

Visual Basic

```
Public Sub New()
```

QuantumXAnalogOutSignal Methods

QuantumXAnalogOutSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。

QuantumXAnalogOutSignal.GetUniqueID Method

信号が属するデバイス間で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェアモジュール上の信号を識別するためにストリーミングで使用される情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXAnalogOutSignal Properties

QuantumXAnalogOutSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンス(読み取り専用)。

QuantumXAnalogOutSignal.Position Property

信号が属するコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXAnalogOutSignal.SignalReference Property

この信号ソースの一意のリファレンス(読み取り専用)。

C++

```
__property string SignalReference;
```

C#

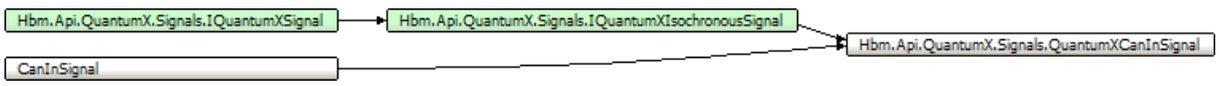
```
public string SignalReference;
```

Visual Basic

```
Public Property SignalReference() As string
```

QuantumXCanInSignal Class

CAN bus経由でセンサから配信されるQuantumX固有の測定信号を表します。



C++

```
class QuantumXCanInSignal : public CanInSignal, public IQuantumXIsochronousSignal;
```

C#

```
public class QuantumXCanInSignal : CanInSignal, IQuantumXIsochronousSignal;
```

Visual Basic

```
Public Class QuantumXCanInSignal
Inherits CanInSignal
Inherits IQuantumXIsochronousSignal
```

ファイル

QuantumXCanInSignal.cs

QuantumXCanInSignal Members

次の表は、QuantumXCanInSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	Description	信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。
	IsDaqAvailable	データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

QuantumXCanInSignal Methods

QuantumXAnalogOutSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。

QuantumXCanInSignal.GetUniqueID Method

信号が属するデバイス間で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェアモジュール上の信号を識別するためにストリーミングで使用される情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXCanInSignal Properties

QuantumXCanInSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Description	信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。
	IsDaqAvailable	データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

QuantumXCanInSignal.Description Property

信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。

C++

```
[JsonProperty]
__property string Description;
```

C#

```
[JsonProperty]
public string Description;
```

Visual Basic

```
<JsonProperty>
Public Property Description() As string
```

QuantumXCanInSignal.IsDaqAvailable Property

データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。

C++

```
[JsonProperty]
__property bool IsDaqAvailable;
```

C#

```
[JsonProperty]
public bool IsDaqAvailable;
```

Visual Basic

```
<JsonProperty>
Public Property IsDaqAvailable() As bool
```

QuantumXCanInSignal.IsIsochronous Property

アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。

C++

```
[JsonProperty]
__property bool IsIsochronous;
```

C#

```
[JsonProperty]
public bool IsIsochronous;
```

Visual Basic

```
<JsonProperty>
Public Property IsIsochronous() As bool
```

備考

このプロパティは、意味を明確にし、GUIで使用する用語に反映するため、V1.0.0.0と比較して名前が変更されました。

QuantumXCanInSignal.Position Property

信号が属するコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXCanInSignal.SignalReference Property

この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

C++

```
__property string SignalReference;
```

C#

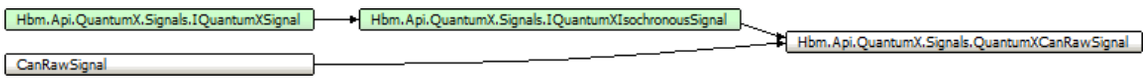
```
public string SignalReference;
```

Visual Basic

```
Public Property SignalReference() As string
```

QuantumXCanRawSignal Class

CAN bus経由でセンサから送信されるQuantumX固有の測定信号 (RAW信号)を表します。



C++

```
class QuantumXCanRawSignal : public CanRawSignal, public IQuantumXIsochronousSignal;
```

C#

```
public class QuantumXCanRawSignal : CanRawSignal, IQuantumXIsochronousSignal;
```

Visual Basic

```
Public Class QuantumXCanRawSignal
Inherits CanRawSignal
Inherits IQuantumXIsochronousSignal
```

ファイル

QuantumXCanRawSignal.cs

QuantumXCanRawSignal Members

次の表は、QuantumXCanRawSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	IsDaqAvailable	データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

QuantumXCanRawSignal Methods

QuantumXCanRawSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。

QuantumXCanRawSignal.GetUniqueID Method

信号が属するデバイス間で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

この情報はスキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXCanRawSignal Properties

QuantumXCanRawSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	IsDaqAvailable	データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

QuantumXCanRawSignal.IsDaqAvailable Property

データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。

C++

```
[JsonProperty]
__property bool IsDaqAvailable;
```

C#

```
[JsonProperty]
public bool IsDaqAvailable;
```

Visual Basic

```
<JsonProperty>
Public Property IsDaqAvailable() As bool
```

QuantumXCanRawSignal.IsIsochronous Property

アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。

C++

```
__property bool IsIsochronous;
```

C#

```
public bool IsIsochronous;
```

Visual Basic

```
Public Property IsIsochronous() As bool
```

備考

このプロパティは、意味を明確にし、GUIで使用する用語に反映するため、V1.0.0.0と比較して名前が変更されました。

QuantumXCanRawSignal.Position Property

信号が属するコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXCanRawSignal.SignalReference Property

この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

C++

```
__property string SignalReference;
```

C#

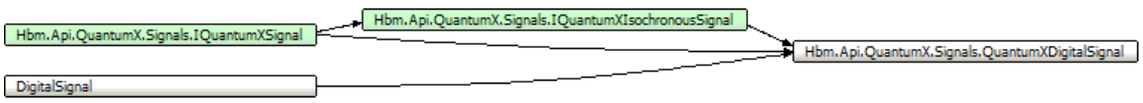
```
public string SignalReference;
```

Visual Basic

```
Public Property SignalReference() As string
```

QuantumXDigitalSignal Class

QuantumX固有のデジタル入力または出力信号を表します。



C++

```
class QuantumXDigitalSignal : public DigitalSignal, public IQuantumXSignal, public
IQuantumXIsochronousSignal;
```

C#

```
public class QuantumXDigitalSignal : DigitalSignal, IQuantumXSignal,
IQuantumXIsochronousSignal;
```

Visual Basic

```
Public Class QuantumXDigitalSignal
Inherits DigitalSignal
Inherits IQuantumXSignal
Inherits IQuantumXIsochronousSignal
```

ファイル

QuantumXDigitalSignal.cs

QuantumXDigitalSignal Members

次の表は、QuantumXDigitalSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	Description	信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。
	IsDaqAvailable	データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	OriginOfName	チャネル名の起源(読み取り専用)。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

QuantumXDigitalSignal Methods

QuantumXDigitalSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。

QuantumXDigitalSignal.GetUniqueID Method

信号が属するデバイス間で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェアモジュール上の信号を識別するためにストリーミングで使用される情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXDigitalSignal Properties

QuantumXDigitalSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Description	信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。
	IsDaqAvailable	データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	OriginOfName	チャネル名の起源(読み取り専用)。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

QuantumXDigitalSignal.Description Property

信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。

C++

```
[JsonProperty]
__property string Description;
```

C#

```
[JsonProperty]
public string Description;
```

Visual Basic

```
<JsonProperty>
Public Property Description() As string
```

QuantumXDigitalSignal.IsDaqAvailable Property

データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。

C++

```
[JsonProperty]
__property bool IsDaqAvailable;
```

C#

```
[JsonProperty]
public bool IsDaqAvailable;
```

Visual Basic

```
<JsonProperty>
Public Property IsDaqAvailable() As bool
```

QuantumXDigitalSignal.IsIsochronous Property

アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。

C++

```
__property bool IsIsochronous;
```

C#

```
public bool IsIsochronous;
```

Visual Basic

```
Public Property IsIsochronous() As bool
```

備考

このプロパティは、意味を明確にし、GUIで使用する用語に反映するため、V1.0.0.0と比較して名前が変更されました。

QuantumXDigitalSignal.OriginOfName Property

チャネル名の起源(読み取り専用)。

C++

```
[JsonProperty]
__property OriginOfNameType OriginOfName;
```

C#

```
[JsonProperty]
public OriginOfNameType OriginOfName;
```

Visual Basic

```
<JsonProperty>
Public Property OriginOfName() As OriginOfNameType
```

QuantumXDigitalSignal.Position Property

信号が属するコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXDigitalSignal.SignalReference Property

この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

C++

```
__property string SignalReference;
```

C#

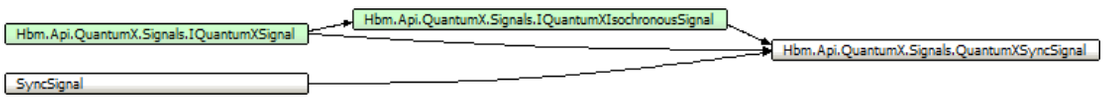
```
public string SignalReference;
```

Visual Basic

```
Public Property SignalReference() As string
```

QuantumXSyncSignal Class

物理センサから配信されるQuantumX固有の測定信号を表します。



C++

```
class QuantumXSyncSignal : public SyncSignal, public IQuantumXSignal, public
IQuantumXIsochronousSignal;
```

C#

```
public class QuantumXSyncSignal : SyncSignal, IQuantumXSignal, IQuantumXIsochronousSignal;
```

Visual Basic

```
Public Class QuantumXSyncSignal
Inherits SyncSignal
Inherits IQuantumXSignal
Inherits IQuantumXIsochronousSignal
```

ファイル

QuantumXSyncSignal.cs

QuantumXSyncSignal Members

次の表は、QuantumXSyncSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	Description	信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。
	IsDaqAvailable	データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	OriginOfName	チャネル名の起源(読み取り専用)。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

QuantumXSyncSignal Methods

QuantumXSyncSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス間で信号を識別する一意の文字列を返します。

QuantumXSyncSignal.GetUniqueID Method

信号が属するデバイス間で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。識別子が設定されていない場合は、String.Emptyになります。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェアモジュール上の信号を識別するためにストリーミングで使用される情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。このメソッドは例外をスローしてはなりません。

QuantumXSyncSignal Properties

QuantumXSyncSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Description	信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。
	IsDaqAvailable	データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	OriginOfName	チャネル名の起源(読み取り専用)。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

QuantumXSyncSignal.Description Property

信号の説明(読み取り専用)。例: "analog input: connector 2, signal 1"。

C++

```
[JsonProperty]
__property string Description;
```

C#

```
[JsonProperty]
public string Description;
```

Visual Basic

```
<JsonProperty>
Public Property Description() As string
```

QuantumXSyncSignal.IsDaqAvailable Property

データ取得のための送信が有効になっているかどうかを示します(読み取り専用)。

C++

```
[JsonProperty]
__property bool IsDaqAvailable;
```

C#

```
[JsonProperty]
public bool IsDaqAvailable;
```

Visual Basic

```
<JsonProperty>
Public Property IsDaqAvailable() As bool
```

QuantumXSyncSignal.IsIsochronous Property

アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。

C++

```
__property bool IsIsochronous;
```

C#

```
public bool IsIsochronous;
```

Visual Basic

```
Public Property IsIsochronous() As bool
```

備考

このプロパティは、意味を明確にし、GUIで使用する用語に反映するため、V1.0.0.0と比較して名前が変更されました。

QuantumXSyncSignal.OriginOfName Property

チャネル名の起源(読み取り専用)。

C++

```
[JsonProperty]
__property OriginOfNameType OriginOfName;
```

C#

```
[JsonProperty]
public OriginOfNameType OriginOfName;
```

Visual Basic

```
<JsonProperty>
Public Property OriginOfName() As OriginOfNameType
```

QuantumXSyncSignal.Position Property

信号が属するコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

QuantumXSyncSignal.SignalReference Property

この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

C++

```
__property string SignalReference;
```

C#

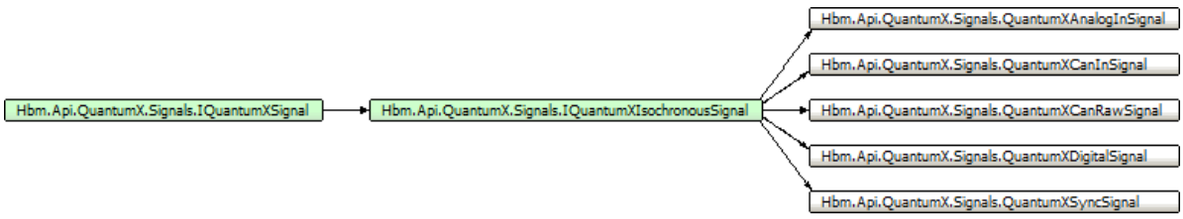
```
public string SignalReference;
```

Visual Basic

```
Public Property SignalReference() As string
```

IQuantumXIsochronousSignal Interface

すべてのアイソクロナスQuantumX信号で実装する必要があるインタフェース。このインタフェースは、アイソクロナスQuantumX信号を識別するために使用されます。



C++

```
__interface IQuantumXIsochronousSignal : public IQuantumXSignal;
```

C#

```
public interface IQuantumXIsochronousSignal : IQuantumXSignal;
```

Visual Basic

```
Public Interface IQuantumXIsochronousSignal
Inherits IQuantumXSignal
```

ファイル

IQuantumXIsochronousSignal.cs

IQuantumXIsochronousSignal Members

次の表は、IQuantumXIsochronousSignalによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

IQuantumXIsochronousSignal Properties

QuantumXSyncSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	IsIsochronous	アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。

IQuantumXIsochronousSignal.IsIsochronous Property

アイソクロナスデータ転送(システム全体のブロードキャストのリアルタイム転送)が有効になっているかどうかを示します。

C++

```
[JsonProperty]
__property bool IsIsochronous;
```

C#

```
[JsonProperty]
public bool IsIsochronous;
```

Visual Basic

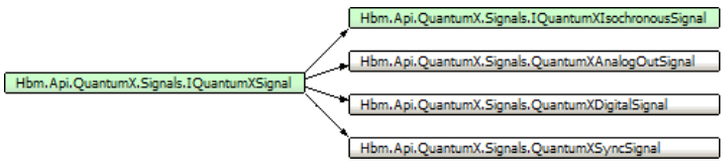
```
<JsonProperty>
Public Property IsIsochronous() As bool
```

備考

このプロパティは、意味を明確にし、GUIで使用する用語に反映するため、V1.0.0.0と比較して名前が変更されました。

IQuantumXSignal Interface

すべてのQuantumX信号で実装する必要があるインタフェース。このインタフェースは、QuantumX信号を識別するために使用されます。



C++

```
__interface IQuantumXSignal;
```

C#

```
public interface IQuantumXSignal;
```

Visual Basic

```
Public Interface IQuantumXSignal
```

ファイル

IQuantumXSignal.cs

IQuantumXSignal Members

次の表は、IQuantumXSignalによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

IQuantumXSignal Properties

IQuantumXSignalクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Position	信号が属するコネクタの位置。
	SignalReference	この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

IQuantumXSignal.Position Property

信号が属するコネクタの位置。

C++

```
[JsonProperty]
__property int Position;
```

C#

```
[JsonProperty]
int Position;
```

Visual Basic

```
<JsonProperty>
Property Position() As Integer
```

IQuantumXSignal.SignalReference Property

この信号ソースの一意のリファレンスで、出力信号または仮想信号のソースとして使用できます(読み取り専用)。

C++

```
[JsonProperty]
__property string SignalReference;
```

C#

```
[JsonProperty]
string SignalReference;
```

Visual Basic

```
<JsonProperty>
Property SignalReference() As string
```

Hbm.Api.QuantumX.Statuses Namespace

これは名前空間Hbm.Api.QuantumX.Statusesです。

クラス

名前	説明
QuantumXChannelTestSignalActiveStatus	1つのチャネルのテスト信号ステータス情報を表します。
QuantumXConnectorShuntStatus	1つのコネクタのシャントステータス情報を表します。

QuantumXChannelTestSignalActiveStatus Class

1つのチャネルのテスト信号ステータス情報を表します。



C++

```
class QuantumXChannelTestSignalActiveStatus : public DeviceStatus;
```

C#

```
public class QuantumXChannelTestSignalActiveStatus : DeviceStatus;
```

Visual Basic

```
Public Class QuantumXChannelTestSignalActiveStatus
Inherits DeviceStatus
```

ファイル

QuantumXChannelTestSignalActiveStatus.cs

QuantumXChannelTestSignalActiveStatus Members

次の表は、QuantumXChannelTestSignalActiveStatusによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。
	QuantumXChannelTestSignalActiveStatus	QuantumXChannelTestSignalActiveStatusクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	Value	ステータスの値を取得します。

QuantumXChannelTestSignalActiveStatus.QuantumXChannelTestSignalActiveStatus Constructor

QuantumXChannelTestSignalActiveStatusクラスの新しいインスタンスを初期化します。

C++

```
QuantumXChannelTestSignalActiveStatus(
    int originalID,
    QuantumXTestSignalActiveValueType value,
    Channel channel
);
```

C#

```
public QuantumXChannelTestSignalActiveStatus(
    int originalID,
    QuantumXTestSignalActiveValueType value,
    Channel channel
);
```

Visual Basic

```
Public Sub  New(
    originalID As Integer,
    value As QuantumXTestSignalActiveValueType,
    channel As Channel
)
```

引数

引数	説明
originalID	元のID。
value	値。
channel	チャネル。

QuantumXChannelTestSignalActiveStatus Methods

QuantumXChannelTestSignalActiveStatusクラスのメソッドの一覧です。

Public Methods

	名前	説明
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

QuantumXChannelTestSignalActiveStatus.IsEqual Method

指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

C++

```
bool IsEqual(
    DeviceStatus status
);
```

C#

```
public override bool IsEqual(
    DeviceStatus status
);
```

Visual Basic

```
Public override Function IsEqual(
    states As DeviceStatus
) As bool
```

引数

引数	説明
states	ステータス。

戻り値

指定したステータスが等しい場合は`true`。それ以外の場合は`false`。

QuantumXChannelTestSignalActiveStatus Properties

QuantumXChannelTestSignalActiveStatusクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Value	ステータスの値を取得します。

QuantumXChannelTestSignalActiveStatus.Value Property

ステータスの値を取得します。

C++

```
__property QuantumXTestSignalActiveValueType Value;
```

C#

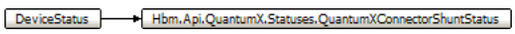
```
public QuantumXTestSignalActiveValueType Value;
```

Visual Basic

```
Public Property Value() As QuantumXTestSignalActiveValueType
```

QuantumXConnectorShuntStatus Class

1つのコネクタのシャントステータス情報を表します。



C++

```
class QuantumXConnectorShuntStatus : public DeviceStatus;
```

C#

```
public class QuantumXConnectorShuntStatus : DeviceStatus;
```

Visual Basic

```
Public Class QuantumXConnectorShuntStatus
Inherits DeviceStatus
```

ファイル

QuantumXConnectorShuntStatus.cs

QuantumXConnectorShuntStatus Members

次の表は、QuantumXConnectorShuntStatusによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。
	QuantumXConnectorShuntStatus	QuantumXConnectorShuntStatusクラスの新しいインスタンスを初期化します。

Public Properties

	名前	説明
	Value	ステータスの値を取得します。

QuantumXConnectorShuntStatus.QuantumXConnectorShuntStatus Constructor

QuantumXConnectorShuntStatusクラスの新しいインスタンスを初期化します。

C++

```
QuantumXConnectorShuntStatus(
    int originalID,
    ShuntMode value,
    Connector connector
);
```

C#

```
public QuantumXConnectorShuntStatus(
    int originalID,
    ShuntMode value,
    Connector connector
);
```

Visual Basic

```
Public Sub  New(
    originalID As Integer,
    value As ShuntMode,
    connector As Connector
)
```

引数

引数	説明
originalID	元のID。
value	値。
connector	コネクタ。

QuantumXConnectorShuntStatus Methods

QuantumXConnectorShuntStatusクラスのメソッドの一覧です。

Public Methods

	名前	説明
	IsEqual	指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

QuantumXConnectorShuntStatus.IsEqual Method

指定したステータスが現在のデバイスのステータスと等しいかどうかを判断します。

C++

```
bool IsEqual(
    DeviceStatus status
);
```

C#

```
public override bool IsEqual(
    DeviceStatus status
);
```

Visual Basic

```
Public override Function IsEqual(
    states As DeviceStatus
) As bool
```

引数

引数	説明
states	ステータス。

戻り値

指定したステータスが等しい場合は`true`。それ以外の場合は`false`。

QuantumXConnectorShuntStatus Properties

QuantumXConnectorShuntStatusクラスのプロパティの一覧です。

Public Properties

	名前	説明
	Value	ステータスの値を取得します。

QuantumXConnectorShuntStatus.Value Property

ステータスの値を取得します。

C++

```
__property ShuntMode Value;
```

C#

```
public ShuntMode Value;
```

Visual Basic

```
Public Property Value() As ShuntMode
```

Hbm.Api.Pmx Namespace

名前空間Hbm.Api.Pmxには、主にPMXデバイスを使用して測定を設定および実行する機能が含まれています。

したがって、抽象クラスHbm.Api.Common.Device(および抽象クラスHbm.Api.Common.Entities.DeviceFamily)を実装します。

クラス

名前	説明
Hbm.Api.Pmx	これは名前空間Hbm.Api.Pmxです。
Hbm.Api.Pmx.Channels	これは名前空間Hbm.Api.Pmx.Channelsです。
Hbm.Api.Pmx.Connectors	これは名前空間Hbm.Api.Pmx.Connectorsです。
Hbm.Api.Pmx.Enums	これは名前空間Hbm.Api.Pmx.Enumsです。
Hbm.Api.Pmx.Signals	これは名前空間Hbm.Api.Pmx.Signalsです。

トピック

名前	説明
サポートされていないPMXの機能	次の表に、APIで現在サポートされていないいくつかの機能を示します。
PMXの追加機能	これらはPMXの追加機能です。また、デバイス経由でもアクセスできます。これらにアクセスするには、共通デバイス(デバイス スキャンで配信されるなど)をPmxDeviceにキャストする必要があります。PMXのすべての追加機能は、AdditionalFeaturesの下にグループ化されています。

サポートされていないPMXの機能

次の表に、APIで現在サポートされていないいくつかの機能を示します。

機能	説明
仮想チャネル	仮想チャネル/信号は、APIでパラメータ化または作成することはできません。
ファームウェア	02.04以降でサポートされています。3.04.11149でテスト済みです。

サポートされていないフィールドバスボード: PX01EC、PX01PN、PX01EP

PMXの追加機能

これらはPMXの追加機能です。また、デバイス経由でもアクセスできます。これらにアクセスするには、共通デバイス(デバイス スキャンで配信されるなど)をPmxDeviceにキャストする必要があります。PMXのすべての追加機能は、AdditionalFeaturesの下にグループ化されています。

例

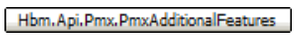
```
DaqEnvironment daqEnvironment = DaqEnvironment.GetInstance(); // Determine all plugable
DeviceFamily implementations
List<Device> devices=daqEnvironment.Scan(new List<String> { "PMX" }); // Scan system for
PMX devices only!
List<Problem> lastProblems;// Used to return warnings and errors
daqEnvironment.Connect(devices[0], out lastProblems);  // Connect to first found device

//Special PMX function: Load parameter set 10 of connected device:
if( (devices[0]as PmxDevice).AdditionalFeatures.LoadParameterSet(10) )
{
  MessageBox.Show("Parameterset 10 has been loaded!");
}
else
{
  MessageBox.Show("Error occurred: Could not load parameterset 10!");
}
...
//Assert to call DaqEnvironment.Dispose() before program ends.
daqEnvironment.Dispose();
...
```

クラス

名前	説明
PmxAdditionalFeatures	PmxDeviceのすべての追加機能を保持するクラス。追加機能は、Deviceクラスの抽象関数によって事前に定義されていないすべてのパブリック関数をカバーします。

PmxAdditionalFeatures Class

PmxDeviceのすべての追加機能を保持するクラス。追加機能は、Deviceクラスの抽象関数によって事前に定義されていないすべてのパブリック関数をカバーします。



C++

```
class PmxAdditionalFeatures;
```

C#

```
public class PmxAdditionalFeatures;
```

Visual Basic

```
Public Class PmxAdditionalFeatures
```

ファイル

PmxAdditionalFeatures.cs

PmxAdditionalFeatures Members

次の表は、PmxAdditionalFeaturesによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ActivateTEDS	指定したコネクタでTEDS設定をロードしてアクティブ化します。
	AssignLimitSwitch	LimitSwitch設定をデバイスに割り当てます。
	ClearPeakValues	指定した信号のピーク値(最小、最大、ピークツーピーク)をクリアします。
	GetCurrentParameterSetNumber	現在のパラメータセット番号(最近ロードされたもの)を返します。
	GetDeviceStatus	デバイスの現在のステータスを返します。
	GetDigitalOutBitMask	PMXデバイスのデジタル出力ポートを読み取ります(16ビット)
	GetEngineeringUnit	指定した信号の工学単位を取得します。
	GetLimitSwitch	デバイスからLimitSwitch設定を取得します。
	GetMaxValue	指定した信号の最大値を取得します。
	GetMinValue	指定した信号の最小値を取得します。
	GetPeakToPeakValue	指定した信号のピークツーピーク値を取得します。
	HoldPeakValues	信号のピーク値機能を保持または有効にします。
	LoadParameterSet	指定したパラメータセット番号を読み込んでアクティブ化します。
	PmxAdditionalFeatures	コンストラクタ。
	ReceiveReply	以前送信したコマンドの応答を受信します。
	SendCommand	PMXデバイスに直接コマンドを送信し、デバイスの応答を文字列として返します。この関数をバイナリアクセスに使用しないでください(連続測定機能の更新など)。
	SendCommandWithoutReply	応答を待たずに、コマンドをデバイスに直接送信します。

PmxAdditionalFeatures.PmxAdditionalFeatures Constructor

コンストラクタ。

C++

```
PmxAdditionalFeatures(
    PmxDevice pmxDevice
);
```

C#

```
public PmxAdditionalFeatures(
    PmxDevice pmxDevice
);
```

Visual Basic

```
Public Sub New(
    pmxDevice As PmxDevice
)
```

引数

引数	説明
pmxDevice	追加機能をサポートするPmxDevice。

PmxAdditionalFeatures Methods

PmxAdditionalFeaturesクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ActivateTEDS	指定したコネクタでTEDS設定をロードしてアクティブ化します。
	AssignLimitSwitch	LimitSwitch設定をデバイスに割り当てます。
	ClearPeakValues	指定した信号のピーク値(最小、最大、ピークツーピーク)をクリアします。
	GetCurrentParameterSetNumber	現在のパラメータセット番号(最近ロードされたもの)を返します。
	GetDeviceStatus	デバイスの現在のステータスを返します。
	GetDigitalOutBitMask	PMXデバイスのデジタル出力ポートを読み取ります(16ビット)。
	GetEngineeringUnit	指定した信号の工学単位を取得します。
	GetLimitSwitch	デバイスからLimitSwitch設定を取得します。
	GetMaxValue	指定した信号の最大値を取得します。
	GetMinValue	指定した信号の最小値を取得します。
	GetPeakToPeakValue	指定した信号のピークツーピーク値を取得します。
	HoldPeakValues	信号のピーク値機能を保持または有効にします。
	LoadParameterSet	指定したパラメータセット番号を読み込んでアクティブ化します。
	ReceiveReply	以前送信したコマンドの応答を受信します。
	SendCommand	PMXデバイスに直接コマンドを送信し、デバイスの応答を文字列として返します。この関数をバイナリアクセスに使用しないでください(連続測定機能の更新など)。
	SendCommandWithoutReply	応答を待たずに、コマンドをデバイスに直接送信します。

PmxAdditionalFeatures.ActivateTEDS Method

指定したコネクタでTEDS設定をロードしてアクティブ化します。

C++

```
[Obsolete( "Please use the common device function ActivateTeds(...) instead.", false)]
bool ActivateTEDS(
    Connector connectorAtWhichToActivateTEDS
);
```

C#

```
[Obsolete( "Please use the common device function ActivateTeds(...) instead.", false)]
public bool ActivateTEDS(
    Connector connectorAtWhichToActivateTEDS
);
```

Visual Basic

```
<Obsolete( "Please use the common device function ActivateTeds(...) instead.", false)>
Public Function ActivateTEDS(
    connectorAtWhichToActivateTEDS As Connector
) As bool
```

引数

引数	説明
connectorAtWhichToActivateTEDS	

戻り値

TEDS設定を読み込んでアクティブ化できる場合は`true`。

備考

成功した場合、指定したコネクタのセンサオブジェクトは、TEDS設定に従って更新されたバージョンに置き換えられます。

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
サポートされていないPMXの機能 -> PmxAdditionalFeatures Class ->
PmxAdditionalFeatures.PmxAdditionalFeatures Constructor

PmxAdditionalFeatures.AssignLimitSwitch Method

LimitSwitch設定をデバイスに割り当てます。

C++

```
bool AssignLimitSwitch(
    LimitSwitch limitSwitch
);
```

C#

```
public bool AssignLimitSwitch(
    LimitSwitch limitSwitch
);
```

Visual Basic

```
Public Function AssignLimitSwitch(
    limitSwitch As LimitSwitch
) As bool
```

引数

引数	説明
limitSwitch	割り当てるLimitSwitchオブジェクト。

戻り値

割り当てが成功した場合は`true`、それ以外の場合は `false`。

備考

使用中のLimitSwitchNumberのリミットスイッチに割り当てる場合、既存のリミットスイッチは上書きされます。

PmxAdditionalFeatures.ClearPeakValues Method

指定した信号のピーク値(最小、最大、ピークツーピーク)をクリアします。

C++

```
void ClearPeakValues(
    Signal signal,
    bool clearMin,
    bool clearMax,
    bool clearPeakToPeak
);
```

C#

```
public void ClearPeakValues(
    Signal signal,
    bool clearMin,
    bool clearMax,
    bool clearPeakToPeak
);
```

Visual Basic

```
Public Function ClearPeakValues(
    signal As Signal,
    clearMin As bool,
    clearMax As bool,
    clearPeakToPeak As bool
) As void
```

引数

引数	説明
signal	ピーク値を保持する信号。
clearMin	最小値をクリアする場合はtrueに設定します。
clearMax	最大値をクリアする場合はtrueに設定します。
clearPeakToPeak	ピークツーピーク値をクリアする場合はtrueに設定します。

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
サポートされていないPMXの機能 -> PmxAdditionalFeatures Class ->
PmxAdditionalFeatures.PmxAdditionalFeatures Constructor

PmxAdditionalFeatures.GetCurrentParameterSetNumber Method

現在のパラメータセット番号(最近ロードされたもの)を返します。

C++

```
int GetCurrentParameterSetNumber();
```

C#

```
public int GetCurrentParameterSetNumber();
```

Visual Basic

```
Public Function GetCurrentParameterSetNumber() As Integer
```

PmxAdditionalFeatures.GetDeviceStatus Method

デバイスの現在のステータスを返します。

C++

```
PmxDeviceStatus GetDeviceStatus();
```

C#

```
public PmxDeviceStatus GetDeviceStatus();
```

Visual Basic

```
Public Function GetDeviceStatus() As PmxDeviceStatus
```

戻り値

デバイスのステータス。Enums.PmxDeviceStatusを参照してください。

PmxAdditionalFeatures.GetDigitalOutBitMask Method

PMXデバイスのデジタル出力ポートを読み取ります(16ビット)。

C++

```
UInt16 GetDigitalOutBitMask();
```

C#

```
public UInt16 GetDigitalOutBitMask();
```

Visual Basic

```
Public Function GetDigitalOutBitMask() As UInt16
```

戻り値

PMXの16デジタル出力のビットマスク。

PmxAdditionalFeatures.GetEngineeringUnit Method

指定した信号の工学単位を取得します。

C++

```
[Obsolete("Workaround until we find a position to place the engineering unit for analog in
signals AND virtual signals within the common object model.")]
string GetEngineeringUnit(
    Signal signal
);
```

C#

```
[Obsolete("Workaround until we find a position to place the engineering unit for analog in
signals AND virtual signals within the common object model.")]
public string GetEngineeringUnit(
    Signal signal
);
```

Visual Basic

```
<Obsolete("Workaround until we find a position to place the engineering unit for analog in
signals AND virtual signals within the common object model.")>
Public Function GetEngineeringUnit(
    signal As Signal
) As string
```

引数

引数	説明
signal	工学単位を取得する信号。

戻り値

指定した信号の工学単位。

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
サポートされていないPMXの機能 -> PmxAdditionalFeatures Class ->
PmxAdditionalFeatures.PmxAdditionalFeatures Constructor

PmxAdditionalFeatures.GetLimitSwitch Method

デバイスからLimitSwitch設定を取得します。

C++

```
LimitSwitch GetLimitSwitch(
    int limitSwitchNumber
);
```

C#

```
public LimitSwitch GetLimitSwitch(
    int limitSwitchNumber
);
```

Visual Basic

```
Public Function GetLimitSwitch(
    limitSwitchNumber As Integer
) As LimitSwitch
```

引数

引数	説明
limitSwitchNumber	取得するリミットスイッチの数(1..32)。

戻り値

要求数に応じたリミットスイッチ設定。

PmxAdditionalFeatures.GetMaxValue Method

指定した信号の最大値を取得します。

C++

```
double GetMaxValue(
    Signal signal
);
```

C#

```
public double GetMaxValue(
    Signal signal
);
```

Visual Basic

```
Public Function GetMaxValue(
    signal As Signal
) As double
```

引数

引数	説明
signal	

戻り値

指定した信号の最大値。

PmxAdditionalFeatures.GetMinValue Method

指定した信号の最小値を取得します。

C++

```
double GetMinValue(
    Signal signal
);
```

C#

```
public double GetMinValue(
    Signal signal
);
```

Visual Basic

```
Public Function GetMinValue(
    signal As Signal
) As double
```

引数

引数	説明
signal	

戻り値

指定した信号の最小値。

PmxAdditionalFeatures.GetPeakToPeakValue Method

指定した信号のピークツーピーク値を取得します。

C++

```
double GetPeakToPeakValue(
    Signal signal
);
```

C#

```
public double GetPeakToPeakValue(
    Signal signal
);
```

Visual Basic

```
Public Function GetPeakToPeakValue(
    signal As Signal
) As double
```

引数

引数	説明
signal	

戻り値

指定した信号の最小値。

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
サポートされていないPMXの機能 -> PmxAdditionalFeatures Class ->
PmxAdditionalFeatures.PmxAdditionalFeatures Constructor

PmxAdditionalFeatures.HoldPeakValues Method

信号のピーク値機能を保持または有効にします。

C++

```
void HoldPeakValues(
    Signal signal,
    bool holdMin,
    bool holdMax,
    bool holdPeakToPeak
);
```

C#

```
public void HoldPeakValues(
    Signal signal,
    bool holdMin,
    bool holdMax,
    bool holdPeakToPeak
);
```

Visual Basic

```
Public Function HoldPeakValues(
    signal As Signal,
    holdMin As bool,
    holdMax As bool,
    holdPeakToPeak As bool
) As void
```

引数

引数	説明
signal	ピーク値を有効/無効にする信号。
holdMin	最小値を固定する場合はtrueに設定します。
holdMax	最大値を固定する場合はtrueに設定します。
holdPeakToPeak	ピークツーピーク値を固定する場合はtrueに設定します。

PmxAdditionalFeatures.LoadParameterSet Method

指定したパラメータセット番号を読み込んでアクティブ化します。

C++

```
bool LoadParameterSet(
    int parameterSetNumber
);
```

C#

```
public bool LoadParameterSet(
    int parameterSetNumber
);
```

Visual Basic

```
Public Function LoadParameterSet(
    parameterSetNumber As Integer
) As bool
```

引数

引数	説明
parameterSetNumber	ロードするパラメータ セット(0,1,2、...)。-1は、工場出荷時の設定を現在アクティブなパラメータセットにロードしてアクティブにすることを意味します。

戻り値

パラメータ セットが存在し、ロードされている場合は`true`。

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
サポートされていないPMXの機能 -> PmxAdditionalFeatures Class ->
PmxAdditionalFeatures.PmxAdditionalFeatures Constructor

PmxAdditionalFeatures.ReceiveReply Method

以前送信したコマンドの応答を受信します。

C++

```
string ReceiveReply();
```

C#

```
public string ReceiveReply();
```

Visual Basic

```
Public Function ReceiveReply() As string
```

戻り値

デバイスからの応答。

PmxAdditionalFeatures.SendCommand Method

PMXデバイスに直接コマンドを送信し、デバイスの応答を文字列として返します。この関数をバイナリアクセスに使用しないでください(連続測定機能の更新など)。

C++

```
string SendCommand(
    string directAccessCommand
);
```

C#

```
public string SendCommand(
    string directAccessCommand
);
```

Visual Basic

```
Public Function SendCommand(
    directAccessCommand As string
) As string
```

引数

引数	説明
directAccessCommand	

戻り値

directAccessCommandに対するデバイスの応答。

PmxAdditionalFeatures.SendCommandWithoutReply Method

応答を待たずに、コマンドをデバイスに直接送信します。

C++

```
void SendCommandWithoutReply(
    string directAccessCommand
);
```

C#

```
public void SendCommandWithoutReply(
    string directAccessCommand
);
```

Visual Basic

```
Public Function SendCommandWithoutReply(
    directAccessCommand As string
) As void
```

引数

引数	説明
directAccessCommand	送信コマンド。

Hbm.Api.Pmx Namespace

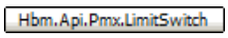
これは名前空間Hbm.Api.Pmxです。

クラス

名前	説明
LimitSwitch	リミットスイッチを記述するクラス。
PmxDevice	PMXデバイスを表します。
PmxDeviceFamily	PMXデバイスの具体的なデバイスファミリの実装。

LimitSwitch Class

リミットスイッチを記述するクラス。



C++

```
class LimitSwitch;
```

C#

```
public class LimitSwitch;
```

Visual Basic

```
Public Class LimitSwitch
```

ファイル

LimitSwitch.cs

備考

PMXは最大32個のリミットスイッチをサポートします。

LimitSwitch Members

次の表は、LimitSwitchによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	Enabled	リミットスイッチの動作を有効にするかどうかを決定します。
	Hysteresis	ヒステリシス値。`OperationDirection`に依存し、バンドスパンを定義にも使用できます。
	IgnoreMeasurementValueStatus	リミットスイッチステータスの評価時に測定値のステータスを無視するかどうかを決定します。
	InputSignal	リミットスイッチを評価するために測定値が使用する信号。
	InvertResetBehaviour	trueの場合、定義された`ResetBehavior`が反転されます。
	Limit	制限値。`OperationDirection`に依存し、下限バンド値を定義するためにも使用できます。
	LimitSwitchNumber	リミットスイッチの番号。範囲は1～32です。
	OperatingDirection	リミットスイッチの動作方向。
	ResetBehaviorMask	リセット動作。すべてのデジタル入力とAND演算されるバイナリマスク。リセット動作は`InvertResetBehavior`で反転できます。デフォルト設定は0です。

LimitSwitch Properties

LimitSwitchクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	Enabled	リミットスイッチの動作を有効にするかどうかを決定します。
	Hysteresis	ヒステリシス値。`OperationDirection`に依存し、バンドスパンを定義にも使用できます。
	IgnoreMeasurementValueStatus	リミットスイッチステータスの評価時に測定値のステータスを無視するかどうかを決定します。
	InputSignal	リミットスイッチを評価するために測定値が使用する信号。
	InvertResetBehaviour	trueの場合、定義された`ResetBehavior`が反転されます。
	Limit	制限値。`OperationDirection`に依存し、下限バンド値を定義するためにも使用できます。
	LimitSwitchNumber	リミットスイッチの番号。範囲は1～32です。
	OperatingDirection	リミットスイッチの動作方向。
	ResetBehaviorMask	リセット動作。すべてのデジタル入力とAND演算されるバイナリマスク。リセット動作は`InvertResetBehavior`で反転できます。デフォルト設定は0です。

LimitSwitch.Enabled Property

リミットスイッチの動作を有効にするかどうかを決定します。

C++

```
__property bool Enabled;
```

C#

```
public bool Enabled;
```

Visual Basic

```
Public Property Enabled() As bool
```

備考

リミットスイッチ操作が有効な場合は`true`、それ以外の場合は`false`。

LimitSwitch.Hysteresis Property

ヒステリシス値。OperationDirectionに依存し、バンドスパンを定義にも使用できます。

C++

```
__property double Hysteresis;
```

C#

```
public double Hysteresis;
```

Visual Basic

```
Public Property Hysteresis() As double
```

LimitSwitch.IgnoreMeasurementValueStatus Property

リミットスイッチステータスの評価時に測定値のステータスを無視するかどうかを決定します。

C++

```
__property bool IgnoreMeasurementValueStatus;
```

C#

```
public bool IgnoreMeasurementValueStatus;
```

Visual Basic

```
Public Property IgnoreMeasurementValueStatus() As bool
```

備考

`true`に設定すると、リミットスイッチステータスの評価時に測定値のステータスは無視されます。`false`に設定すると、測定値のステータスがOKの場合にのみリミットスイッチが評価されます。それ以外の場合、リミットスイッチステータスは固定されます。

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
Hbm.Api.Pmx Namespace -> LimitSwitch Class ->
LimitSwitch Members

LimitSwitch.InputSignal Property

リミットスイッチを評価するために測定値が使用する信号。

C++

```
__property Signal InputSignal;
```

C#

```
public Signal InputSignal;
```

Visual Basic

```
Public Property InputSignal() As Signal
```

LimitSwitch.InvertResetBehaviour Property

trueの場合、定義されたResetBehaviorが反転されます。

C++

```
__property bool InvertResetBehaviour;
```

C#

```
public bool InvertResetBehaviour;
```

Visual Basic

```
Public Property InvertResetBehaviour() As bool
```

LimitSwitch.Limit Property

制限値。OperationDirectionに依存し、下限バンド値を定義するためにも使用できます。

C++

```
__property double Limit;
```

C#

```
public double Limit;
```

Visual Basic

```
Public Property Limit() As double
```

LimitSwitch.LimitSwitchNumber Property

リミットスイッチの番号。範囲は1～32です。

C++

```
__property int LimitSwitchNumber;
```

C#

```
public int LimitSwitchNumber;
```

Visual Basic

```
Public Property LimitSwitchNumber() As Integer
```

LimitSwitch.OperatingDirection Property

リミットスイッチの動作方向。

C++

```
__property LimitSwitchOperatingDirection OperatingDirection;
```

C#

```
public LimitSwitchOperatingDirection OperatingDirection;
```

Visual Basic

```
Public Property OperatingDirection() As LimitSwitchOperatingDirection
```

LimitSwitch.ResetBehaviorMask Property

リセット動作。すべてのデジタル入力とAND演算されるバイナリマスク。リセット動作は`InvertResetBehavior`で反転できます。デフォルト設定は0です。

C++

```
__property int ResetBehaviorMask;
```

C#

```
public int ResetBehaviorMask;
```

Visual Basic

```
Public Property ResetBehaviorMask() As Integer
```

備考

結果が0(true)より大きい場合、このリミットスイッチはクリアされます。値の範囲: 0、1、2、4、8、16、...32768は、PMXの16個のデジタル入力に対応します。ヒステリシス管理に使用できます。

PmxDevice Class

PMXデバイスを表します。



C++

```
class PmxDevice : public Device;
```

C#

```
public class PmxDevice : Device;
```

Visual Basic

```
Public Class PmxDevice
Inherits Device
```

ファイル

PmxDevice.cs

PmxDevice Members

次の表は、PmxDeviceによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	LogContextDevPmxStatus	特定のデバイスに分類ステータスログメッセージを記録するLog Context。内部使用のみ。

Protected Methods

	名前	説明
	HandleChannelsChangesTriggeredByTedsActivation	TEDSのアクティブ化によってトリガされるチャネルの変更を処理します。
	MapConnectionInfo	指定したConnectionInfoをデバイスの可能なConnectionInfoにマップします。これが不可能な場合、関数はnullを返します。
	Reconnect	すでに部分的にまたは完全に入力されているオブジェクトモデル(Connectorなど)を使用して、基礎となる物理デバイスに再接続します。すでに使用可能なコネクタは物理デバイスに書き込まれ、不足しているコネクタは元のオブジェクトインスタンスを保持することでデバイスから埋め戻されます。
	RefreshDeviceInfo	名前、シリアル番号などの関連するデバイスプロパティを更新します。すべてのコネクタを読み取らずにデバイス自体に関する情報を取得する場合は、この関数を使用します。

Public Methods

	名前	説明
	ActivateTeds	指定したコネクタのTEDS設定をアクティブにし、新しいチャネルパラメータをロードします。
	AssignAutoCalibrationMode	指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignChannel	指定したチャネルの設定(センサ、信号などを含む)をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignConnector	指定したコネクタの設定(チャネル、センサ信号などを含む)をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSensor	指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSignal	指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignTimeSource	TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignZero	指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	ChangeIp	デバイスのipAddressを変更します。
	FlashLed	これは、FlashLedメソッドの多重定義の概要です。
	GetAmplifierSerialNumber	チャネルのアンプのシリアル番号を取得します。
	GetAmplifierType	チャネルのアンプタイプを取得します。
	GetCalibrationCertificateFileInfos	要求したリリースの情報に応じて、校正証明ファイル情報のリストを返します。校正証明書を作成し、指定した出力ディレクトリにダウンロードします。

	GetChannelHardwareInfo	チャネルのハードウェア情報(アンプの種類など)を取得します。
	GetConnectorOfSlotAndPosition	指定されたスロットと位置に属するコネクタを検索します。
	GetDeviceErrorStatusList	これは、GetDeviceErrorStatusListメソッドの多重定義の概要です。
	GetDeviceStatusList	これは、GetDeviceStatusListメソッドの多重定義の概要です。
	GetPositionNumberOfSignal	信号の種類に関係なく、信号の位置番号を返します。
	GetPossibleDataAcquisitionModes	デバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブル経由で確立)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedを含みます。デバイスが時間ベースの同期(PTPまたはNTPなど)を使用している場合、返されるリストにはDataAcquisitionMode.TimestampSynchronizedモードを含みます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはDataAcquisitionMode.Unsynchronizedモードを含む可能性が高くなります。デバイスが接続されていない場合、使用可能なデータ収集モードのリストは空になります。
	GetShunt	これは、GetShuntメソッドの多重定義の概要です。
	GetSignalOfSlotAndPosition	指定したスロットと位置に属する信号を検索します。
	GetSlotNumberOfSignal	信号の種類に関係なく、信号のスロット番号を返します。
	GetSupportedCanBitRates	サポートしているCANのビットレートを取得します。
	GetSupportedFilterFrequencies	指定したフィルタのサポートされているフィルタ周波数のリストを取得します。
	GetSupportedFilters	指定した信号のサポートされているフィルタのリストを取得します。
	GetSupportedJackType	指定した信号でサポートしているジャックの種類を取得します。
	GetSupportedSampleRates	サポートしているサンプルレートのリストを取得します。
	GetSupportedScalingTypes	指定したチャネルでサポートしているスケーリングタイプのリストを取得します。
	GetSupportedSensorTypes	指定したチャネルでサポートしているセンサタイプのリストを取得します。
	GetTimeSourceQuality	オフセット(同期タイムソースとデバイスの時間の間のタイムスパン)をミリ秒単位で返し、同期の品質に関する詳細情報を含むTimeSourceに依存する文字列を返します。NtpTimeSourceの場合、この文字列にはパラメータ(カンマ区切り)が含まれます: "remote、refid、st、t、when、pol、reach、delay、offset、jitter"。PtpTimeSourceの場合、この文字列には次のパラメータ(カンマ区切り)が含まれます: "GrandmasterID、SyncMode"、TimeScaleFlags、UtcOffset、UtcOffsetused、MasterOffset"
	IsFirmwareVersionSupported	指定したファームウェアのバージョンをサポートしているかどうかを判断します。
	PmxDevice	これは、PmxDeviceコンストラクタの多重定義の概要です。
	ReadSingleMeasurementValue	DAQセッションを完全に初期化せずに、指定したSignalごとの単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadSingleMeasurementValueOfAllSignals	DAQセッションを完全に初期化せずに、デバイスのすべてのSignalに対して単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadTeds	指定したコネクタに接続されているTEDSからセンサを読み取ります。
	RemoveInputSignalReference	指定したチャネルから入力信号のリファレンスを削除します。

	Reset	デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数の呼び出し後に、デバイスは切断されます。これが必要なのはデバイスのパラメータ設定が変更され、デバイスがデバイスの関連オブジェクトモデルと一致しなくなったとき、工場出荷時のデフォルトをロードします(この関数を実装するときは、必ずデバイスを切断し、Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知します。
	SetAnalogOut	アナログ出力信号を指定した値に設定します。
	SetCanBusParameters	指定したCanConnectorのすべてのCAN busパラメータ(CanConnector.BitRate、CanConnector.IsTerminated、CanConnector.SupplyVoltageなど)を割り当てますが、チャネルは割り当てません。
	SetCanChannelActivation	提供されたチャネルのリストをアクティブまたは非アクティブにします。CanInChannelとCanOutChannelのみをアクティブ化または非アクティブ化できます。
	SetChannelName	指定したチャネルにチャネル名を割り当てます。
	SetDataRateDomain	dataRateDomainを設定します。
	SetDeviceName	デバイスの名前を設定します。PMXデバイスの名前を変更することはできません。
	SetDigitalIO	指定した値に従ってデジタル信号を設定します。
	SetInputSignalReference	入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。
	SetShunt	これは、SetShuntメソッドの多重定義の概要です。
	SetSignalName	指定した信号に信号名を割り当てます。
	SetTedsUsageMode	TEDS使用モードを設定し、コネクタ設定を再読み込みします。
	SetUnit	指定したチャネルの単位を割り当てます。
	WriteTeds	指定したコネクタに接続されたTEDSに指定したセンサを書き込みます。

Public Properties

	名前	説明
	AdditionalFeatures	PMXデバイスの追加機能。
R	FamilyName	このデバイスが属するファミリの名前。

PmxDevice Constructor

これは、PmxDeviceコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	PmxDevice.PmxDevice ()	クラスの新しいインスタンスを作成します。
	PmxDevice.PmxDevice (string, int)	クラスの新しいインスタンスを作成します。

PmxDevice.PmxDevice Constructor ()

クラスの新しいインスタンスを作成します。

C++

```
PmxDevice();
```

C#

```
public PmxDevice();
```

Visual Basic

```
Public Sub New()
```

PmxDevice.PmxDevice Constructor (string, int)

クラスの新しいインスタンスを作成します。

C++

```
PmxDevice(
    string ip4Address,
    int port = PMXDEFAULTPORT
);
```

C#

```
public PmxDevice(
    string ip4Address,
    int port
);
```

Visual Basic

```
Public Sub New(
    ip4Address As string,
    port As Integer = PMXDEFAULTPORT
)
```

引数

引数	説明
ip4Address	デバイスのIPアドレス。
port	デバイス接続用のポート。

PmxDevice Methods

PmxDeviceクラスのメソッドの一覧です。

Protected Methods

	名前	説明
	HandleChannelsChangesTriggeredByTedsActivation	TEDSのアクティブ化によってトリガされるチャネルの変更を処理します。
	MapConnectionInfo	指定したConnectionInfoをデバイスの可能なConnectionInfoにマップします。これが不可能な場合、関数はnullを返します。
	Reconnect	すでに部分的にまたは完全に入力されているオブジェクトモデル(Connectorなど)を使用して、基礎となる物理デバイスに再接続します。すでに使用可能なコネクタは物理デバイスに書き込まれ、不足しているコネクタは元のオブジェクトインスタンスを保持することでデバイスから埋め戻されます。
	RefreshDeviceInfo	名前、シリアル番号などの関連するデバイスプロパティを更新します。すべてのコネクタを読み取らずにデバイス自体に関する情報を取得する場合は、この関数を使用します。

Public Methods

	名前	説明
	ActivateTeds	指定したコネクタのTEDS設定をアクティブにし、新しいチャネルパラメータをロードします。
	AssignAutoCalibrationMode	指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignChannel	指定したチャネルの設定(センサ、信号などを含む)をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignConnector	指定したコネクタの設定(チャネル、センサ信号などを含む)をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSensor	指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSignal	指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignTimeSource	TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignZero	指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	ChangeIp	デバイスのipAddressを変更します。
	FlashLed	これは、FlashLedメソッドの多重定義の概要です。
	GetAmplifierSerialNumber	チャネルのアンプのシリアル番号を取得します。
	GetAmplifierType	チャネルのアンプタイプを取得します。
	GetCalibrationCertificateFileInfos	要求したリリースの情報に応じて、校正証明ファイル情報のリストを返します。校正証明書を作成し、指定した出力ディレクトリにダウンロードします。
	GetChannelHardwareInfo	チャネルのハードウェア情報(アンプの種類など)を取得します。
	GetConnectorOfSlotAndPosition	指定されたスロットと位置に属するコネクタを検索します。
	GetDeviceErrorStatusList	これは、GetDeviceErrorStatusListメソッドの多重定義の概要です。

	GetDeviceStatusList	これは、GetDeviceStatusListメソッドの多重定義の概要です。
	GetPositionNumberOfSignal	信号の種類に関係なく、信号の位置番号を返します。
	GetPossibleDataAcquisitionModes	デバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブル経由で確立)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedを含みます。デバイスが時間ベースの同期(PTPまたはNTPなど)を使用している場合、返されるリストにはDataAcquisitionMode.TimestampSynchronizedモードを含みます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはDataAcquisitionMode.Unsynchronizedモードを含む可能性が高くなります。デバイスが接続されていない場合、使用可能なデータ収集モードのリストは空になります。
	GetShunt	これは、GetShuntメソッドの多重定義の概要です。
	GetSignalOfSlotAndPosition	指定したスロットと位置に属する信号を検索します。
	GetSlotNumberOfSignal	信号の種類に関係なく、信号のスロット番号を返します。
	GetSupportedCanBitRates	サポートしているCANのビットレートを取得します。
	GetSupportedFilterFrequencies	指定したフィルタのサポートされているフィルタ周波数のリストを取得します。
	GetSupportedFilters	指定した信号のサポートされているフィルタのリストを取得します。
	GetSupportedJackType	指定した信号でサポートしているジャックの種類を取得します。
	GetSupportedSampleRates	サポートしているサンプルレートのリストを取得します。
	GetSupportedScalingTypes	指定したチャネルでサポートしているスケーリングタイプのリストを取得します。
	GetSupportedSensorTypes	指定したチャネルでサポートしているセンサタイプのリストを取得します。
	GetTimeSourceQuality	オフセット(同期タイムソースとデバイスの時間の間のタイムスパン)をミリ秒単位で返し、同期の品質に関する詳細情報を含むTimeSourceに依存する文字列を返します。NtpTimeSourceの場合、この文字列にはパラメータ(カンマ区切り)が含まれます: "remote、refid、st、t、when、pol、reach、delay、offset、jitter"。PtpTimeSourceの場合、この文字列には次のパラメータ (カンマ区切り)が含まれます: "GrandmasterID、SyncMode"、TimeScaleFlags、UtcOffset、UtcOffsetused、MasterOffset"
	IsFirmwareVersionSupported	指定したファームウェアのバージョンをサポートしているかどうかを判断します。
	ReadSingleMeasurementValue	DAQセッションを完全に初期化せずに、指定したSignalごとの単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadSingleMeasurementValueOfAllSignals	DAQセッションを完全に初期化せずに、デバイスのすべてのSignalに対して単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadTeds	指定したコネクタに接続されているTEDSからセンサを読み取ります。
	RemoveInputSignalReference	指定したチャネルから入力信号のリファレンスを削除します。

	Reset	デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数の呼び出し後に、デバイスは切断されます。これが必要なのはデバイスのパラメータ設定が変更され、デバイスがデバイスの関連オブジェクトモデルと一致しなくなったとき、工場出荷時のデフォルトをロードします(この関数を実装するときは、必ずデバイスを切断し、Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知します。
	SetAnalogOut	アナログ出力信号を指定した値に設定します。
	SetCanBusParameters	指定したCanConnectorのすべてのCAN busパラメータ(CanConnector.BitRate、CanConnector.IsTerminated、CanConnector.SupplyVoltageなど)を割り当てますが、チャネルは割り当てません。
	SetCanChannelActivation	提供されたチャネルのリストをアクティブまたは非アクティブにします。CanInChannelとCanOutChannelのみをアクティブ化または非アクティブ化できます。
	SetChannelName	指定したチャネルにチャネル名を割り当てます。
	SetDataRateDomain	dataRateDomainを設定します。
	SetDeviceName	デバイスの名前を設定します。PMXデバイスの名前を変更することはできません。
	SetDigitalIO	指定した値に従ってデジタル信号を設定します。
	SetInputSignalReference	入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。
	SetShunt	これは、SetShuntメソッドの多重定義の概要です。
	SetSignalName	指定した信号に信号名を割り当てます。
	SetTedsUsageMode	TEDS使用モードを設定し、コネクタ設定を再読み込みします。
	SetUnit	指定したチャネルの単位を割り当てます。
	WriteTeds	指定したコネクタに接続されたTEDSに指定したセンサを書き込みます。

PmxDevice.ActivateTeds Method

指定したコネクタのTEDS設定をアクティブにし、新しいチャネルパラメータをロードします。

C++

```
bool ActivateTeds(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override bool ActivateTeds(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function ActivateTeds(
    connector As Connector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	接続されたTEDSをアクティブ化するコネクタ。
problems	空きまたは警告リスト、およびアクティブ化プロセス中に発生したエラー。

戻り値

TEDS設定がアクティブ化されている場合は`true`、それ以外の場合は`false`。

備考

成功すると、指定したコネクタのセンサオブジェクトと一部のチャネルプロパティが、TEDS設定に従って更新されたバージョンに置き換えられます。

PmxDevice.AssignAutoCalibrationMode Method

指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignAutoCalibrationMode(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignAutoCalibrationMode(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignAutoCalibrationMode(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

PmxDevice.AssignChannel Method

指定したチャネルの設定(センサ、信号などを含む)をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignChannel(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignChannel(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignChannel(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル 。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	チャネルが`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

PmxDevice.AssignConnector Method

指定したコネクタの設定(チャネル、センサ信号などを含む)をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignConnector(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override bool AssignConnector(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignConnector(
    connector As Connector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	既存の(変更された)コネクタ。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

問題が発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	コネクタが`null`です。
Hbm.Api.Common.Exceptions.UnknownConnectorException	コネクタがこのデバイスに属していない場合にスローされます。

PmxDevice.AssignSensor Method

指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignSensor(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignSensor(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignSensor(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	信号が`null`です。
System.ArgumentException	チャネルがセンサをサポートしていない場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

PmxDevice.AssignSignal Method

指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignSignal(
    Signal signal,
    out List<Problem> problems
);
```

C#

```
public override bool AssignSignal(
    Signal signal,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignSignal(
    signal As Signal,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
signal	既存の(変更された)信号。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	信号が`null`です。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

PmxDevice.AssignTimeSource Method

TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignTimeSource(
    out List<Problem> problems
);
```

C#

```
public override bool AssignTimeSource(
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignTimeSource(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

PmxDevice.AssignZero Method

指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignZero(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignZero(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignZero(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	チャネルが`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

PmxDevice.ChangeIp Method

デバイスのipAddressを変更します。

C++

```
bool ChangeIp(
    string ipAddress,
    string subnetMask,
    bool isDhcp,
    string gateWay,
    unsigned tcpPort,
    string interfaceName,
    out List<Problem> problems
);
```

C#

```
public override bool ChangeIp(
    string ipAddress,
    string subnetMask,
    bool isDhcp,
    string gateWay,
    uint tcpPort,
    string interfaceName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function ChangeIp(
    ipAddress As string,
    subnetMask As string,
    isDhcp As bool,
    gateWay As string,
    tcpPort As UInteger,
    interfaceName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
ipAddress	適用するIPアドレス。
subnetMask	サブネットマスク。
isDhcp	自動割り当てであるかどうかを示す表示器。
gateWay	ゲートウェイ。
tcpPort	TCPポート。
interfaceName	ネットワークインタフェースの名前。
problems	問題のリスト。

戻り値

成功した場合はtrue、それ以外の場合はfalse。

FlashLed Method

これは、FlashLedメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	PmxDevice.FlashLed (Connector, LedFlashMode, out List<Problem>)	指定したコネクタのLEDを指定したモード/色で点滅させます。
	PmxDevice.FlashLed (LedFlashMode, out List<Problem>)	指定したモード/色でデバイスのLEDを点滅させます。
	PmxDevice.FlashLed (List<Connector>, LedFlashMode, out List<Problem>)	指定したコネクタのLEDを指定したモード/色で点滅させます。

PmxDevice.FlashLed Method (Connector, LedFlashMode, out List<Problem>)

指定したコネクタのLEDを指定したモード/色で点滅させます。

C++

```
bool FlashLed(
    Connector connector,
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool FlashLed(
    Connector connector,
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function ActivateTeds(
    connector As Connector,
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	点滅させるコネクタ。
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

PmxDevice.FlashLed Method (LedFlashMode, out List<Problem>)

指定したモード/色でデバイスのLEDを点滅させます。

C++

```
bool FlashLed(
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool FlashLed(
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function FlashLed(
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

PmxDevice.FlashLed Method (List<Connector>, LedFlashMode, out List<Problem>)

指定したコネクタのLEDを指定したモード/色で点滅させます。

C++

```
bool FlashLed(
    List<Connector> connectors,
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool FlashLed(
    List<Connector> connectors,
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function FlashLed(
    connectors As List<Connector>,
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connectors	点滅させるコネクタ。
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

PmxDevice.GetAmplifierSerialNumber Method

チャネルのアンプのシリアル番号を取得します。

C++

```
string GetAmplifierSerialNumber(
    Channel channel
);
```

C#

```
public override string GetAmplifierSerialNumber(
    Channel channel
);
```

Visual Basic

```
Public override Function GetAmplifierSerialNumber(
    channel As Channel
) As string
```

引数

引数	説明
channel	アンプシリアル番号を返すチャネル。

戻り値

指定したチャネルのアンプのシリアル番号。

例外

例外	説明
NotImplementedException	

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
Hbm.Api.Pmx Namespace -> PmxDevice Class ->
PmxDevice Methods

PmxDevice.GetAmplifierType Method

チャネルのアンプタイプを取得します。

C++

```
string GetAmplifierType(
    Channel channel
);
```

C#

```
public override string GetAmplifierType(
    Channel channel
);
```

Visual Basic

```
Public override Function GetAmplifierType(
    channel As Channel
) As string
```

引数

引数	説明
channel	アンプの種類を返すチャネル。

戻り値

指定したチャネルのアンプの種類。

PmxDevice.GetCalibrationCertificateFileInfos Method

要求したリリースの情報に応じて、校正証明ファイル情報のリストを返します。校正証明書を作成し、指定した出力ディレクトリにダウンロードします。

C++

```
List<CalibrationCertificateFileInfo> GetCalibrationCertificateFileInfos(
    ReleaseType releaseType,
    string outputDirectory,
    out List<Problem> problems
);
```

C#

```
public override List<CalibrationCertificateFileInfo> GetCalibrationCertificateFileInfos(
    ReleaseType releaseType,
    string outputDirectory,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetCalibrationCertificateFileInfos(
    releaseType As ReleaseType,
    outputDirectory As string,
    problems As out List<Problem>
) As List<CalibrationCertificateFileInfo>
```

引数

引数	説明
releaseType	要求された校正証明のリリースタイプ。
outputDirectory	校正証明書を保存/作成するディレクトリ。
problems	校正証明の取得中に発生した警告とエラー。

戻り値

校正証明書ファイル情報のリスト。

備考

一時的に作成された校正証明書(デバイスからダウンロード)は忘れずに削除してください。

PmxDevice.GetChannelHardwareInfo Method

チャネルのハードウェア情報(アンプの種類など)を取得します。

C++

```
string GetChannelHardwareInfo(
    Channel channel
);
```

C#

```
public override string GetChannelHardwareInfo(
    Channel channel
);
```

Visual Basic

```
Public override Function GetChannelHardwareInfo(
    channel As Channel
) As string
```

引数

引数	説明
channel	ハードウェア情報を返すチャネル。

戻り値

指定したチャネルのハードウェア情報。

PmxDevice.GetConnectorOfSlotAndPosition Method

指定したスロットと位置に属するコネクタを検索します。

C++

```
Connector GetConnectorOfSlotAndPosition(
    int slot,
    int position
);
```

C#

```
public Connector GetConnectorOfSlotAndPosition(
    int slot,
    int position
);
```

Visual Basic

```
Public Function GetConnectorOfSlotAndPosition(
    slot As Integer,
    position As Integer
) As Connector
```

引数

引数	説明
slot	コネクタが属しているスロット。
position	信号が属しているスロット内の位置。

戻り値

指定されたスロット/位置で見つかったコネクタ、または対応するコネクタが見つからなかった場合はnull。

GetDeviceErrorStatusList Method

これは、GetDeviceErrorStatusListメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	PmxDevice.GetDeviceErrorStatusList (Channel, out List<Problem>)	指定したチャネルのデバイスエラーステータスのリストを取得します。
	PmxDevice.GetDeviceErrorStatusList (Connector, out List<Problem>)	指定したコネクタのデバイスエラーステータスのリストを取得します。
	PmxDevice.GetDeviceErrorStatusList (out List<Problem>)	デバイスエラーステータスのリストを取得します。

PmxDevice.GetDeviceErrorStatusList Method (Channel, out List<Problem>)

指定したチャネルのデバイスエラーステータスのリストを取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceErrorStatusList(
    channel As Channel,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
channel	エラーステータスを取得するチャネル。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

指定したチャネルのデバイスエラーステータスのリスト。

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
Hbm.Api.Pmx Namespace -> PmxDevice Class ->
PmxDevice Methods -> PmxDevice.GetConnectorOfSlotAndPosition Method

PmxDevice.GetDeviceErrorStatusList Method (Connector, out List<Problem>)

指定したコネクタのデバイスエラーステータスのリストを取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceErrorStatusList(
    connector As Connector,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
connector	エラーステータスを取得するコネクタ。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

指定したコネクタのデバイスエラーステータスのリスト。

PmxDevice.GetDeviceErrorStatusList Method (out List<Problem>)

デバイスエラーステータスのリストを取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    out List<Problem> problems
);
```

C#

```
public override List<DeviceErrorStatus> GetDeviceErrorStatusList(
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceErrorStatusList(
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

デバイスエラーステータスのリスト。

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
Hbm.Api.Pmx Namespace -> PmxDevice Class ->
PmxDevice Methods -> PmxDevice.GetConnectorOfSlotAndPosition Method

GetDeviceStatusList Method

これは、GetDeviceStatusListメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	PmxDevice.GetDeviceStatusList (Channel, out List<Problem>)	指定したチャネルのデバイスステータスのリストを取得します。
	PmxDevice.GetDeviceStatusList (Connector, out List<Problem>)	指定したコネクタのデバイスステータスのリストを取得します。
	PmxDevice.GetDeviceStatusList (out List<Problem>, StatusType)	デバイスステータスのリストを取得します。

PmxDevice.GetDeviceStatusList Method (Channel, out List<Problem>)

指定したチャネルのデバイスステータスのリストを取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceStatus> GetDeviceStatusList(
    Channel channel
    out List<Problem> problems,
);
```

Visual Basic

```
Public override Function GetDeviceStatusList(
    channel As Channel,
    problems As out List<Problem>
) As List<DeviceStatus>
```

引数

引数	説明
channel	ステータスを取得するチャネル。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

指定したチャネルのデバイスステータスのリスト。

PmxDevice.GetDeviceStatusList Method (Connector, out List<Problem>)

指定したコネクタのデバイスステータスのリストを取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceStatus> GetDeviceStatusList(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceStatusList(
    connector As Connector,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
connector	ステータスを取得するコネクタ。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

指定したコネクタのデバイスステータスのリスト。

PmxDevice.GetDeviceStatusList Method (out List<Problem>, StatusType)

デバイスステータスのリストを取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    out List<Problem> problems,
    StatusType statusTypesToGet = StatusType.All
);
```

C#

```
public override List<DeviceStatus> GetDeviceStatusList(
    out List<Problem> problems,
    StatusType statusTypesToGet
);
```

Visual Basic

```
Public override Function GetDeviceStatusList(
    problems As out List<Problem>,
    statusTypesToGet As StatusType = StatusType.All
) As List<DeviceStatus>
```

引数

引数	説明
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。
statusTypesToGet	要求したステータスタイプを含むT:Hbm.Api.Common.Enums.StatusType(デフォルトはすべてのステータス情報)。

戻り値

指定したチャネルのデバイスステータスのリスト。

PmxDevice.GetPositionNumberOfSignal Method

信号の種類に関係なく、信号の位置番号を返します。

C++

```
int GetPositionNumberOfSignal(
    Signal signal
);
```

C#

```
public int GetPositionNumberOfSignal(
    Signal signal
);
```

Visual Basic

```
Public Function GetPositionNumberOfSignal(
    signal As Signal
) As Integer
```

引数

引数	説明
signal	位置を返す信号。

戻り値

信号の位置番号。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

PmxDevice.GetPossibleDataAcquisitionModes Method

デバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブル経由で確立)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedを含みます。デバイスが時間ベースの同期(PTPまたはNTPなど)を使用している場合、返されるリストにはDataAcquisitionMode.TimestampSynchronizedモードを含みます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはDataAcquisitionMode.Unsynchronizedモードを含む可能性が高くなります。デバイスが接続されていない場合、使用可能なデータ収集モードのリストは空になります。

C++

```
List<DataAcquisitionMode> GetPossibleDataAcquisitionModes();
```

C#

```
public override List<DataAcquisitionMode> GetPossibleDataAcquisitionModes();
```

Visual Basic

```
Public override Function GetPossibleDataAcquisitionModes() As List<DataAcquisitionMode>
```

戻り値

現在の同期設定に応じて、このデバイスに使用できるDataAcquisitionModeのリスト。

GetShunt Method

これは、GetShuntメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	PmxDevice.GetShunt (Connector)	指定したコネクタのShuntModeを取得します。
	PmxDevice.GetShunt (List<Connector>)	指定したコネクタのShuntModeを取得します。

PmxDevice.GetShunt Method (Connector)

指定したコネクタのShuntModeを取得します。

C++

```
ShuntMode GetShunt(
    Connector connector
);
```

C#

```
public override ShuntMode GetShunt(
    Connector connector
);
```

Visual Basic

```
Public override Function GetShunt(
   connector As Connector
) As ShuntMode
```

引数

引数	説明
connector	ShuntModeを取得するコネクタ。

戻り値

指定されたコネクタのShuntMode。

PmxDevice.GetShunt Method (List<Connector>)

指定したコネクタのShuntModeを取得します。

C++

```
List<ShuntMode> GetShunt(
    List<Connector> connectors
);
```

C#

```
public override List<ShuntMode> GetShunt(
    List<Connector> connectors
);
```

Visual Basic

```
Public override Function GetShunt(
   connectors As List<Connector>
) As List<ShuntMode>
```

引数

引数	説明
connectors	ShuntModeを取得するコネクタのリスト。

戻り値

指定されたコネクタのShuntModeのリスト。

PmxDevice.GetSignalOfSlotAndPosition Method

指定したスロットと位置に属する信号を検索します。

C++

```
Signal GetSignalOfSlotAndPosition(
    int slot,
    int position
);
```

C#

```
public Signal GetSignalOfSlotAndPosition(
    int slot,
    int position
);
```

Visual Basic

```
Public Function GetSignalOfSlotAndPosition(
    slot As Integer,
    position As Integer
) As Signal
```

引数

引数	説明
slot	信号が属しているスロット。
position	信号が属しているスロット内の位置。

戻り値

指定されたスロット/位置で見つかった信号、または対応する信号が見つからなかった場合はnull。

PmxDevice.GetSlotNumberOfSignal Method

信号の種類に関係なく、信号のスロット番号を返します。

C++

```
int GetSlotNumberOfSignal(
    Signal signal
);
```

C#

```
public int GetSlotNumberOfSignal(
    Signal signal
);
```

Visual Basic

```
Public Function GetSlotNumberOfSignal(
    signal As Signal
) As Integer
```

引数

引数	説明
signal	位置を返す信号。

戻り値

信号のスロット番号。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

PmxDevice.GetSupportedCanBitRates Method

サポートしているCANのビットレートを取得します。

C++

```
List<int> GetSupportedCanBitRates();
```

C#

```
public override List<int> GetSupportedCanBitRates();
```

Visual Basic

```
Public override Function GetSupportedCanBitRates() As List<int>
```

戻り値

サポートしている CANビットレートのリスト。

PmxDevice.GetSupportedFilterFrequencies Method

指定したフィルタのサポートされているフィルタ周波数のリストを取得します。

C++

```
List<decimal> GetSupportedFilterFrequencies(
    Filter Filter,
    bool isOfflineQuery = false
);
```

C#

```
public override List<decimal> GetSupportedFilterFrequencies(
    Filter Filter,
    bool isOfflineQuery
);
```

Visual Basic

```
Public override Function GetSupportedFilterFrequencies(
   Filter As Filter
   isOfflineQuery As bool = false
) As List<decimal>
```

引数

引数	説明
filter	取得したいフィルタの対象となるフィルタ周波数。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なフィルタ周波数のリスト。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownFilterException	フィルタがこのデバイスに属していない場合にスローされます。

PmxDevice.GetSupportedFilters Method

指定した信号のサポートされているフィルタのリストを取得します。

C++

```
List<Filter> GetSupportedFilters(
    Signal signal,
    bool isOffineQuery = false
);
```

C#

```
public override List<Filter> GetSupportedFilters(
    Signal signal,
    bool isOffineQuery
);
```

Visual Basic

```
Public override Function GetSupportedFilters(
   signal As Signal
   isOffineQuery As bool = false
) As List<Filter>
```

引数

引数	説明
signal	取得したいフィルタの対象となる信号。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なフィルタのリスト。

例外

例外	説明
System.ArgumentNullException	信号がnullです。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

PmxDevice.GetSupportedJackType Method

指定した信号でサポートしているジャックの種類を取得します。

C++

```
string GetSupportedJackType(
    Signal signal,
    bool isOfflineQuery = false
);
```

C#

```
public override string GetSupportedJackType(
    Signal signal,
    bool isOfflineQuery
);
```

Visual Basic

```
Public override Function GetSupportedJackType(
   signal As Signal
   isOfflineQuery As bool = false
) As List<Filter>
```

引数

引数	説明
signal	信号。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

ジャックの種類名。

例外

例外	説明
System.NotImplementedException	

PmxDevice.GetSupportedSampleRates Method

サポートしているサンプルレートのリストを取得します。

C++

```
List<decimal> GetSupportedSampleRates(
    Signal signal,
    bool isOffineQuery = false
);
```

C#

```
public override List<decimal> GetSupportedSampleRates(
    Signal signal,
    bool isOffineQuery
);
```

Visual Basic

```
Public override Function GetSupportedSampleRates(
   signal As Signal
   isOffineQuery As bool = false
) As List<decimal>
```

引数

引数	説明
signal	取得したいサンプルレートの信号。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なサンプルレートのリスト。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

PmxDevice.GetSupportedScalingTypes Method

指定したチャネルでサポートしているスケーリングタイプのリストを取得します。

C++

```
List<ScalingType> GetSupportedScalingTypes(
    Channel channel,
    bool isOfflineQuery = false
);
```

C#

```
public override List<ScalingType> GetSupportedScalingTypes(
    Channel channel,
    bool isOfflineQuery
);
```

Visual Basic

```
Public override Function GetSupportedScalingTypes(
    channel As Channel,
   isOfflineQuery As bool = false
) As List<ScalingType>
```

引数

引数	説明
channel	取得したいスケーリングタイプのチャネル。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なスケーリングタイプのリスト。

戻り値

可能なスケーリングタイプは、チャネルの現在のセンサによっても異なります。

例外

例外	説明
System.ArgumentNullException	チャネルが`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

PmxDevice.GetSupportedSensorTypes Method

指定したチャネルでサポートしているセンサタイプのリストを取得します。

C++

```
List<SensorType> GetSupportedSensorTypes(
    Channel channel,
    bool isOffineQuery = false
);
```

C#

```
public override List<SensorType> GetSupportedSensorTypes(
    Channel channel,
    bool isOffineQuery
);
```

Visual Basic

```
Public override Function GetSupportedSensorTypes(
    channel As Channel,
   isOffineQuery As bool = false
) As List<SensorType>
```

引数

引数	説明
channel	取得したいセンサタイプのチャネル。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なセンサタイプのリスト。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

PmxDevice.GetTimeSourceQuality Method

オフセット(同期タイムソースとデバイスの時間の間のタイムスパン)をミリ秒単位で返し、同期の品質に関する詳細情報を含むTimeSourceに依存する文字列を返します。NtpTimeSourceの場合、この文字列にはパラメータ(カンマ区切り)が含まれます: "remote、refid、st、t、when、pol、reach、delay、offset、jitter"。PtpTimeSourceの場合、この文字列には次のパラメータ (カンマ区切り)が含まれます: "GrandmasterID、SyncMode"、TimeScaleFlags、UtcOffset、UtcOffsetused、MasterOffset"

C++

```
void GetTimeSourceQuality(
    out double offsetInMs,
    out string quality
);
```

C#

```
public override void GetTimeSourceQuality(
    out double offsetInMs,
    out string quality
);
```

Visual Basic

```
Public override Function GetTimeSourceQuality(
    offsetInMs As out double,
    quality As out string
) As void
```

引数

引数	説明
offsetInMs	タイムソースへのオフセット(ミリ秒単位)。オフセットを決定できない場合は double.MaxValue になります。
quality	TimeSourceタイプに依存する同期品質に関する情報。

PmxDevice.HandleChannelsChangesTriggeredByTedsActivation Method

TEDSのアクティブ化によってトリガされるチャネルの変更を処理します。

C++

```
void HandleChannelsChangesTriggeredByTedsActivation(
    Dictionary<Connector, Channel> channels,
    List<Problem> problems
);
```

C#

```
public override void HandleChannelsChangesTriggeredByTedsActivation(
    Dictionary<Connector, Channel> channels,
    List<Problem> problems
);
```

Visual Basic

```
Public override Function HandleChannelsChangesTriggeredByTedsActivation(
    channels As Dictionary<Connector, Channel>,
    problems As List<Problem>
) As void
```

引数

引数	説明
channels	チャネル。
problems	警告とエラー。

PmxDevice.IsFirmwareVersionSupported Method

指定したファームウェアのバージョンをサポートしているかどうかを判断します。

C++

```
bool IsFirmwareVersionSupported(
    string firmwareVersion
);
```

C#

```
public override bool IsFirmwareVersionSupported(
    string firmwareVersion
);
```

Visual Basic

```
Public override Function IsFirmwareVersionSupported(
    firmwareVersion As string
) As bool
```

引数

引数	説明
firmwareVersion	確認するファームウェアのバージョン。

戻り値

指定したファームウェアバージョンがサポートされていない場合はfalseになります。

備考

この関数は、指定したファームウェアバージョンが最新の既知のバージョンよりも高い場合(APIがリリースされた時点)でもtrueを返します。

PmxDevice.MapConnectionInfo Method

指定したConnectionInfoをデバイスの可能なConnectionInfoにマップします。これが不可能な場合、関数はnullを返します。

C++

```
ConnectionInfo MapConnectionInfo(
    ConnectionInfo connectionInfo
);
```

C#

```
public override ConnectionInfo MapConnectionInfo(
    ConnectionInfo connectionInfo
);
```

Visual Basic

```
Public override Function MapConnectionInfo(
    connectionInfo As ConnectionInfo
) As ConnectionInfo
```

引数

引数	説明
connectionInfo	サポートされているConnectionInfoにマップするConnectionInfo。

戻り値

デバイスの有効なConnectionInfo、またはマッピングが不可能な場合はnull。

例

ConnectionInfoがEthernetConnectionInfoであるが、デバイスがStreamingConnectionInfoのみをサポートしている場合: 関数はEthernetConnectionInfoをStreamingConnectionInfoにマッピングし、不足しているパラメータ(可能な場合、デフォルトのストリーミングパラメータなど)を入力します。これが不可能な場合、関数はnullを返す必要があります。

PmxDevice.ReadSingleMeasurementValue Method

DAQセッションを完全に初期化せずに、指定したSignalごとの単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。

C++

```
void ReadSingleMeasurementValue(
    List<Signal> measurementSignals
);
```

C#

```
public override void ReadSingleMeasurementValue(
    List<Signal> measurementSignals
);
```

Visual Basic

```
Public override Function ReadSingleMeasurementValue(
    measurementSignals As List<Signal>
) As void
```

引数

引数	説明
measurementSignals	測定値を取得する信号。

備考

デバイスが単一の測定値のタイムスタンプをサポートしていない場合、PCの現在の時刻を使用してタイムスタンプを生成します。

例外

例外	説明
System.ArgumentNullException	計測信号がnullです。
System.ArgumentException	計測信号の数が0の場合にスローされます
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

PmxDevice.ReadSingleMeasurementValueOfAllSignals Method

DAQセッションを完全に初期化せずに、デバイスのすべてのSignalに対して単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。

C++

```
void ReadSingleMeasurementValueOfAllSignals();
```

C#

```
public override void ReadSingleMeasurementValueOfAllSignals();
```

Visual Basic

```
Public override Function ReadSingleMeasurementValueOfAllSignals() As void
```

備考

デバイスが単一の測定値のタイムスタンプをサポートしていない場合、PCの現在の時刻を使用してタイムスタンプを生成します。

PmxDevice.ReadTeds Method

指定したコネクタに接続されているTEDSからセンサを読み取ります。

C++

```
Sensor ReadTeds(
    Connector connector,
    out string tid,
    out List<Problem> problems
);
```

C#

```
public override Sensor ReadTeds(
    Connector connector,
    out string tid,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function ReadTeds(
    connector As Connector,
    tid As out string,
    problems As out List<Problem>
) As Sensor
```

PmxDevice.Reconnect Method

すでに部分的にまたは完全に入力されているオブジェクトモデル(Connectorなど)を使用して、基礎となる物理デバイスに再接続します。すでに使用可能なコネクタは物理デバイスに書き込まれ、不足しているコネクタは元のオブジェクトインスタンスを保持することでデバイスから埋め戻されます。

C++

```
bool Reconnect(
    out List<Problem> problems
);
```

C#

```
protected override bool Reconnect(
    out List<Problem> problems
);
```

Visual Basic

```
Protected override Function Reconnect(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	再接続プロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

PmxDevice.RefreshDeviceInfo Method

名前、シリアル番号などの関連するデバイスプロパティを更新します。すべてのコネクタを読み取らずにデバイス自体に関する情報を取得する場合は、この関数を使用します。

C++

```
bool RefreshDeviceInfo(
    List<Problem> problems
);
```

C#

```
protected override bool RefreshDeviceInfo(
    List<Problem> problems
);
```

Visual Basic

```
Protected override Function RefreshDeviceInfo(
    problems As List<Problem>
) As bool
```

引数

引数	説明
problems	更新プロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

PmxDevice.RemoveInputSignalReference Method

指定したチャネルから入力信号のリファレンスを削除します。

C++

```
bool RemoveInputSignalReference(
    IInputSignalReference channel,
    out List<Problem> problems
);
```

C#

```
public override bool RemoveInputSignalReference(
    IInputSignalReference channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function RemoveInputSignalReference(
    channel As IInputSignalReference
    problems As out List<Problem>
) As bool
```

PmxDevice.Reset Method

デバイスを工場出荷時のデフォルト設定にリセットします。注意！この関数の呼び出し後に、デバイスは切断されます。これが必要なのはデバイスのパラメータ設定が変更され、デバイスがデバイスの関連オブジェクトモデルと一致しなくなったとき、工場出荷時のデフォルトをロードします(この関数を実装するときは、必ずデバイスを切断し、Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知します。

C++

```
void Reset(
    out List<Problem> problems
);
```

C#

```
public override void Reset(
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function Reset(
    problems As out List<Problem>
) As void
```

引数

引数	説明
problems	デバイスのリセット中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

この関数の実行後にデバイスが切断され、一定時間アクセス不能になる可能性があることに注意してください。

PmxDevice.SetAnalogOut Method

アナログ出力信号を指定した値に設定します。

C++

```
void SetAnalogOut(
    AnalogOutSignal analogOutSignal,
    decimal value
);
```

C#

```
public override void SetAnalogOut(
    AnalogOutSignal analogOutSignal,
    decimal value
);
```

Visual Basic

```
Public override Function SetAnalogOut(
    analogOutSignal As AnalogOutSignal,
    value out decimal
) As void
```

引数

引数	説明
analogOutSignal	設定するアナログ出力信号。
value	設定する値。

例外

例外	説明
System.ArgumentNullException	アナログ出力信号が`null`です。
Hbm.Api.Common.Exceptions.UnknownSignalException	アナログ出力信号がこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

PmxDevice.SetCanBusParameters Method

指定したCanConnectorのすべてのCAN busパラメータ(CanConnector.BitRate、CanConnector.IsTerminated、CanConnector.SupplyVoltageなど)を割り当てますが、チャネルは割り当てません。

C++

```
bool SetCanBusParameters(
    CanConnector canConnector,
    out List<Problem> problems
);
```

C#

```
public override bool SetCanBusParameters(
    CanConnector canConnector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetCanBusParameters(
    canConnector As CanConnector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
canConnector	プロパティを割り当てる必要がある既存の(変更された)CanConnector。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

PmxDevice.SetCanChannelActivation Method

提供されたチャネルのリストをアクティブまたは非アクティブにします。CanInChannelとCanOutChannelのみをアクティブ化または非アクティブ化できます。

C++

```
bool SetCanChannelActivation(
    List<Channel> canChannels,
    bool activate,
    out List<Problem> problems
);
```

C#

```
public override bool SetCanChannelActivation(
    List<Channel> canChannels,
    bool activate,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetCanChannelActivation(
    canChannels As List<Channel>,
    activate As bool,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
canChannels	アクティブ化/非アクティブ化する必要があるCANチャネル。
activate	指定したチャネルをアクティブにする場合はtrue、それ以外の場合はfalse。
problems	アクティブ化/非アクティブ化プロセス中に発生した警告とエラーのリスト。

戻り値

TEDS設定が有効になっている場合はtrue、それ以外の場合はfalse。

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
Hbm.Api.Pmx Namespace -> PmxDevice Class ->
PmxDevice Methods

PmxDevice.SetChannelName Method

指定したチャネルにチャネル名を割り当てます。

C++

```
bool SetChannelName(
    Channel channel,
    string channelName,
    out List<Problem> problems
);
```

C#

```
public override bool SetChannelName(
    Channel channel,
    string channelName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetChannelName(
    channel As Channel,
    channelName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	変更するチャネルの名。
channelName	新しいチャネル名。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

実装では、チャネル名が正常に割り当て後にHbm.Api.Common.Messaging.MessageBroker.RaiseChannelNameChangedAsyncを呼び出す必要があります。

例外

例外	説明
System.ArgumentNullException	チャネルが`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

PmxDevice.SetDataRateDomain Method

dataRateDomainを設定します。

C++

```
bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

C#

```
public override bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetDataRateDomain(
    dataRateDomain As DataRateDomainType,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
dataRateDomain	モジュールに設定するデータレートドメインタイプ。
problems	データレートドメインの設定中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

PmxDevice.SetDeviceName Method

デバイスの名前を設定します。PMXデバイスの名前を変更することはできません。

C++

```
bool SetDeviceName(
    string newName,
    out List<Problem> problems
);
```

C#

```
public override bool SetDeviceName(
    string newName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetDeviceName(
    newName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
newName	新しい名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

PmxDevice.SetDigitalIO Method

指定した値に従ってデジタル信号を設定します。

C++

```
void SetDigitalIO(
    DigitalSignal digitalSignal,
    DigitalValueType value
);
```

C#

```
public override void SetDigitalIO(
    DigitalSignal digitalSignal,
    DigitalValueType value
);
```

Visual Basic

```
Public override Function SetDigitalIO(
    digitalSignal As DigitalSignal,
    value As out DigitalValueType
) As void
```

引数

引数	説明
digitalSignal	調整するデジタル信号。
value	設定する値。

例外

例外	説明
System.ArgumentNullException	デジタル信号が`null`です。
System.InvalidOperationException	デジタル信号のコネクタが出力操作をサポートしていない場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownSignalException	デジタル信号がこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

PmxDevice.SetInputSignalReference Method

入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。

C++

```
bool SetInputSignalReference(
    IInputSignalReference channel,
    Device inputSignalDevice,
    Signal inputSignal,
    out List<Problem> problems
);
```

C#

```
public override bool SetInputSignalReference(
    IInputSignalReference channel,
    Device inputSignalDevice,
    Signal inputSignal,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetInputSignalReference(
    channel As IInputSignalReference
    inputSignalDevice As Device
    inputSignal As Signal
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	入力信号リファレンスで設定される(出力)チャネル。
inputSignalDevice	入力信号のデバイス。
inputSignal	入力信号
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	channel、inputSignalDeviceまたはinputSignalが`null`です。
System.ArgumentException	inputSignalがDigitalSignalまたはAnalogOutSignalの場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownSignalException	inputSignalがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

SetShunt Method

これは、SetShuntメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	PmxDevice.SetShunt (Connector, ShuntMode, out List<Problem>)	指定したコネクタのシャント出力の構成を設定します。
	PmxDevice.SetShunt (List<Connector>, ShuntMode, out List<Problem>)	指定したコネクタのシャント出力の構成を設定します。

PmxDevice.SetShunt Method (Connector, ShuntMode, out List<Problem>)

指定したコネクタのシャント出力の構成を設定します。

C++

```
bool SetShunt(
    Connector connector,
    ShuntMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool SetShunt(
    Connector connector,
    ShuntMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetShunt(
    connector As Connector,
    mode As ShuntMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	シャント出力を設定するコネクタ。
mode	指定したコネクタのShuntMode。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

PmxDevice.SetShunt Method (List<Connector>, ShuntMode, out List<Problem>)

指定したコネクタのシャント出力の構成を設定します。

C++

```
bool SetShunt(
    List<Connector> connectors,
    ShuntMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool SetShunt(
    List<Connector> connectors,
    ShuntMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetShunt(
    connectors As List<Connector>,
    mode As ShuntMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connectors	シャント出力を設定するコネクタ。
mode	指定したコネクタのShuntMode。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合はtrue、それ以外の場合はfalse。

PmxDevice.SetSignalName Method

指定した信号に信号名を割り当てます。

C++

```
bool SetSignalName(
    Signal signal,
    string signalName,
    out List<Problem> problems
);
```

C#

```
public override bool SetSignalName(
    Signal signal,
    string signalName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetSignalName(
    signal As Signal,
    signalName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
signal	名前を変更する信号。
signalName	信号の新しい名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

実装では、信号名が正常に割り当てられた後にHbm.Api.Common.Messaging.MessageBroker.RaiseSignalNameChangedAsyncを呼び出す必要があります。

PmxDevice.SetTedsUsageMode Method

TEDS使用モードを設定し、コネクタ設定を再読み込みします。

C++

```
bool SetTedsUsageMode(
    Connector connector,
    TedsUsageMode tedsUsageMode,
    out List<Problem> problems
);
```

C#

```
public override bool SetTedsUsageMode(
    Connector connector,
    TedsUsageMode tedsUsageMode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetTedsUsageMode(
    connector As Connector,
    tedsUsageMode As TedsUsageMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	コネクタ。
tedsUsageMode	設定するTEDS使用モード。
problems	出力: 警告やエラーのリスト。何も発生しなかった場合は空です。

現在位置: Supported devices -> Hbm.Api.Pmx Namespace ->
Hbm.Api.Pmx Namespace -> PmxDevice Class ->
PmxDevice Methods

PmxDevice.SetUnit Method

指定したチャネルの単位を割り当てます。

C++

```
bool SetUnit(
    Channel channel,
    string unit,
    out List<Problem> problems
);
```

C#

```
public override bool SetUnit(
    Channel channel,
    string unit,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetUnit(
    channel As Channel,
    unit As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	単位を設定するチャネル。
unit	信号の新しい名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

PmxDevice.WriteTeds Method

指定したコネクタに接続されたTEDSに指定したセンサを書き込みます。

C++

```
bool WriteTeds(
    Connector connector,
    out List<Problem> problems,
    Sensor sensor,
    string tid = null,
    string channelName = null
);
```

C#

```
public override bool WriteTeds(
    Connector connector,
    out List<Problem> problems,
    Sensor sensor,
    string tid,
    string channelName
);
```

Visual Basic

```
Public override Function WriteTeds(
    connector As Connector,
    problems As out List<Problem>,
    sensor As Sensor,
    tid As string = null,
    channelName As string = null
) As bool
```

PmxDevice Properties

PmxDeviceクラスのプロパティの一覧です。

Internal Properties

	名前	説明
	LogContextDevPmxStatus	特定のデバイスに分類ステータスログメッセージを記録するLog Context。内部使用のみ。

Public Properties

	名前	説明
	AdditionalFeatures	PMXデバイスの追加機能。
	FamilyName	このデバイスが属するファミリの名前。

PmxDevice.AdditionalFeatures Property

PMXデバイスの追加機能。

C++

```
__property PmxAdditionalFeatures AdditionalFeatures;
```

C#

```
public PmxAdditionalFeatures AdditionalFeatures;
```

Visual Basic

```
Public Property AdditionalFeatures() As PmxAdditionalFeatures
```

備考

追加機能は、Deviceクラスの抽象関数によって事前に定義されていないすべてのパブリック関数をカバーします。例: PmxDeviceのLoadParameterSet()関数はPmxDeviceのこれらの追加機能に属しますが、AssignConnector()はDeviceクラスの抽象関数であるため、追加機能には属しません。

PmxDevice.FamilyName Property

このデバイスが属するファミリの名前。

C++

```
__property string FamilyName;
```

C#

```
public override string FamilyName;
```

Visual Basic

```
Public override Property FamilyName() As string
```

備考

これは、クラスHbm.Api.Pmx.PmxDeviceFamilyのプロパティFamilyNameとまったく同じ文字列でなければなりません。

PmxDevice.LogContextDevPmxStatus Property

特定のデバイスに分類ステータスログメッセージを記録するLog Context。内部使用のみ。

C++

```
__property LogContext LogContextDevPmxStatus;
```

C#

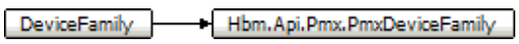
```
internal LogContext LogContextDevPmxStatus;
```

Visual Basic

```
Friend Property LogContextDevPmxStatus() As LogContext
```

PmxDeviceFamily Class

PMXデバイスの具体的なデバイスファミリの実装。



C++

```
class PmxDeviceFamily : public DeviceFamily;
```

C#

```
public class PmxDeviceFamily : DeviceFamily;
```

Visual Basic

```
Public Class PmxDeviceFamily
Inherits DeviceFamily
```

ファイル

PmxDeviceFamily.cs

PmxDeviceFamily Members

次の表は、PmxDeviceFamilyによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Find	指定したConnectionInfoに従ってデバイスを検索します。
	GetPossibleDataAcquisitionModes	ファミリの指定したすべてのデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSourceは、ファミリのすべてのデバイスで使用するタイムソースに設定します(指定したデバイスでDataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関連します)。
	PmxDeviceFamily	PmxDeviceFamilyクラスの新しいインスタンスを初期化します。
	PrepareDataAcquisition	指定したDataAcquisitionModeに従って最終的なデバイス設定を行う関数。この関数を使用して、トリガマスタとトリガスレーブなどの設定、指定したデータ収集モードに関連する特定のタイムスタンプ設定の選択などを行います。測定に関与する特定のデバイスファミリのすべてのデバイスもこの関数に渡されるため、特定のデバイスをさまざまな方法で処理できます(たとえば、トリガマスタはスレーブと異なる設定/準備を取得します)。
	Scan	ネットワーク上のPMXデバイスをスキャンします。

Public Properties

	名前	説明
R	FamilyName	デバイスファミリの名前を取得します(読み取り専用)。
R	IsScanSupported	デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。
	SkillReader	SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

PmxDeviceFamily.PmxDeviceFamily Constructor

PmxDeviceFamilyクラスの新しいインスタンスを初期化します。

C++

```
PmxDeviceFamily();
```

C#

```
public PmxDeviceFamily();
```

Visual Basic

```
Public Sub New()
```

PmxDeviceFamily Methods

PmxDeviceFamilyクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Find	指定したConnectionInfoに従ってデバイスを検索します。
	GetPossibleDataAcquisitionModes	ファミリの指定したすべてのデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSourceは、ファミリのすべてのデバイスで使用するタイムソースに設定します(指定したデバイスでDataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関連します)。
	PrepareDataAcquisition	指定したDataAcquisitionModeに従って最終的なデバイス設定を行う関数。この関数を使用して、トリガマスタとトリガスレーブなどの設定、指定したデータ収集モードに関連する特定のタイムスタンプ設定の選択などを行います。測定に関与する特定のデバイスファミリのすべてのデバイスもこの関数に渡されるため、特定のデバイスをさまざまな方法で処理できます(たとえば、トリガマスタはスレーブと異なる設定/準備を取得します)。
	Scan	ネットワーク上のPMXデバイスをスキャンします。

PmxDeviceFamily.Find Method

指定したConnectionInfoに従ってデバイスを検索します。

C++

```
List<Device> Find(
    List<ConnectionInfo> connectionInfos,
    out List<Problem> problems
);
```

C#

```
public override List<Device> Find(
    List<ConnectionInfo> connectionInfos,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function Find(
    connectionInfos As List<ConnectionInfo>,
    problems As out List<Problem>
) As List<Device>
```

引数

引数	説明
connectionInfos	接続情報。
problems	検索プロセス中に発生した警告とエラー。

戻り値

見つかったすべてのデバイスのリスト。

PmxDeviceFamily.GetPossibleDataAcquisitionModes Method

ファミリの指定したすべてのデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSourceは、ファミリのすべてのデバイスで使用するタイムソースに設定します(指定したデバイスでDataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関連します)。

C++

```
List<DataAcquisitionMode> GetPossibleDataAcquisitionModes(
    List<Device> devices,
    out TimeSource commonTimeSource
);
```

C#

```
public override List<DataAcquisitionMode> GetPossibleDataAcquisitionModes(
    List<Device> devices,
    out TimeSource commonTimeSource
);
```

Visual Basic

```
Public override Function GetPossibleDataAcquisitionModes(
    devices As List<Device>,
    commonTimeSource As out TimeSource
) As List<DataAcquisitionMode>
```

引数

引数	説明
devices	ファミリに属し、測定に参加するデバイス。
commonTimeSource	すべてのデバイスでサポートされているTimeSource(通常はNtpTimeSourceなど)。

戻り値

指定したデバイスの可能性に応じた可能なDataAcquisitionMode。

PmxDeviceFamily.PrepareDataAcquisition Method

指定したDataAcquisitionModeに従って最終的なデバイス設定を行う関数。この関数を使用して、トリガマスタとトリガスレーブなどの設定、指定したデータ収集モードに関連する特定のタイムスタンプ設定の選択などを行います。測定に関与する特定のデバイスファミリのすべてのデバイスもこの関数に渡されるため、特定のデバイスをさまざまな方法で処理できます(たとえば、トリガマスタはスレーブと異なる設定/準備を取得します)。

C++

```
void PrepareDataAcquisition(
    DataAcquisitionMode dataAcquisitionMode,
    List<Device> devicesToPrepare
);
```

C#

```
public override void PrepareDataAcquisition(
    DataAcquisitionMode dataAcquisitionMode,
    List<Device> devicesToPrepare
);
```

Visual Basic

```
Public override Function PrepareDataAcquisition(
    dataAcquisitionMode As DataAcquisitionMode,
    devicesToPrepare As List<Device>
) As void
```

引数

引数	説明
dataAcquisitionMode	DataAcquisitionModeを使用して、測定を開始する方法を記述します。
devicesToPrepare	ファミリに属し、測定に参加するデバイス。

戻り値

DataAcquisitionMode.Autoではなく、関数 GetPossibleDataAcquisitionModes(List{Device}, out TimeSource)によって返されるDataAcquisitionModeのいずれかを使用します。

PmxDeviceFamily.Scan Method

ネットワーク上のPMXデバイスをスキャンします。

C++

```
List<Device> Scan();
```

C#

```
public override List<Device>) Scan;
```

Visual Basic

```
Public override Function Scan() As List<Device>
```

戻り値

見つかったPMXデバイスのリスト。

例外

例外	説明
Hbm.Api.Scan.Entities.ScanFailedException	スキャン中にエラーが発生した場合にスローされます。

PmxDeviceFamily Properties

PmxDeviceFamilyクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	FamilyName	デバイスファミリの名前を取得します(読み取り専用)。
	IsScanSupported	デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。
	SkillReader	SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

PmxDeviceFamily.FamilyName Property

デバイスファミリの名前を取得します(読み取り専用)。

C++

```
__property string FamilyName;
```

C#

```
public override string FamilyName;
```

Visual Basic

```
Public override Property FamilyName() As string
```

PmxDeviceFamily.IsScanSupported Property

デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。

C++

```
__property bool IsScanSupported;
```

C#

```
public override bool IsScanSupported;
```

Visual Basic

```
Public override Property IsScanSupported() As bool
```

PmxDeviceFamily.SkillReader Property

SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

C++

```
__property ISkillReader SkillReader;
```

C#

```
public override ISkillReader SkillReader;
```

Visual Basic

```
Public override Property SkillReader() As ISkillReader
```

Hbm.Api.Pmx.Channels Namespace

これは名前空間Hbm.Api.Pmx.Channelsです。

クラス

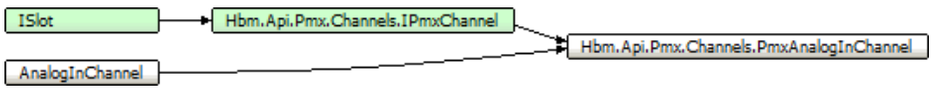
名前	説明
PmxAnalogInChannel	センサを接続するチャネルを表します。
PmxAnalogOutChannel	PMX固有のアナログ出力チャネルの実装。
PmxDigitalChannel	デジタル信号を保持するチャネルを表します。
PmxVirtualChannel	センサを接続しないチャネル(演算チャネルなど)を表します。

インタフェース

	名前	説明
	IPmxChannel	PmxChannelの追加プロパティのインタフェース。

PmxAnalogInChannel Class

センサを接続するチャネルを表します。



C++

```
class PmxAnalogInChannel : public AnalogInChannel, public IPmxChannel;
```

C#

```
public class PmxAnalogInChannel : AnalogInChannel, IPmxChannel;
```

Visual Basic

```
Public Class PmxAnalogInChannel
Inherits AnalogInChannel
Inherits IPmxChannel
```

ファイル

PmxAnalogInChannel.cs

PmxAnalogInChannel Members

次の表は、PmxAnalogInChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。
	TedsUsageMode	TEDS使用モードを取得または設定します。

PmxAnalogInChannel Methods

PmxAnalogInChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

PmxAnalogInChannel.GetUniqueID Method

チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。

PmxAnalogInChannel Properties

PmxAnalogInChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。
	TedsUsageMode	TEDS使用モードを取得または設定します。

PmxAnalogInChannel.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

PmxAnalogInChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

PmxAnalogInChannel.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

PmxAnalogInChannel.Position Property

チャネルが属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxAnalogInChannel.Slot Property

チャネルが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

PmxAnalogInChannel.TedsUsageMode Property

TEDS使用モードを取得または設定します。

C++

```
__property TedsUsageMode TedsUsageMode;
```

C#

```
public override TedsUsageMode TedsUsageMode;
```

Visual Basic

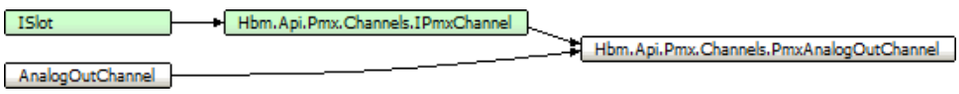
```
Public override Property TedsUsageMode() As TedsUsageMode
```

備考

TedsUsageMode.IgnoreおよびTedsUsageMode.UseManuallyのみをサポートしています。

PmxAnalogOutChannel Class

PMX固有のアナログ出力チャネルの実装。



C++

```
class PmxAnalogOutChannel : public AnalogOutChannel, public IPmxChannel;
```

C#

```
public class PmxAnalogOutChannel : AnalogOutChannel, IPmxChannel;
```

Visual Basic

```
Public Class PmxAnalogOutChannel
Inherits AnalogOutChannel
Inherits IPmxChannel
```

ファイル

PmxAnalogOutChannel.cs

PmxAnalogOutChannel Members

次の表は、PmxAnalogOutChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

PmxAnalogOutChannel Methods

PmxAnalogOutChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

PmxAnalogOutChannel.GetUniqueID Method

チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。

PmxAnalogOutChannel Properties

PmxAnalogOutChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

PmxAnalogOutChannel.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

PmxAnalogOutChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

PmxAnalogOutChannel.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

PmxAnalogOutChannel.Position Property

チャネルが属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxAnalogOutChannel.Slot Property

チャネルが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

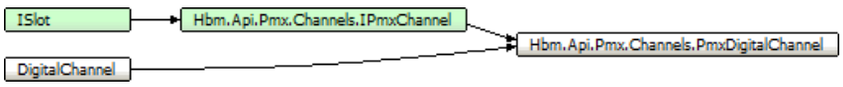
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

PmxDigitalChannel Class

デジタル信号を保持するチャネルを表します。



C++

```
class PmxDigitalChannel : public DigitalChannel, public IPmxChannel;
```

C#

```
public class PmxDigitalChannel : DigitalChannel, IPmxChannel;
```

Visual Basic

```
Public Class PmxDigitalChannel
Inherits DigitalChannel
Inherits IPmxChannel
```

ファイル

PmxDigitalChannel.cs

PmxDigitalChannel Members

次の表は、PmxDigitalChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Pin	このチャネルにマップされているデジタルピン。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

PmxDigitalChannel Methods

PmxDigitalChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

PmxDigitalChannel.GetUniqueID Method

デバイス内のチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。

PmxDigitalChannel Properties

PmxDigitalChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Pin	このチャネルにマップされているデジタルピン。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

PmxDigitalChannel.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

PmxDigitalChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

PmxDigitalChannel.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

PmxDigitalChannel.Pin Property

このチャネルにマップされているデジタルピン。

C++

```
[JsonProperty]
__property int Pin;
```

C#

```
[JsonProperty]
public int Pin;
```

Visual Basic

```
<JsonProperty>
Public Property Pin() As Integer
```

PmxDigitalChannel.Position Property

チャネルが属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxDigitalChannel.Slot Property

チャネルが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

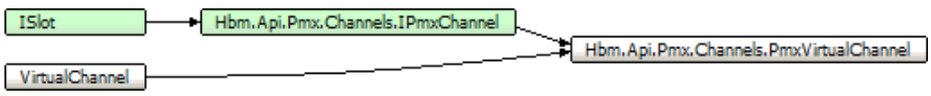
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

PmxVirtualChannel Class

センサを接続しないチャネル(演算チャネルなど)を表します。



C++

```
class PmxVirtualChannel : public VirtualChannel, public IPmxChannel;
```

C#

```
public class PmxVirtualChannel : VirtualChannel, IPmxChannel;
```

Visual Basic

```
Public Class PmxVirtualChannel
Inherits VirtualChannel
Inherits IPmxChannel
```

ファイル

PmxVirtualChannel.cs

PmxVirtualChannel Members

次の表は、PmxVirtualChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

PmxVirtualChannel Methods

PmxVirtualChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

PmxVirtualChannel.GetUniqueID Method

デバイス内のチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。

PmxVirtualChannel Properties

PmxVirtualChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

PmxVirtualChannel.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

PmxVirtualChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

PmxVirtualChannel.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

PmxVirtualChannel.Position Property

チャネルが属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxVirtualChannel.Slot Property

チャネルが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

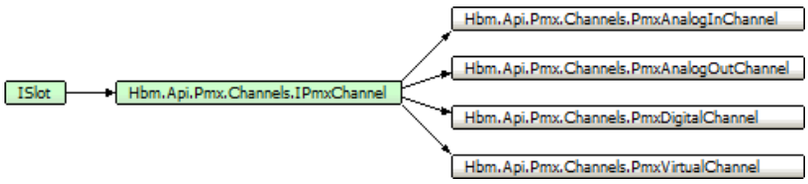
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

IPmxChannel Interface

PmxChannelの追加プロパティのインタフェース。



C++

```
__interface IPmxChannel : public ISlot;
```

C#

```
public interface IPmxChannel : ISlot;
```

Visual Basic

```
Public Interface IPmxChannel
Inherits ISlot
```

ファイル

IPmxChannel.cs

IPmxChannel Members

次の表は、IPmxChannelによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	AmplifierType	チャネルが属するアンプの種類。

IPmxChannel Properties

IPmxChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierType	チャネルが属するアンプの種類。

IPmxChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
[JsonProperty]
__property AmplifierType AmplifierType;
```

C#

```
[JsonProperty]
AmplifierType AmplifierType;
```

Visual Basic

```
<JsonProperty>
Property AmplifierType() As AmplifierType
```

Hbm.Api.Pmx.Connectors Namespace

これは名前空間Hbm.Api.Pmx.Connectorsです。

クラス

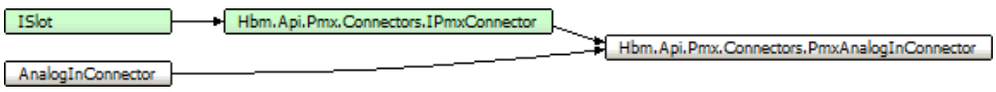
名前	説明
PmxAnalogInConnector	センサを接続するコネクタを表します。
PmxAnalogOutConnector	アナログ出力コネクタのPMX固有の実装。
PmxDigitalConnector	デジタルコネクタ(デジタル入出力ポートの1つのピンを保持可能)を表します。
PmxVirtualConnector	センサを接続しないコネクタ(演算チャネルを持つコネクタなど)を表します。

インタフェース

	名前	説明
	IPmxConnector	PmxConnectorの追加プロパティのインタフェース。

PmxAnalogInConnector Class

センサを接続するコネクタを表します。



C++

```
class PmxAnalogInConnector : public AnalogInConnector, public IPmxConnector;
```

C#

```
public class PmxAnalogInConnector : AnalogInConnector, IPmxConnector;
```

Visual Basic

```
Public Class PmxAnalogInConnector
Inherits AnalogInConnector
Inherits IPmxConnector
```

ファイル

PmxAnalogInConnector.cs

PmxAnalogInConnector Members

次の表は、PmxAnalogInConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	スロットに接続されているアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	アンプ上のコネクタの位置。
	Slot	チャネルが属するスロット。

PmxAnalogInConnector Methods

PmxAnalogInConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

PmxAnalogInConnector.GetUniqueID Method

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタ識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。一意のコネクタIDが設定されていない場合、メソッド実装はcref="System.InvalidOperationException">をスローする必要があります。

例外

例外	説明
System.InvalidOperationException	値が含まれていない場合はスローされます。

PmxAnalogInConnector Properties

PmxAnalogInConnectorクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	スロットに接続されているアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	アンプ上のコネクタの位置。
	Slot	チャネルが属するスロット。

PmxAnalogInConnector.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

PmxAnalogInConnector.AmplifierType Property

スロットに接続されているアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

PmxAnalogInConnector.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

PmxAnalogInConnector.Position Property

アンプ上のコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxAnalogInConnector.Slot Property

チャネルが属するスロット。

C++

```
__property int Slot;
```

C#

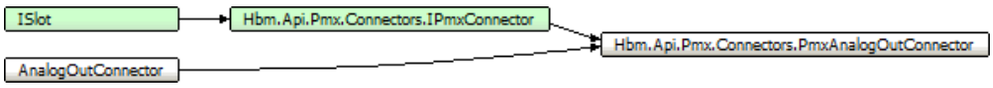
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

PmxAnalogOutConnector Class

アナログ出力コネクタのPMX固有の実装。



C++

```
class PmxAnalogOutConnector : public AnalogOutConnector, public IPmxConnector;
```

C#

```
public class PmxAnalogOutConnector : AnalogOutConnector, IPmxConnector;
```

Visual Basic

```
Public Class PmxAnalogOutConnector
Inherits AnalogOutConnector
Inherits IPmxConnector
```

ファイル

PmxAnalogOutConnector.cs

PmxAnalogOutConnector Members

次の表は、PmxAnalogOutConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	スロットに接続されているアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	アンプ上のコネクタの位置。
	Slot	チャネルが属するスロット。

PmxAnalogOutConnector Methods

PmxAnalogOutConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

PmxAnalogOutConnector.GetUniqueID Method

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。一意のコネクタIDが設定されていない場合、メソッド実装はcref="System.InvalidOperationException">をスローする必要があります。

例外

例外	説明
System.InvalidOperationException	値が含まれていない場合はスローされます。

PmxAnalogOutConnector Properties

PmxAnalogOutConnectorクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	スロットに接続されているアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	アンプ上のコネクタの位置。
	Slot	チャネルが属するスロット。

PmxAnalogOutConnector.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

PmxAnalogOutConnector.AmplifierType Property

スロットに接続されているアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

PmxAnalogOutConnector.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

PmxAnalogOutConnector.Position Property

アンプ上のコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxAnalogOutConnector.Slot Property

チャネルが属するスロット。

C++

```
__property int Slot;
```

C#

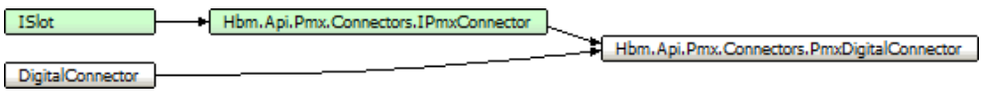
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

PmxDigitalConnector Class

デジタルコネクタ(デジタル入出力ポートの1つのピンを保持可能)を表します。



C++

```
class PmxDigitalConnector : public DigitalConnector, public IPmxConnector;
```

C#

```
public class PmxDigitalConnector : DigitalConnector, IPmxConnector;
```

Visual Basic

```
Public Class PmxDigitalConnector
Inherits DigitalConnector
Inherits IPmxConnector
```

ファイル

PmxDigitalConnector.cs

PmxDigitalConnector Members

次の表は、PmxDigitalConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	スロットに接続されているアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Pin	このコネクタにマップされているデジタルピン。
	Position	アンプ上のコネクタの位置。
	Slot	チャネルが属するスロット。

PmxDigitalConnector Methods

PmxDigitalConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

PmxDigitalConnector.GetUniqueID Method

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。一意のコネクタIDが設定されていない場合、メソッド実装はcref="System.InvalidOperationException">をスローする必要があります。

例外

例外	説明
System.InvalidOperationException	値が含まれていない場合はスローされます。

PmxDigitalConnector Properties

PmxDigitalConnectorクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	スロットに接続されているアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Pin	このコネクタにマップされているデジタルピン。
	Position	アンプ上のコネクタの位置。
	Slot	チャネルが属するスロット。

PmxDigitalConnector.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

PmxDigitalConnector.AmplifierType Property

スロットに接続されているアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

PmxDigitalConnector.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

PmxDigitalConnector.Pin Property

このコネクタにマップされているデジタルピン。

C++

```
[JsonProperty]
__property int Pin;
```

C#

```
[JsonProperty]
public int Pin;
```

Visual Basic

```
<JsonProperty>
Public Property Pin() As Integer
```

PmxDigitalConnector.Position Property

アンプ上のコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxDigitalConnector.Slot Property

チャネルが属するスロット。

C++

```
__property int Slot;
```

C#

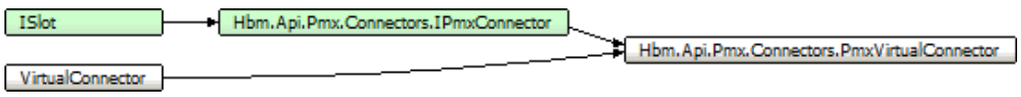
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

PmxVirtualConnector Class

センサを接続しないコネクタ(演算チャネルを持つコネクタなど)を表します。



C++

```
class PmxVirtualConnector : public VirtualConnector, public IPmxConnector;
```

C#

```
public class PmxVirtualConnector : VirtualConnector, IPmxConnector;
```

Visual Basic

```
Public Class PmxVirtualConnector
Inherits VirtualConnector
Inherits IPmxConnector
```

ファイル

PmxVirtualConnector.cs

PmxVirtualConnector Members

次の表は、PmxVirtualConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	スロットに接続されているアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	アンプ上のコネクタの位置。
	Slot	チャネルが属するスロット。

PmxVirtualConnector Methods

PmxVirtualConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

PmxVirtualConnector.GetUniqueID Method

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。

PmxVirtualConnector Properties

PmxVirtualConnectorクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	スロットに接続されているアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	Position	アンプ上のコネクタの位置。
	Slot	チャネルが属するスロット。

PmxVirtualConnector.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

PmxVirtualConnector.AmplifierType Property

スロットに接続されているアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

PmxVirtualConnector.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

PmxVirtualConnector.Position Property

アンプ上のコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxVirtualConnector.Slot Property

コネクタが属するデバイス内でコネクタを識別する一意の文字列を返します。

C++

```
__property int Slot;
```

C#

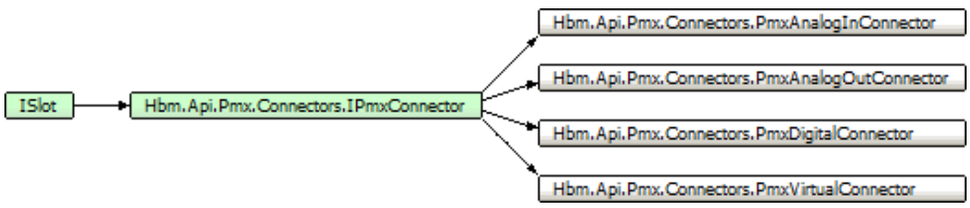
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

IPmxConnector Interface

PmxConnectorの追加プロパティのインタフェース。



C++

```
__interface IPmxConnector : public ISlot;
```

C#

```
public interface IPmxConnector : ISlot;
```

Visual Basic

```
Public Interface IPmxConnector
Inherits ISlot
```

ファイル

IPmxConnector.cs

IPmxConnector Members

次の表は、IPmxConnectorによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	AmplifierType	コネクタが属するアンプの種類。

IPmxConnector Properties

PmxVirtualConnectorクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierType	コネクタが属するアンプの種類。

IPmxConnector.AmplifierType Property

コネクタが属するアンプの種類。

C++

```
[JsonProperty]
__property AmplifierType AmplifierType;
```

C#

```
[JsonProperty]
AmplifierType AmplifierType;
```

Visual Basic

```
<JsonProperty>
Property AmplifierType() As AmplifierType
```

Hbm.Api.Pmx.Enums Namespace

これは名前空間Hbm.Api.Pmx.Enumsです。

列挙型

	名前	説明
	AmplifierType	PMXアンプの種類。
	EngineeringUnit	工学単位の文字列。
	ExcitationAmplitude	PMXがサポートする励起振幅。
	FilterCharacteristic	PMXがサポートするフィルタ特性。
	FilterFrequency	周波数(Hz)。
	InputType	PX460専用の入力タイプ。
	LimitSwitchOperatingDirection	PMXリミットスイッチの動作方向。
	PmxDeviceStatus	Deviceのステータス。
	PmxSensorType	センサの種類。
	SampleRate	周波数(Hz)。
	SsiBaudRate	PMXがサポートするSSIボーレート。
	TransducerSensitivity	PMXがサポートするトランスデューサの感度。

Hbm.Api.Pmx.Enums.AmplifierType Enumeration

PMXアンプの種類。

C++

```
enum AmplifierType {
  Unknown = -1,
  PX401 = 5125,
  PX455 = 5126,
  PX878 = 5127,
  PX460 = 5128,
  VirtualCalc = 5130,
  DigitalIO = 5131
};
```

C#

```
public enum AmplifierType {
  Unknown = -1,
  PX401 = 5125,
  PX455 = 5126,
  PX878 = 5127,
  PX460 = 5128,
  VirtualCalc = 5130,
  DigitalIO = 5131
}
```

Visual Basic

```
Public Enum AmplifierType
  Unknown = -1
  PX401 = 5125
  PX455 = 5126
  PX878 = 5127
  PX460 = 5128
  VirtualCalc = 5130
  DigitalIO = 5131
End Enum
```

ファイル

AmplifierType.cs

メンバー

メンバー	説明
Unknown	不明なアンプ。
PX401	PX401アンプ(電圧、電流)。
PX455	PX455アンプ(ブリッジ、LVDT、ポテンショメータ)。
PX878	PX878アンプ(アナログ出力)。
PX460	PX460アンプ(周波数、インパルスカウンタ、PWM、SSI)。
VirtualCalc	仮想演算チャネル用のアンプ。
DigitalIO	デジタルIO用のアンプ。物理IOの数(決定できません)は、PMXに接続されているPX878モジュールの数によって異なります。常に16点のDIと16点のDOがあります。最初のPX878は最初の8点のDIとDO(0..7)を提供し、2番目のPX878は2番目の8点のDIとDO(8..15)を提供します。

備考

列挙項目の値は、デバイスのクエリ言語によって返されるアンプの種類番号と同じです。

Hbm.Api.Pmx.Enums.EngineeringUnit Enumeration

工学単位の文字列。

C++

```
[AdditionalValueType(typeof(string))]
enum EngineeringUnit {
  UNKNOWN = -1,
  none = 0,
  rad = 100,
  radian = 101,
  deg = 102,
  percento = 103,
  m = 300,
  um = 301,
  mm = 302,
  cm = 303,
  dm = 304,
  km = 305,
  inch = 306,
  feet = 307,
  yard = 308,
  mile = 309,
  kg = 400,
  g = 401,
  t = 402,
  kt = 403,
  tons = 404,
  lbs = 405,
  s = 500,
  ms = 501,
  us = 502,
  min = 503,
  h = 504,
  days = 505,
  A = 600,
  Arms = 601,
  mA = 602,
  uA = 603,
  mArms = 604,
  uArms = 605,
  K = 700,
  oC = 701,
  oF = 702,
  oRank = 703,
  oR = 704,
  V_V = 1000,
  mV_V = 1001,
  uV_V = 1002,
  V = 1100,
  mV = 1101,
  uV = 1102,
  Vrms = 1103,
  mVrms = 1104,
  uVrms = 1105,
  Ohm = 1200,
  kOhm = 1201,
  MOhm_ = 1202,
  mOhm = 1203,
  H_ = 1300,
  mH = 1301,
  uH = 1302,
```

```
  nH = 1303,
  F = 1400,
  mF = 1401,
  uF = 1402,
  nF = 1403,
  pF = 1404,
  C = 1500,
  nC = 1501,
  pC = 1502,
  Hz = 1600,
  kHz = 1601,
  MHz = 1602,
  One_s = 1603,
  mHz_ = 1604,
  radian_s = 1700,
  U_min = 1701,
  rpm = 1702,
  One_min = 1703,
  W = 1800,
  mW = 1801,
  kW = 1802,
  MW_ = 1803,
  GW = 1804,
  N = 1900,
  kN = 1901,
  MN = 1902,
  kp = 1903,
  kgf = 1904,
  lb = 1905,
  GN = 1906,
  Pa = 2000,
  bar = 2001,
  mbar = 2002,
  kbar = 2003,
  pas = 2004,
  hPa = 2005,
  kPa = 2006,
  psi = 2007,
  N_mm2 = 2008,
  N_m2 = 2009,
  N_cm2 = 2010,
  MPa = 2011,
  GPa = 2012,
  J = 2100,
  Nm = 2101,
  kNm = 2102,
  MNm = 2103,
  ftlb = 2104,
  inlb = 2105,
  GNm = 2106,
  Nm_radian = 2107,
  ozin = 2108,
  m_m = 2300,
  um_m = 2301,
  strain = 2302,
  mm_m = 2303,
  m_s = 2400,
  km_h = 2401,
  mph = 2402,
  fps = 2403,
  m_h = 2404,
  m_s2 = 2500,
  ga = 2501,
```


現在位置: Supported devices -> Hbm.Api.Pmx Namespace -> Hbm.Api.Pmx.Enums Namespace

```
  mm_s2 = 2502,
  kg_m3 = 2700,
  g_l = 2701,
  m3_s = 2800,
  l_min = 2801,
  m3_h = 2802,
  gpm = 2803,
  cfm = 2804,
  l_h = 2805,
  l_s = 2806,
  percent = 2900,
  promill = 2901,
  ppm = 2902,
  percent_oC = 3000,
  promill_oC = 3001,
  ppm_oC = 3002,
  Imp = 3100,
  kImp = 3101,
  percent_decade = 5001,
  dB = 5002,
  l_l = 5003,
  m3_m3 = 5004,
  m3 = 5005,
  mm2 = 5006,
  kg_s = 5007,
  mole_l = 5008,
  mole_m3 = 5009,
  N_m = 5010,
  RH = 5011,
  V__m_s2 = 5012,
  V_C = 5013,
  V_N = 5014,
  V_PA = 5015,
  W_oC = 5016,
  UserDefined = 100000
};
```

C#

```
[AdditionalValueType(typeof(string))]
internal enum EngineeringUnit {
  UNKNOWN = -1,
  none = 0,
  rad = 100,
  radian = 101,
  deg = 102,
  percento = 103,
  m = 300,
  um = 301,
  mm = 302,
  cm = 303,
  dm = 304,
  km = 305,
  inch = 306,
  feet = 307,
  yard = 308,
  mile = 309,
  kg = 400,
  g = 401,
  t = 402,
  kt = 403,
  tons = 404,
```

```
lbs = 405,
s = 500,
ms = 501,
us = 502,
min = 503,
h = 504,
days = 505,
A = 600,
Arms = 601,
mA = 602,
uA = 603,
mArms = 604,
uArms = 605,
K = 700,
oC = 701,
oF = 702,
oRank = 703,
oR = 704,
V_V = 1000,
mV_V = 1001,
uV_V = 1002,
V = 1100,
mV = 1101,
uV = 1102,
Vrms = 1103,
mVrms = 1104,
uVrms = 1105,
Ohm = 1200,
kOhm = 1201,
MOhm_ = 1202,
mOhm = 1203,
H_ = 1300,
mH = 1301,
uH = 1302,
nH = 1303,
F = 1400,
mF = 1401,
uF = 1402,
nF = 1403,
pF = 1404,
C = 1500,
nC = 1501,
pC = 1502,
Hz = 1600,
kHz = 1601,
MHz = 1602,
One_s = 1603,
mHz_ = 1604,
radian_s = 1700,
U_min = 1701,
rpm = 1702,
One_min = 1703,
W = 1800,
mW = 1801,
kW = 1802,
MW_ = 1803,
GW = 1804,
N = 1900,
kN = 1901,
MN = 1902,
kp = 1903,
kgf = 1904,
lb = 1905,
```


現在位置: Supported devices -> Hbm.Api.Pmx Namespace -> Hbm.Api.Pmx.Enums Namespace

```
    GN = 1906,
    Pa = 2000,
    bar = 2001,
    mbar = 2002,
    kbar = 2003,
    pas = 2004,
    hPa = 2005,
    kPa = 2006,
    psi = 2007,
    N_mm2 = 2008,
    N_m2 = 2009,
    N_cm2 = 2010,
    MPa = 2011,
    GPa = 2012,
    J = 2100,
    Nm = 2101,
    kNm = 2102,
    MNm = 2103,
    ftlb = 2104,
    inlb = 2105,
    GNm = 2106,
    Nm_radian = 2107,
    ozin = 2108,
    m_m = 2300,
    um_m = 2301,
    strain = 2302,
    mm_m = 2303,
    m_s = 2400,
    km_h = 2401,
    mph = 2402,
    fps = 2403,
    m_h = 2404,
    m_s2 = 2500,
    ga = 2501,
    mm_s2 = 2502,
    kg_m3 = 2700,
    g_l = 2701,
    m3_s = 2800,
    l_min = 2801,
    m3_h = 2802,
    gpm = 2803,
    cfm = 2804,
    l_h = 2805,
    l_s = 2806,
    percent = 2900,
    promill = 2901,
    ppm = 2902,
    percent_oC = 3000,
    promill_oC = 3001,
    ppm_oC = 3002,
    Imp = 3100,
    kImp = 3101,
    percent_decade = 5001,
    dB = 5002,
    l_l = 5003,
    m3_m3 = 5004,
    m3 = 5005,
    mm2 = 5006,
    kg_s = 5007,
    mole_l = 5008,
    mole_m3 = 5009,
    N_m = 5010,
    RH = 5011,
```

```
  V__m_s2 = 5012,
  V_C = 5013,
  V_N = 5014,
  V_PA = 5015,
  W_oC = 5016,
  UserDefined = 100000
}
```

Visual Basic

```
<AdditionalValueType(typeof(string))>
internal Enum EngineeringUnit
  UNKNOWN = -1
  none = 0
  rad = 100
  radian = 101
  deg = 102
  percento = 103
  m = 300
  um = 301
  mm = 302
  cm = 303
  dm = 304
  km = 305
  inch = 306
  feet = 307
  yard = 308
  mile = 309
  kg = 400
  g = 401
  t = 402
  kt = 403
  tons = 404
  lbs = 405
  s = 500
  ms = 501
  us = 502
  min = 503
  h = 504
  days = 505
  A = 600
  Arms = 601
  mA = 602
  uA = 603
  mArms = 604
  uArms = 605
  K = 700
  oC = 701
  oF = 702
  oRank = 703
  oR = 704
  V_V = 1000
  mV_V = 1001
  uV_V = 1002
  V = 1100
  mV = 1101
  uV = 1102
  Vrms = 1103
  mVrms = 1104
  uVrms = 1105
  Ohm = 1200
  kOhm = 1201
```

```
    MOhm_ = 1202
    mOhm = 1203
    H_ = 1300
    mH = 1301
    uH = 1302
    nH = 1303
    F = 1400
    mF = 1401
    uF = 1402
    nF = 1403
    pF = 1404
    C = 1500
    nC = 1501
    pC = 1502
    Hz = 1600
    kHz = 1601
    MHz = 1602
    One_s = 1603
    mHz_ = 1604
    radian_s = 1700
    U_min = 1701
    rpm = 1702
    One_min = 1703
    W = 1800
    mW = 1801
    kW = 1802
    MW_ = 1803
    GW = 1804
    N = 1900
    kN = 1901
    MN = 1902
    kp = 1903
    kgf = 1904
    lb = 1905
    GN = 1906
    Pa = 2000
    bar = 2001
    mbar = 2002
    kbar = 2003
    pas = 2004
    hPa = 2005
    kPa = 2006
    psi = 2007
    N_mm2 = 2008
    N_m2 = 2009
    N_cm2 = 2010
    MPa = 2011
    GPa = 2012
    J = 2100
    Nm = 2101
    kNm = 2102
    MNm = 2103
    ftlb = 2104
    inlb = 2105
    GNm = 2106
    Nm_radian = 2107
    ozin = 2108
    m_m = 2300
    um_m = 2301
    strain = 2302
    mm_m = 2303
    m_s = 2400
    km_h = 2401
```

```
  mph = 2402
  fps = 2403
  m_h = 2404
  m_s2 = 2500
  ga = 2501
  mm_s2 = 2502
  kg_m3 = 2700
  g_l = 2701
  m3_s = 2800
  l_min = 2801
  m3_h = 2802
  gpm = 2803
  cfm = 2804
  l_h = 2805
  l_s = 2806
  percent = 2900
  promill = 2901
  ppm = 2902
  percent_oC = 3000
  promill_oC = 3001
  ppm_oC = 3002
  Imp = 3100
  kImp = 3101
  percent_decade = 5001
  dB = 5002
  l_l = 5003
  m3_m3 = 5004
  m3 = 5005
  mm2 = 5006
  kg_s = 5007
  mole_l = 5008
  mole_m3 = 5009
  N_m = 5010
  RH = 5011
  V__m_s2 = 5012
  V_C = 5013
  V_N = 5014
  V_PA = 5015
  W_oC = 5016
  UserDefined = 100000
End Enum
```

ファイル

EngineeringUnit.cs

メンバー

メンバー	説明
MOhm_	ActiveXでは大文字と小文字が区別されないため、アンダースコアを使用します(MOhm = mOhm)。
mHz_	ActiveXでは大文字と小文字が区別されないため、アンダースコアを使用します(MHz = mHz)。

Hbm.Api.Pmx.Enums.ExcitationAmplitude Enumeration

PMXがサポートする励起振幅。

C++

```
enum ExcitationAmplitude {
  NoSupply = 10,
  _2Point5V = 13
};
```

C#

```
internal enum ExcitationAmplitude {
  NoSupply = 10,
  _2Point5V = 13
}
```

Visual Basic

```
Friend Enum ExcitationAmplitude
  NoSupply = 10
  _2Point5V = 13
End Enum
```

ファイル

ExcitationAmplitude.cs

備考

列挙項目の値は、デバイスのクエリ言語によって返される励起振幅数値と同じです。

Hbm.Api.Pmx.Enums.FilterCharacteristic Enumeration

PMXがサポートするフィルタ特性。

C++

```
enum FilterCharacteristic {
  NoFilter = 140,
  Butterworth = 141,
  Bessel = 142
};
```

C#

```
internal enum FilterCharacteristic {
  NoFilter = 140,
  Butterworth = 141,
  Bessel = 142
}
```

Visual Basic

```
Friend Enum FilterCharacteristic
  NoFilter = 140
  Butterworth = 141
  Bessel = 142
End Enum
```

ファイル

FilterCharacteristic.cs

備考

列挙項目の値は、デバイスのクエリ言語によって返されるフィルタ特性番号と同じです。

Hbm.Api.Pmx.Enums.FilterFrequency Enumeration

周波数(Hz)。

C++

```
[AdditionalValueType(typeof(decimal))]
enum FilterFrequency {
  Hz01 = 914,
  Hz02 = 917,
  Hz05 = 921,
  Hz1 = 927,
  Hz2 = 931,
  Hz5 = 931,
  Hz10 = 941,
  Hz20 = 945,
  Hz50 = 949,
  Hz100 = 955,
  Hz200 = 958,
  Hz500 = 962,
  Hz1000 = 969,
  Hz2000 = 973,
  Hz3000 = 976,
  Hz4000 = 977,
  Hz5000 = 978,
  Hz6000 = 979,
  Hz100000 = 1150
};
```

C#

```
[AdditionalValueType(typeof(decimal))]
internal enum FilterFrequency {
  Hz01 = 914,
  Hz02 = 917,
  Hz05 = 921,
  Hz1 = 927,
  Hz2 = 931,
  Hz5 = 931,
  Hz10 = 941,
  Hz20 = 945,
  Hz50 = 949,
  Hz100 = 955,
  Hz200 = 958,
  Hz500 = 962,
  Hz1000 = 969,
  Hz2000 = 973,
  Hz3000 = 976,
  Hz4000 = 977,
  Hz5000 = 978,
  Hz6000 = 979,
  Hz100000 = 1150
}
```

Visual Basic

```
<AdditionalValueType(typeof(decimal))>
Friend Enum FilterFrequency
  Hz01 = 914
  Hz02 = 917
  Hz05 = 921
  Hz1 = 927
  Hz2 = 931
  Hz5 = 931
  Hz10 = 941
  Hz20 = 945
  Hz50 = 949
  Hz100 = 955
  Hz200 = 958
  Hz500 = 962
  Hz1000 = 969
  Hz2000 = 973
  Hz3000 = 976
  Hz4000 = 977
  Hz5000 = 978
  Hz6000 = 979
  Hz100000 = 1150
End Enum
```

ファイル

FilterFrequency.cs

メンバー

メンバー	説明
Hz100000	つまり、アナログアンチエイリアスフィルタのみがアクティブになります...???changed??? _NoFilter = -1

Hbm.Api.Pmx.Enums.InputType Enumeration

PX460専用の入力タイプ。

C++

```
enum InputType {
  Direct = 23,
  Inductive = 24
};
```

C#

```
internal enum InputType {
  Direct = 23,
  Inductive = 24
}
```

Visual Basic

```
Friend Enum InputType
  Direct = 23
  Inductive = 24
End Enum
```

ファイル

InputType.cs

Hbm.Api.Pmx.Enums.LimitSwitchOperatingDirection Enumeration

PMXリミットスイッチの動作方向。

C++

```
enum LimitSwitchOperatingDirection {
  Off = -1,
  AboveLimit = 130,
  BelowLimit = 131,
  InBand = 132,
  OutOfBand = 133
};
```

C#

```
public enum LimitSwitchOperatingDirection {
  Off = -1,
  AboveLimit = 130,
  BelowLimit = 131,
  InBand = 132,
  OutOfBand = 133
}
```

Visual Basic

```
Public Enum LimitSwitchOperatingDirection
  Off = -1
  AboveLimit = 130
  BelowLimit = 131
  InBand = 132
  OutOfBand = 133
End Enum
```

ファイル

LimitSwitchOperatingDirection.cs

メンバー

メンバー	説明
Off	リミットスイッチは評価しません。
AboveLimit	リミットスイッチは指定値を超えるとトリガされます。
BelowLimit	リミットスイッチは指定値を下回るとトリガされます。
InBand	リミット スイッチは指定した帯域幅内でトリガされます。
OutOfBand	リミット スイッチは、指定した帯域幅を超えるとトリガされます。

備考

列挙項目の値は、デバイスのクエリ言語によって返される値と同じです。

Hbm.Api.Pmx.Enums.PmxDeviceStatus Enumeration

Deviceのステータス。

C++

```
[Flags()]
enum PmxDeviceStatus {
  NoError = 0,
  FactorySettingsError = 1,
  SyncMaster = 2,
  SyncMessageError = 4,
  SyncUnlockedSlaveError = 8,
  Alive = 16,
  PowerOverLoad = 32,
  CatBufferOverrun = 64,
  SystemNotReady = 128,
  DSPOverRun = 256
};
```

C#

```
[Flags()]
public enum PmxDeviceStatus {
  NoError = 0,
  FactorySettingsError = 1,
  SyncMaster = 2,
  SyncMessageError = 4,
  SyncUnlockedSlaveError = 8,
  Alive = 16,
  PowerOverLoad = 32,
  CatBufferOverrun = 64,
  SystemNotReady = 128,
  DSPOverRun = 256
}
```

Visual Basic

```
<Flags()>
Public Enum PmxDeviceStatus
  NoError = 0
  FactorySettingsError = 1
  SyncMaster = 2
  SyncMessageError = 4
  SyncUnlockedSlaveError = 8
  Alive = 16
  PowerOverLoad = 32
  CatBufferOverrun = 64
  SystemNotReady = 128
  DSPOverRun = 256
End Enum
```

ファイル

DeviceStatus.cs

メンバー

メンバー	説明
NoError	エラーなし。
FactorySettingsError	PMXハウジングの工場出荷時の校正が破損しています(測定カードではありません!)。
SyncMaster	1 = デバイスは同期マスタ。0 = デバイスは同期スレーブ。
SyncMessageError	同期メッセージが見つからないか無効です。
SyncUnlockedSlaveError	ロックできません。スレーブ側がマスタと同期できません。
Alive	約1Hzで切り替えます。
PowerOverLoad	過負荷が発生しました。
CatBufferOverrun	"Catman"インタフェースのバッファオーバーラン。
SystemNotReady	進行中のパラメータセットの変更に失敗など。
DSPOverRun	演算チャネルが多数ある場合など。

Hbm.Api.Pmx.Enums.PmxSensorType Enumeration

センサの種類。

C++

```
enum PmxSensorType {
  FullBridge = 350,
  HalfBridge = 351,
  InductiveFullBridge = 356,
  InductiveHalfBridge = 357,
  LVDT = 380,
  Potentiometer = 385,
  VoltageDC10V = 426,
  CurrentDC20mA = 427,
  CurrentDC4To20mA = 435,
  Frequency = 520,
  ImpulseCounter = 525,
  PWM = 527,
  SSI = 580,
  VirtualSensor = 575
};
```

C#

```
internal enum PmxSensorType {
  FullBridge = 350,
  HalfBridge = 351,
  InductiveFullBridge = 356,
  InductiveHalfBridge = 357,
  LVDT = 380,
  Potentiometer = 385,
  VoltageDC10V = 426,
  CurrentDC20mA = 427,
  CurrentDC4To20mA = 435,
  Frequency = 520,
  ImpulseCounter = 525,
  PWM = 527,
  SSI = 580,
  VirtualSensor = 575
}
```

Visual Basic

```
Friend Enum PmxSensorType
  FullBridge = 350
  HalfBridge = 351
  InductiveFullBridge = 356
  InductiveHalfBridge = 357
  LVDT = 380
  Potentiometer = 385
  VoltageDC10V = 426
  CurrentDC20mA = 427
  CurrentDC4To20mA = 435
  Frequency = 520
  ImpulseCounter = 525
  PWM = 527
  SSI = 580
  VirtualSensor = 575
End Enum
```

ファイル
PmxSensorType.cs

メンバー

メンバー	説明
FullBridge	サポートしています。
HalfBridge	サポートしています。
InductiveFullBridge	サポートしています。
LVDT	サポートしています。
Potentiometer	サポートしています。
VoltageDC10V	サポートしています。
CurrentDC20mA	サポートしています。
CurrentDC4To20mA	サポートしています。
Frequency	サポートしています。
ImpulseCounter	サポートしています。
PWM	サポートしています。
SSI	サポートしています。
VirtualSensor	サポートしています。

備考
列挙項目の値は、デバイスのクエリ言語によって返されるSensorType番号と同じです。

Hbm.Api.Pmx.Enums.SampleRate Enumeration

周波数(Hz)。

C++

```
[AdditionalValueType(typeof(decimal))]
enum SampleRate {
  _1Hz = 6300,
  _2Hz = 6301,
  _5Hz = 6302,
  _10Hz = 6303,
  _25Hz = 6304,
  _50Hz = 6305,
  _75Hz = 6307,
  _100Hz = 6308,
  _150Hz = 6309,
  _200Hz = 6310,
  _300Hz = 6311,
  _600Hz = 6313,
  _1200Hz = 6315,
  _2400Hz = 6317,
  _4800Hz = 6319,
  _9600Hz = 6320,
  _19200Hz = 6345,
  _38400Hz = 63464
};
```

C#

```
[AdditionalValueType(typeof(decimal))]
internal enum SampleRate {
  _1Hz = 6300,
  _2Hz = 6301,
  _5Hz = 6302,
  _10Hz = 6303,
  _25Hz = 6304,
  _50Hz = 6305,
  _75Hz = 6307,
  _100Hz = 6308,
  _150Hz = 6309,
  _200Hz = 6310,
  _300Hz = 6311,
  _600Hz = 6313,
  _1200Hz = 6315,
  _2400Hz = 6317,
  _4800Hz = 6319,
  _9600Hz = 6320,
  _19200Hz = 6345,
  _38400Hz = 63464
}
```

Visual Basic

```
<AdditionalValueType(typeof(decimal))>
Friend Enum SampleRate
  _1Hz = 6300
  _2Hz = 6301
  _5Hz = 6302
  _10Hz = 6303
  _25Hz = 6304
  _50Hz = 6305
  _75Hz = 6307
  _100Hz = 6308
  _150Hz = 6309
  _200Hz = 6310
  _300Hz = 6311
  _600Hz = 6313
  _1200Hz = 6315
  _2400Hz = 6317
  _4800Hz = 6319
  _9600Hz = 6320
  _19200Hz = 6345
  _38400Hz = 63464
End Enum
```

ファイル

SampleRate.cs

Hbm.Api.Pmx.Enums.SsiBaudRate Enumeration

PMXがサポートするSSIボーレート。

C++

```
[AdditionalValueType(typeof(int))]
enum SsiBaudRate {
  _None = 0,
  _10KBit = 1,
  _100KBit = 2,
  _200KBit = 3,
  _500KBit = 4,
  _1000Bit = 5
};
```

C#

```
{[AdditionalValueType(typeof(int))]
internal enum SsiBaudRate {
  _None = 0,
  _10KBit = 1,
  _100KBit = 2,
  _200KBit = 3,
  _500KBit = 4,
  _1000Bit = 5
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum SsiBaudRate
  _None = 0,
  _10KBit = 1,
  _100KBit = 2,
  _200KBit = 3,
  _500KBit = 4,
  _1000Bit = 5
End Enum
```

ファイル

SSIBaudRate.cs

メンバー

メンバー	説明
_None	これが何を意味するのか確認してください!!! None?

備考

列挙項目の値は、デバイスのクエリ言語によって返される SSIボーレート番号と同じです。

Hbm.Api.Pmx.Enums.TransducerSensitivity Enumeration

PMXがサポートするトランスデューサの感度。

C++

```
enum TransducerSensitivity {
  _4mVV = 778,
  _100mVV  = 774,
  _1000mVV  = 776
};
```

C#

```
internal enum TransducerSensitivity {
  _4mVV = 778,
  _100mVV  = 774,
  _1000mVV  = 776
}
```

Visual Basic

```
Friend Enum TransducerSensitivity
  _4mVV = 778,
  _100mVV  = 774,
  _1000mVV  = 776
End Enum
```

ファイル

TransducerSensitivity.cs

備考

列挙項目の値は、デバイスのクエリ言語によって返される励起振幅数値と同じです。

Hbm.Api.Pmx.Signals Namespace

これは名前空間Hbm.Api.Pmx.Signalsです。

クラス

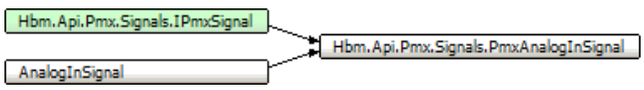
名前	説明
PmxAnalogInSignal	物理センサから送られる測定信号を表します。
PmxAnalogOutSignal	PMX固有のアナログ出力信号の実装。
PmxDigitalSignal	マスクされたビットに応じて、デジタル信号の一部(実際にはPMXの"ネイティブ"デジタル入力またはデジタル出力信号の1ビット)を表します。
PmxVirtualSignal	センサを接続しない信号(演算信号など)を表します。

インタフェース

	名前	説明
	IPmxSignal	PmxSignalの追加プロパティのインタフェース。

PmxAnalogInSignal Class

物理センサから送られる測定信号を表します。



C++

```
class PmxAnalogInSignal : public AnalogInSignal, public IPmxSignal;
```

C#

```
public class PmxAnalogInSignal : AnalogInSignal, IPmxSignal;
```

Visual Basic

```
Public Class PmxAnalogInSignal
Inherits AnalogInSignal
Inherits IPmxSignal
```

ファイル

PmxAnalogInSignal.cs

PmxAnalogInSignal Members

次の表は、PmxAnalogInSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

PmxAnalogInSignal Methods

PmxAnalogInSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

PmxAnalogInSignal.GetUniqueID Method

信号が属するデバイス内で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別子。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェア モジュール上の信号を識別するためにストリーミングで使用する情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。

PmxAnalogInSignal Properties

PmxAnalogInSignalクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

PmxAnalogInSignal.Position Property

信号が属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxAnalogInSignal.Slot Property

信号が属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

PmxAnalogOutSignal Class

PMX固有のアナログ出力信号の実装。



C++

```
class PmxAnalogOutSignal : public AnalogOutSignal, public IPmxSignal;
```

C#

```
public class PmxAnalogOutSignal : AnalogOutSignal, IPmxSignal;
```

Visual Basic

```
Public Class PmxAnalogOutSignal
Inherits AnalogOutSignal
Inherits IPmxSignal
```

ファイル

PmxAnalogOutSignal.cs

備考

この種の信号を測定に含めることはできません。

PmxAnalogOutSignal Members

次の表は、PmxAnalogOutSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

PmxAnalogOutSignal Methods

PmxAnalogOutSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

PmxAnalogOutSignal.GetUniqueID Method

信号が属するデバイス内で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別子。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェア モジュール上の信号を識別するためにストリーミングで使用する情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。

PmxAnalogOutSignal Properties

PmxAnalogOutSignalクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

PmxAnalogOutSignal.Position Property

信号が属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxAnalogOutSignal.Slot Property

信号が属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

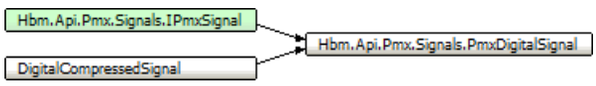
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

PmxDigitalSignal Class

マスクされたビットに応じて、デジタル信号の一部(実際にはPMXの"ネイティブ"デジタル入力またはデジタル出力信号の1ビット)を表します。



C++

```
class PmxDigitalSignal : public DigitalCompressedSignal, public IPmxSignal;
```

C#

```
public class PmxDigitalSignal : DigitalCompressedSignal, IPmxSignal;
```

Visual Basic

```
Public Class PmxDigitalSignal
Inherits DigitalCompressedSignal
Inherits IPmxSignal
```

ファイル

PmxDigitalSignal.cs

PmxDigitalSignal Members

次の表は、PmxDigitalSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	Pin	この信号にマップされているデジタルピン。
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

PmxDigitalSignal Methods

PmxDigitalSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

PmxDigitalSignal.GetUniqueID Method

信号が属するデバイス内で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別子。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェア モジュール上の信号を識別するためにストリーミングで使用する情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。

PmxDigitalSignal Properties

PmxDigitalSignalクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	Pin	この信号にマップされているデジタルピン。
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

PmxDigitalSignal.Pin Property

この信号にマップされているデジタルピン。

C++

```
[JsonProperty]
__property int Pin;
```

C#

```
[JsonProperty]
public int Pin;
```

Visual Basic

```
<JsonProperty>
Public Property Pin() As Integer
```

PmxDigitalSignal.Position Property

信号が属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxDigitalSignal.Slot Property

信号が属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

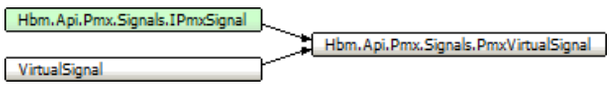
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

PmxVirtualSignal Class

センサを接続しない信号(演算信号など)を表します。



C++

```
class PmxVirtualSignal : public VirtualSignal, public IPmxSignal;
```

C#

```
public class PmxVirtualSignal : VirtualSignal, IPmxSignal;
```

Visual Basic

```
Public Class PmxVirtualSignal
Inherits VirtualSignal
Inherits IPmxSignal
```

ファイル

PmxVirtualSignal.cs

PmxVirtualSignal Members

次の表は、PmxVirtualSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

PmxVirtualSignal Methods

PmxVirtualSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

PmxVirtualSignal.GetUniqueID Method

信号が属するデバイス内で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別子。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェア モジュール上の信号を識別するためにストリーミングで使用する情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。

PmxVirtualSignal Properties

PmxVirtualSignalクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

PmxVirtualSignal.Position Property

信号が属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

PmxVirtualSignal.Slot Property

信号が属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

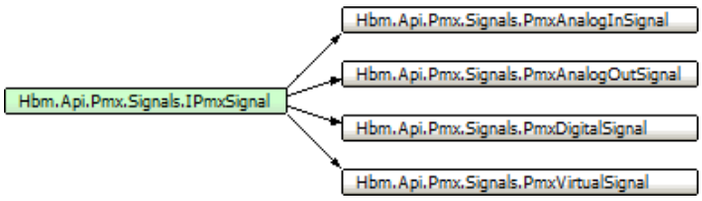
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

IPmxSignal Interface

PmxSignalの追加プロパティのインタフェース。



C++

```
__interface IPmxSignal;
```

C#

```
public interface IPmxSignal;
```

Visual Basic

```
Public Interface IPmxSignal
```

ファイル

IPmxSignal.cs

IPmxSignal Members

次の表は、IPmxSignalによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

IPmxSignal Properties

IPmxSignalクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

IPmxSignal.Position Property

信号が属するスロット内の位置。

C++

```
[JsonProperty]
__property int Position;
```

C#

```
[JsonProperty]
int Position;
```

Visual Basic

```
<JsonProperty>
Property Position() As Integer
```

IPmxSignal.Slot Property

信号が属するデバイスのスロット番号。

C++

```
[JsonProperty]
__property int Slot;
```

C#

```
[JsonProperty]
int Slot;
```

Visual Basic

```
<JsonProperty>
Property Slot() As Integer
```

Hbm.Api.Mgc Namespace

名前空間Hbm.Api.Mgcには主に、MGCデバイスを使用して測定を設定および実行するための機能が含まれています。

したがって、抽象クラスHbm.Api.Common.Device(および抽象クラスHbm.Api.Common.Entities.DeviceFamily)を実装します。

クラス

名前	説明
Hbm.Api.Mgc	これは名前空間Hbm.Api.Mgcです。
Hbm.Api.Mgc.Channels	これは名前空間Hbm.Api.Mgc.Channelsです。
Hbm.Api.Mgc.Connectors	これは名前空間Hbm.Api.Mgc.Connectorsです。
Hbm.Api.Mgc.Enums	これは名前空間Hbm.Api.Mgc.Enumsです。
Hbm.Api.Mgc.Signals	これは名前空間Hbm.Api.Mgc.Signalsです。

トピック

名前	説明
サポートされていないMGCの機能	MGCデバイスファミリのCP52のみ、HBMスキャンメカニズムをサポートしています。CP22やCP42を使用するMGCデバイスはスキャンメカニズムをサポートしていないため、"手動で"接続する必要があります。 次の表に、APIで現在サポートされていないいくつかの機能を示します。
MGCの追加機能	これらはMGCの追加機能です。デバイス経由でもアクセス可能です。これらにアクセスするには、共通デバイス(デバイススキャンで配信など)をMgcDeviceにキャストする必要があります。MGCのすべての追加機能は、AdditionalFeaturesの下にグループ化されています。

サポートされていないMGCの機能

MGCデバイスファミリのCP52のみ、HBMスキャンメカニズムをサポートしています。CP22やCP42を使用するMGCデバイスはスキャンメカニズムをサポートしていないため、"手動で"接続する必要があります。

次の表に、APIで現在サポートされていないいくつかの機能を示します。

機能	説明
トリガ	内部トリガ機構はサポートされていません。
ハードディスク記録	内蔵ハードディスクへの記録はサポートされていません。
シリアルポート	シリアルインタフェースRS-232はサポートされていません。
仮想チャネル	演算した数学チャネルなどの仮想チャネル/信号はサポートされていません。
センサ	次のセンサタイプは実際にはサポートされていません: SSIは通常、MGCではサポートされていません
ファームウェア	MGCデバイスファミリドライバはファームウェアバージョン5.3.2.6285でテスト済みです。

サポートされていないアンプモジュール: ML70、ML71S6、ML77、(ML74は測定のみ - パラメータ化なし)

MGCの追加機能

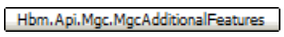
これらはMGCの追加機能です。デバイス経由でもアクセス可能です。これらにアクセスするには、共通デバイス(デバイススキャンで配信など)をMgcDeviceにキャストする必要があります。MGCのすべての追加機能は、AdditionalFeaturesの下にグループ化されています。

クラス

名前	説明
MgcAdditionalFeatures	MgcDeviceのすべての追加機能を保持するクラス。追加機能は、Deviceクラスの抽象関数によって事前に定義されていないすべてのパブリック関数をカバーします。

MgcAdditionalFeatures Class

MgcDeviceのすべての追加機能を保持するクラス。追加機能は、Deviceクラスの抽象関数によって事前に定義されていないすべてのパブリック関数をカバーします。



C++

```
class MgcAdditionalFeatures;
```

C#

```
public class MgcAdditionalFeatures;
```

Visual Basic

```
Public Class MgcAdditionalFeatures
```

ファイル

MgcAdditionalFeatures.cs

MgcAdditionalFeatures Members

次の表は、MgcAdditionalFeaturesによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetDeviceStatus	デバイスの現在のステータスを返します。
	GetTareValue	指定した信号の風袋値を取得します。
	MgcAdditionalFeatures	コンストラクタ。
	ReceiveReply	送信したコマンドの応答を受信します。
	SendCommand	MGCデバイスに直接コマンドを送信し、デバイスの応答を文字列として返します。この関数をバイナリアクセスに使用しないでください(連続測定機能の更新など)。
	SendCommandWithoutReply	応答を待たずに、コマンドをデバイスに直接送信します。
	SetSyncMode	データ取得に関連するデバイスの役割を定義する同期モードを設定します。
	SetTareValue	指定した信号の風袋値を設定します。
	SetTimeChannelFormat	データ取得に使用する時間チャネルタイプを設定します。現在、常に64ビットNTP時間チャネルを使用します。時間チャネルの他のタイプはまだサポートされていません。

MgcAdditionalFeatures.MgcAdditionalFeatures Constructor

コンストラクタ。

C++

```
MgcAdditionalFeatures(
    MgcDevice mgcDevice
);
```

C#

```
public MgcAdditionalFeatures(
    MgcDevice mgcDevice
);
```

Visual Basic

```
Public Sub New(
    mgcDevice As MgcDevice
)
```

引数

引数	説明
mgcDevice	追加機能をサポートするMGCデバイス。

MgcAdditionalFeatures Methods

MgcAdditionalFeaturesクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetDeviceStatus	デバイスの現在のステータスを返します。
	GetTareValue	指定した信号の風袋値を取得します。
	ReceiveReply	送信したコマンドの応答を受信します。
	SendCommand	MGCデバイスに直接コマンドを送信し、デバイスの応答を文字列として返します。この関数をバイナリアクセスに使用しないでください(連続測定機能の更新など)。
	SendCommandWithoutReply	応答を待たずに、コマンドをデバイスに直接送信します。
	SetSyncMode	データ取得に関連するデバイスの役割を定義する同期モードを設定します。
	SetTareValue	指定した信号の風袋値を設定します。
	SetTimeChannelFormat	データ取得に使用する時間チャネルタイプを設定します。現在、常に64ビットNTP時間チャネルを使用します。時間チャネルの他のタイプはまだサポートされていません。

MgcAdditionalFeatures.GetDeviceStatus Method

デバイスの現在のステータスを返します。

C++

```
MgcDeviceStatus GetDeviceStatus();
```

C#

```
public MgcDeviceStatus GetDeviceStatus();
```

Visual Basic

```
Public Function GetDeviceStatus() As MgcDeviceStatus
```

戻り値

デバイスのステータス。Enums.MgcDeviceStatusを参照してください。

MgcAdditionalFeatures.GetTareValue Method

指定した信号の風袋値を取得します。

C++

```
double GetTareValue(
    Signal signal
);
```

C#

```
public double GetTareValue(
    Signal signal
);
```

Visual Basic

```
Public Function GetTareValue(
    signal As Signal
) As double
```

引数

引数	説明
signal	風袋値を取得する信号。

戻り値

指定した信号の風袋値。

MgcAdditionalFeatures.ReceiveReply Method

以前送信したコマンドの応答を受信します。

C++

```
string ReceiveReply();
```

C#

```
public string ReceiveReply();
```

Visual Basic

```
Public Function ReceiveReply() As string
```

戻り値

デバイスからの応答。

MgcAdditionalFeatures.SendCommand Method

MGCデバイスに直接コマンドを送信し、デバイスの応答を文字列として返します。この関数をバイナリアクセスに使用しないでください(連続測定機能の更新など)。

C++

```
string SendCommand(
    string directAccessCommand
);
```

C#

```
public string SendCommand(
    string directAccessCommand
);
```

Visual Basic

```
Public Function SendCommand(
    directAccessCommand As string
) As string
```

引数

引数	説明
directAccessCommand	

戻り値

directAccessCommandに対するデバイスの応答。

MgcAdditionalFeatures.SendCommandWithoutReply Method

応答を待たずに、コマンドをデバイスに直接送信します。

C++

```
void SendCommandWithoutReply(
    string directAccessCommand
);
```

C#

```
public void SendCommandWithoutReply(
    string directAccessCommand
);
```

Visual Basic

```
Public Function SendCommandWithoutReply(
    directAccessCommand As string
) As void
```

引数

引数	説明
directAccessCommand	送信コマンド。

MgcAdditionalFeatures.SetSyncMode Method

データ取得に関連するデバイスの役割を定義する同期モードを設定します。

C++

```
void SetSyncMode(
    SyncModeType syncMode
);
```

C#

```
public void SetSyncMode(
    SyncModeType syncMode
);
```

Visual Basic

```
Public Function SetSyncMode(
    syncMode As SyncModeType
) As void
```

引数

引数	説明
syncMode	設定する同期モード。

MgcAdditionalFeatures.SetTareValue Method

指定した信号の風袋値を設定します。

C++

```
void SetTareValue(
    Signal signal
    double tareValue
);
```

C#

```
public void SetTareValue(
    Signal signal
    double tareValue
);
```

Visual Basic

```
Public Function SetTareValue(
    signal As Signal
    tareValue As double
) As void
```

引数

引数	説明
signal	指定した風袋値を設定する信号。
tareValue	設定する風袋値。

戻り値

指定した信号の風袋値。

MgcAdditionalFeatures.SetTimeChannelFormat Method

データ取得に使用する時間チャネルタイプを設定します。現在、常に64ビットNTP時間チャネルを使用します。時間チャネルの他のタイプはまだサポートされていません。

C++

```
void SetTimeChannelFormat(
    TimeChannelType timeChannelType
);
```

C#

```
public void SetTimeChannelFormat(
    TimeChannelType timeChannelType
);
```

Visual Basic

```
Public Function SetTimeChannelFormat(
    timeChannelType As TimeChannelType
) As void
```

引数

引数	説明
timeChannelType	設定する風袋値。

現在位置: Supported devices -> Hbm.Api.Mgc Namespace ->
MGCの追加機能 -> MgcAdditionalFeatures Class ->
MgcAdditionalFeatures Methods

Hbm.Api.Mgc Namespace

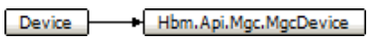
これは名前空間Hbm.Api.Mgcです。

クラス

名前	説明
MgcDevice	MGCデバイスを表します。
MgcDeviceFamily	MGCデバイスの具体的なデバイスファミリの実装。
MgcIndicationScalingValues	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint。
MgcLimitSwitch	リミットスイッチを記述するクラス。

MgcDevice Class

MGCデバイスを表します。



C++

```
class MgcDevice : public Device;
```

C#

```
public class MgcDevice : Device;
```

Visual Basic

```
Public Class MgcDevice
Inherits Device
```

ファイル

MgcDevice.cs

MgcDevice Members

次の表は、MgcDeviceによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	ActivateTeds	指定したコネクタのTEDS設定をアクティブにし、新しいチャネルパラメータをロードします。
	AssignAutoCalibrationMode	指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignChannel	指定したチャネルの設定(センサ、信号などを含む)をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignConnector	指定したコネクタの設定(チャネル、センサ信号などを含む)をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSensor	指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSignal	指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignTimeSource	TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignZero	指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	ChangeIp	デバイスのipAddressを変更します。
	FlashLed	これは、FlashLedメソッドの多重定義の概要です。
	GetAmplifierSerialNumber	チャネルのアンプのシリアル番号を取得します。
	GetAmplifierType	チャネルのアンプタイプを取得します。
	GetChannelHardwareInfo	チャネルのハードウェア情報(アンプの種類など)を取得します。
	GetConnectionBoardSerialNumber	チャネルに接続しているボードのシリアル番号を取得します。
	GetConnectorOfSlotAndPosition	指定されたスロットと位置に属するコネクタを検索します。
	GetDeviceErrorStatusList	これは、GetDeviceErrorStatusListメソッドの多重定義の概要です。
	GetDeviceStatusList	これは、GetDeviceStatusListメソッドの多重定義の概要です。
	GetPositionNumberOfSignal	信号の種類に関係なく、信号の位置番号を返します。
	GetPossibleDataAcquisitionModes	デバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブル経由で確立)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedを含みます。デバイスが時間ベースの同期(PTPまたはNTPなど)を使用している場合、返されるリストにはDataAcquisitionMode.TimestampSynchronizedモードを含みます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはDataAcquisitionMode.Unsynchronizedモードを含む可能性が高くなります。デバイスが接続されていない場合、使用可能なデータ収集モードのリストは空になります。
	GetShunt	これは、GetShuntメソッドの多重定義の概要です。
	GetSignalOfSlotAndPosition	指定したスロットと位置に属する信号を検索します。
	GetSlotNumberOfSignal	信号の種類に関係なく、信号のスロット番号を返します。
	GetSupportedCanBitRates	サポートしているCANのビットレートを取得します。

	GetSupportedFilterFrequencies	指定したフィルタのサポートされているフィルタ周波数のリストを取得します。
	GetSupportedFilters	指定した信号のサポートされているフィルタのリストを取得します。
	GetSupportedJackType	指定した信号でサポートしているジャックの種類を取得します。
	GetSupportedSampleRates	サポートしているサンプルレートのリストを取得します。
	GetSupportedScalingTypes	指定したチャネルでサポートしているスケーリングタイプのリストを取得します。
	GetSupportedSensorTypes	指定したチャネルでサポートしているセンサタイプのリストを取得します。
	GetTimeSourceQuality	オフセット(同期タイムソースとデバイスの時間の間のタイムスパン)をミリ秒単位で返し、同期の品質に関する詳細情報を含むTimeSourceに依存する文字列を返します。NtpTimeSourceの場合、この文字列にはパラメータ(カンマ区切り)が含まれます: "remote、refid、st、t、when、pol、reach、delay、offset、jitter"。PtpTimeSourceの場合、この文字列には次のパラメータ(カンマ区切り)が含まれます: "GrandmasterID、SyncMode"、TimeScaleFlags、UtcOffset、UtcOffsetused、MasterOffset"
	IsFirmwareVersionSupported	指定したファームウェアのバージョンをサポートしているかどうかを判断します。
	MgcDevice	これは、MgcDeviceコンストラクタの多重定義の概要です。
	ReadSingleMeasurementValue	DAQセッションを完全に初期化せずに、指定したSignalごとの単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadSingleMeasurementValueOfAllSignals	DAQセッションを完全に初期化せずに、デバイスのすべてのSignalに対して単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadTeds	指定したコネクタに接続されているTEDSからセンサを読み取ります。
	RemoveInputSignalReference	指定したチャネルから入力信号のリファレンスを削除します。
	Reset	デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数の呼び出し後に、デバイスは切断されます。これが必要なのはデバイスのパラメータ設定が変更され、デバイスがデバイスの関連オブジェクトモデルと一致しなくなったとき、工場出荷時のデフォルトをロードします(この関数を実装するときは、必ずデバイスを切断し、Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知します。
	SetAnalogOut	アナログ出力信号を指定した値に設定します。
	SetCanBusParameters	すべての下層チャネル、センサなどを設定することなく、canconnector.supplyvoltage、canconnector.bitrateなどのCAN bus固有のプロパティを設定します。
	SetCanChannelActivation	提供されたチャネルのリストをアクティブまたは非アクティブにします。T:Hbm.Api.Common.Entities.Channels.CanInChannelおよびT:Hbm.Api.Common.Entities.Channels.CanOutChannelのみをアクティブ化または非アクティブ化できます。
	SetChannelName	指定したチャネルにチャネル名を割り当てます。
	SetDataRateDomain	dataRateDomainを設定します。
	SetDeviceName	デバイスの名前を設定します。MGCデバイスの名前を変更することはできません!
	SetDigitalIO	指定した値に従ってデジタル信号を設定します。
	SetInputSignalReference	入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。

	SetShunt	これは、SetShuntメソッドの多重定義の概要です。
	SetSignalName	指定した信号に信号名を割り当てます。
	SetTedsUsageMode	TEDS使用モードを設定し、コネクタ設定を再読み込みします。
	SetUnit	指定したチャネルの単位を割り当てます。
	WriteTeds	指定したコネクタに接続されたTEDSに指定したセンサを書き込みます。

Public Properties

	名前	説明
	AdditionalFeatures	MGCデバイスの追加機能。
R	FamilyName	このデバイスが属するファミリの名前。
	MeasurementSignalType	データ取得に使用する測定信号の種類。これはすべての測定タスクに有効です。測定の実行中にこのプロパティを変更しても、取得した測定値には影響しません。測定を準備して開始する前に、このプロパティを変更してください。デフォルトはMeasurementSignalType.NetSignalTypeです。デバイスを接続すると、MGCの風袋が常に0.0に設定されるため、両方の信号タイプ間で測定値に違いはありません。この機能は、実際には特定のケースでリンクリソースの競合を防ぐためにのみ使用されます(例: ML77はグロス信号を並列に要求します)。

MgcDevice Constructor

これは、MgcDeviceコンストラクタの多重定義の概要です。

多重定義リスト

	名前	説明
	MgcDevice.MgcDevice ()	クラスの新しいインスタンスを作成する。
	MgcDevice.MgcDevice (string, int)	クラスの新しいインスタンスを作成する。

MgcDevice.MgcDevice Constructor ()

クラスの新しいインスタンスを作成する。

C++

```
MgcDevice;
```

C#

```
public MgcDevice;
```

Visual Basic

```
Public Sub New()
```

MgcDevice.MgcDevice Constructor (string, int)

クラスの新しいインスタンスを作成する。

C++

```
MgcDevice(
    string ip4Address,
    int port = MGCDEFAULTPORT
);
```

C#

```
public MgcDevice(
    string ip4Address,
    int port
);
```

Visual Basic

```
Public Sub New(
    ip4Address As string,
    port As Integer= MGCDEFAULTPORT
)
```

引数

引数	説明
ip4Address	デバイスの IP アドレス。
port	デバイスに接続するポート。

MgcDevice Methods

MgcDeviceクラスのメソッドの一覧です。

Public Methods

	名前	説明
	ActivateTeds	指定したコネクタのTEDS設定をアクティブにし、新しいチャネルパラメータをロードします。
	AssignAutoCalibrationMode	指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignChannel	指定したチャネルの設定(センサ、信号などを含む)をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignConnector	指定したコネクタの設定(チャネル、センサ信号などを含む)をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSensor	指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignSignal	指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignTimeSource	TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	AssignZero	指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。
	ChangeIp	デバイスのipAddressを変更します。
	FlashLed	これは、FlashLedメソッドの多重定義の概要です。
	GetAmplifierSerialNumber	チャネルのアンプのシリアル番号を取得します。
	GetAmplifierType	チャネルのアンプタイプを取得します。
	GetChannelHardwareInfo	チャネルのハードウェア情報(アンプの種類など)を取得します。
	GetConnectionBoardSerialNumber	チャネルに接続しているボードのシリアル番号を取得します。
	GetConnectorOfSlotAndPosition	指定されたスロットと位置に属するコネクタを検索します。
	GetDeviceErrorStatusList	これは、GetDeviceErrorStatusListメソッドの多重定義の概要です。
	GetDeviceStatusList	これは、GetDeviceStatusListメソッドの多重定義の概要です。
	GetPositionNumberOfSignal	信号の種類に関係なく、信号の位置番号を返します。
	GetPossibleDataAcquisitionModes	デバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブル経由で確立)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedを含みます。デバイスが時間ベースの同期(PTPまたはNTPなど)を使用している場合、返されるリストにはDataAcquisitionMode.TimestampSynchronizedモードを含みます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはDataAcquisitionMode.Unsynchronizedモードを含む可能性が高くなります。デバイスが接続されていない場合、使用可能なデータ収集モードのリストは空になります。
	GetShunt	これは、GetShuntメソッドの多重定義の概要です。
	GetSignalOfSlotAndPosition	指定したスロットと位置に属する信号を検索します。
	GetSlotNumberOfSignal	信号の種類に関係なく、信号のスロット番号を返します。
	GetSupportedCanBitRates	サポートしているCANのビットレートを取得します。
	GetSupportedFilterFrequencies	指定したフィルタのサポートされているフィルタ周波数のリストを取得します。

	GetSupportedFilters	指定した信号のサポートされているフィルタのリストを取得します。
	GetSupportedJackType	指定した信号でサポートしているジャックの種類を取得します。
	GetSupportedSampleRates	サポートしているサンプルレートのリストを取得します。
	GetSupportedScalingTypes	指定したチャネルでサポートしているスケーリングタイプのリストを取得します。
	GetSupportedSensorTypes	指定したチャネルでサポートしているセンサタイプのリストを取得します。
	GetTimeSourceQuality	オフセット(同期タイムソースとデバイスの時間の間のタイムスパン)をミリ秒単位で返し、同期の品質に関する詳細情報を含むTimeSourceに依存する文字列を返します。NtpTimeSourceの場合、この文字列にはパラメータ(カンマ区切り)が含まれます: "remote、refid、st、t、when、pol、reach、delay、offset、jitter"。PtpTimeSourceの場合、この文字列には次のパラメータ(カンマ区切り)が含まれます: "GrandmasterID、SyncMode"、TimeScaleFlags、UtcOffset、UtcOffsetused、MasterOffset"
	IsFirmwareVersionSupported	指定したファームウェアのバージョンをサポートしているかどうかを判断します。
	ReadSingleMeasurementValue	DAQセッションを完全に初期化せずに、指定したSignalごとの単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadSingleMeasurementValueOfAllSignals	DAQセッションを完全に初期化せずに、デバイスのすべてのSignalに対して単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。
	ReadTeds	指定したコネクタに接続されているTEDSからセンサを読み取ります。
	RemoveInputSignalReference	指定したチャネルから入力信号のリファレンスを削除します。
	Reset	デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数の呼び出し後に、デバイスは切断されます。これが必要なのはデバイスのパラメータ設定が変更され、デバイスがデバイスの関連オブジェクトモデルと一致しなくなったとき、工場出荷時のデフォルトをロードします(この関数を実装するときは、必ずデバイスを切断し、Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知します。
	SetAnalogOut	アナログ出力信号を指定した値に設定します。
	SetCanBusParameters	すべての下層チャネル、センサなどを設定することなく、canconnector.supplyvoltage、canconnector.bitrateなどのCAN bus固有のプロパティを設定します。
	SetCanChannelActivation	提供されたチャネルのリストをアクティブまたは非アクティブにします。T:Hbm.Api.Common.Entities.Channels.CanInChannelおよびT:Hbm.Api.Common.Entities.Channels.CanOutChannelのみをアクティブ化または非アクティブ化できます。
	SetChannelName	指定したチャネルにチャネル名を割り当てます。
	SetDataRateDomain	dataRateDomainを設定します。
	SetDeviceName	デバイスの名前を設定します。MGCデバイスの名前を変更することはできません!
	SetDigitalIO	指定した値に従ってデジタル信号を設定します。
	SetInputSignalReference	入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。
	SetShunt	これは、SetShuntメソッドの多重定義の概要です。
	SetSignalName	指定した信号に信号名を割り当てます。
	SetTedsUsageMode	TEDS使用モードを設定し、コネクタ設定を再読み込みします。

	SetUnit	指定したチャネルの単位を割り当てます。
	WriteTeds	指定したコネクタに接続されたTEDSに指定したセンサを書き込みます。

MgcDevice.ActivateTeds Method

指定したコネクタのTEDS設定をアクティブにし、新しいチャネルパラメータをロードします。

C++

```
bool ActivateTeds(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override bool ActivateTeds(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function ActivateTeds(
    connector As Connector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	接続されたTEDSをアクティブ化するコネクタ。
problems	空きまたは警告リスト、およびアクティブ化プロセス中に発生したエラー。

戻り値

TEDS設定がアクティブ化されている場合は`true`、それ以外の場合は`false`。

備考

成功すると、指定したコネクタのセンサオブジェクトと一部のチャネルプロパティが、TEDS設定に従って更新されたバージョンに置き換えられます。

MgcDevice.AssignAutoCalibrationMode Method

指定したチャネルのAutoCalibrationModeをデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignAutoCalibrationMode(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignAutoCalibrationMode(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignAutoCalibrationMode(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

MgcDevice.AssignChannel Method

指定したチャネルの設定(センサ、信号などを含む)をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignChannel(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignChannel(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignChannel(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル 。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	チャネルが`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

MgcDevice.AssignConnector Method

指定したコネクタの設定(チャネル、センサ信号などを含む)をデバイスの物理コネクタに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignConnector(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override bool AssignConnector(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignConnector(
    connector As Connector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	既存の(変更された)コネクタ。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

問題が発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	コネクタが`null`です。
Hbm.Api.Common.Exceptions.UnknownConnectorException	コネクタがこのデバイスに属していない場合にスローされます。

MgcDevice.AssignSensor Method

指定したチャネルのセンサ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignSensor(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignSensor(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignSensor(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	チャネルが`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

MgcDevice.AssignSignal Method

指定した信号の設定をデバイスの物理信号に割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignSignal(
    Signal signal,
    out List<Problem> problems
);
```

C#

```
public override bool AssignSignal(
    Signal signal,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignSignal(
    signal As Signal,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
signal	既存の(変更された)信号。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	信号が`null`です。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

MgcDevice.AssignTimeSource Method

TimeSourceをデバイスに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignTimeSource(
    out List<Problem> problems
);
```

C#

```
public override bool AssignTimeSource(
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignTimeSource(
    problems As out List<Problem>
) As bool
```

引数

引数	説明
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

MgcDevice.AssignZero Method

指定したチャネルのゼロ設定をデバイスの物理チャネルに割り当てます。割り当てプロセス中の警告とエラーは、問題のリストに収集されます。

C++

```
bool AssignZero(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override bool AssignZero(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function AssignZero(
    channel As Channel,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	既存の(変更された)チャネル。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	チャネルが`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

MgcDevice.ChangeIp Method

デバイスのipAddressを変更します。

C++

```
bool ChangeIp(
    string ipAddress,
    string subnetMask,
    bool isDhcp,
    string gateWay,
    unsigned tcpPort,
    string interfaceName,
    out List<Problem> problems
);
```

C#

```
public override bool ChangeIp(
    string ipAddress,
    string subnetMask,
    bool isDhcp,
    string gateWay,
    uint tcpPort,
    string interfaceName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function ChangeIp(
    ipAddress As string,
    subnetMask As string,
    isDhcp As bool,
    gateWay As string,
    tcpPort As UInteger,
    interfaceName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
ipAddress	適用するIPアドレス。
subnetMask	サブネットマスク。
isDhcp	自動割り当てであるかどうかを示す表示器。
gateWay	ゲートウェイ。
tcpPort	TCPポート。
interfaceName	ネットワークインタフェースの名前。
problems	問題のリスト。

戻り値

成功した場合はtrue、それ以外の場合はfalse。

FlashLed Method

これは、FlashLedメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	MgcDevice.FlashLed (Connector, LedFlashMode, out List<Problem>)	指定したコネクタのLEDを、指定したモード/色で点滅させます。
	MgcDevice.FlashLed (LedFlashMode, out List<Problem>)	デバイスのLEDを指定したモード/色で点滅させます。
	MgcDevice.FlashLed (List<Connector>, LedFlashMode, out List<Problem>)	指定したコネクタのLEDを、指定したモード/色で点滅させます。

MgcDevice.FlashLed Method (Connector, LedFlashMode, out List<Problem>)

指定したコネクタのLEDを、指定したモード/色で点滅させます。

C++

```
bool FlashLed(
    Connector connector,
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool FlashLed(
    Connector connector,
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function FlashLed(
    connector As Connector,
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	点滅させるコネクタ。
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

現在位置: Supported devices -> Hbm.Api.Mgc Namespace ->
Hbm.Api.Mgc Namespace -> MgcDevice Class ->
MgcDevice Methods -> FlashLed Method

MgcDevice.FlashLed Method (LedFlashMode, out List<Problem>)

デバイスのLEDを指定したモード/色で点滅させます。

C++

```
bool FlashLed(
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool FlashLed(
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function FlashLed(
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

MgcDevice.FlashLed Method (List<Connector>, LedFlashMode, out List<Problem>)

指定したコネクタのLEDを、指定したモード/色で点滅させます。

C++

```
bool FlashLed(
    List<Connector> connectors,
    LedFlashMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool FlashLed(
    List<Connector> connectors,
    LedFlashMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function FlashLed(
    connectors As List<Connector>,
    mode As LedFlashMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	点滅させるコネクタ。
mode	点滅に使用するモード/色。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生していない場合は`true`、それ以外の場合は`false`。

現在位置: Supported devices -> Hbm.Api.Mgc Namespace -> Hbm.Api.Mgc Namespace -> MgcDevice Class -> MgcDevice Methods -> FlashLed Method

MgcDevice.GetAmplifierSerialNumber Method

チャネルのアンプのシリアル番号を取得します。

C++

```
string GetAmplifierSerialNumber(
    Channel channel
);
```

C#

```
public override string GetAmplifierSerialNumber(
    Channel channel
);
```

Visual Basic

```
Public  override Function GetAmplifierSerialNumber(
    channel As Channel
) As string
```

引数

引数	説明
channel	アンプのシリアル番号を返すチャネル。

戻り値

指定したチャネルのアンプのシリアル番号。

MgcDevice.GetAmplifierType Method

チャネルのアンプタイプを取得します。

C++

```
string GetAmplifierType(
    Channel channel
);
```

C#

```
public override string GetAmplifierType(
    Channel channel
);
```

Visual Basic

```
Public override Function GetAmplifierType(
    channel As Channel
) As string
```

引数

引数	説明
channel	アンプのタイプを返すチャネル。

戻り値

指定したチャネルのアンプタイプ。

MgcDevice.GetChannelHardwareInfo Method

チャネルのハードウェア情報(アンプの種類など)を取得します。

C++

```
string GetChannelHardwareInfo(
    Channel channel
);
```

C#

```
public override string GetChannelHardwareInfo(
    Channel channel
);
```

Visual Basic

```
Public override Function GetChannelHardwareInfo(
    channel As Channel
) As string
```

引数

引数	説明
channel	ハードウェア情報が返されるチャネル。

戻り値

指定したチャネルのハードウェア情報。

MgcDevice.GetConnectionBoardSerialNumber Method

チャネルに接続しているボードのシリアル番号を取得します。

C++

```
string GetConnectionBoardSerialNumber(
    Channel channel
);
```

C#

```
public string GetConnectionBoardSerialNumber(
    Channel channel
);
```

Visual Basic

```
Public Function GetConnectionBoardSerialNumber(
    channel As Channel
) As string
```

引数

引数	説明
channel	接続ボードのシリアル番号を返すチャネル。

戻り値

指定されたチャネルの接続ボードシリアル番号。

例外

例外	説明
NotImplementedException	

MgcDevice.GetConnectorOfSlotAndPosition Method

指定されたスロットと位置に属するコネクタを検索します。

C++

```
Connector GetConnectorOfSlotAndPosition(
    int slot,
    int position
);
```

C#

```
public Connector GetConnectorOfSlotAndPosition(
    int slot,
    int position
);
```

Visual Basic

```
Public Function GetConnectorOfSlotAndPosition(
    slot As Integer,
    position As Integer
) As Connector
```

引数

引数	説明
slot	コネクタが配置されているスロット。
position	信号が配置されているスロット内の位置。

戻り値

指定されたスロット/位置で見つかったコネクタ、または対応するコネクタが見つからない場合はnull。

GetDeviceErrorStatusList Method

これは、GetDeviceErrorStatusListメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	MgcDevice.GetDeviceErrorStatusList (Channel, out List<Problem>)	指定したチャネルのデバイスエラーのステータスの一覧を取得します。
	MgcDevice.GetDeviceErrorStatusList (Connector, out List<Problem>)	指定したコネクタのデバイスエラーのステータスの一覧を取得します。
	MgcDevice.GetDeviceErrorStatusList (out List<Problem>)	デバイスエラーのステータスの一覧を取得します。

MgcDevice.GetDeviceErrorStatusList Method (Channel, out List<Problem>)

指定したチャネルのデバイスエラーのステータスの一覧を取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Channel channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceErrorStatusList(
    channel As Channel,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
channel	ステータスを取得するチャネル。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

デバイスエラーのステータスの一覧。

MgcDevice.GetDeviceErrorStatusList Method (Connector, out List<Problem>)

指定したコネクタのデバイスエラーのステータスの一覧を取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceErrorStatus> GetDeviceErrorStatusList(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceErrorStatusList(
    connector As Connector,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
connector	ステータスを取得するコネクタ。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

デバイスエラーのステータスの一覧。

MgcDevice.GetDeviceErrorStatusList Method (out List<Problem>)

デバイスエラーのステータスの一覧を取得します。

C++

```
List<DeviceErrorStatus> GetDeviceErrorStatusList(
    out List<Problem> problems
);
```

C#

```
public override List<DeviceErrorStatus> GetDeviceErrorStatusList(
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceErrorStatusList(
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

デバイスエラーのステータスの一覧。

GetDeviceStatusList Method

これは、GetDeviceStatusListメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	MgcDevice.GetDeviceStatusList (Channel, out List<Problem>)	指定したチャネルのデバイスステータスの一覧を取得します。
	MgcDevice.GetDeviceStatusList (Connector, out List<Problem>)	指定したコネクタのデバイスステータスの一覧を取得します。
	MgcDevice.GetDeviceStatusList (out List<Problem>, StatusType)	デバイスステータスの一覧を取得します。

MgcDevice.GetDeviceStatusList Method (Channel, out List<Problem>)

指定したチャネルのデバイスステータスの一覧を取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    Channel channel,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceStatus> GetDeviceStatusList(
    Channel channel
    out List<Problem> problems,
);
```

Visual Basic

```
Public override Function GetDeviceStatusList(
    channel As Channel,
    problems As out List<Problem>
) As List<DeviceStatus>
```

引数

引数	説明
channel	ステータスを取得するチャネル。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

指定したチャネルのデバイスステータスのリスト。

MgcDevice.GetDeviceStatusList Method (Connector, out List<Problem>)

指定したコネクタのデバイスステータスの一覧を取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    Connector connector,
    out List<Problem> problems
);
```

C#

```
public override List<DeviceStatus> GetDeviceStatusList(
    Connector connector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function GetDeviceStatusList(
    connector As Connector,
    problems As out List<Problem>
) As List<DeviceErrorStatus>
```

引数

引数	説明
connector	ステータスを取得するコネクタ。
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。

戻り値

指定したコネクタのデバイスステータスのリスト。

現在位置: Supported devices -> Hbm.Api.Mgc Namespace ->
Hbm.Api.Mgc Namespace -> MgcDevice Class ->
MgcDevice Methods -> GetDeviceStatusList Method

MgcDevice.GetDeviceStatusList Method (out List<Problem>, StatusType)

デバイスステータスの一覧を取得します。

C++

```
List<DeviceStatus> GetDeviceStatusList(
    out List<Problem> problems,
    StatusType statusTypesToGet = StatusType.All
);
```

C#

```
public override List<DeviceStatus> GetDeviceStatusList(
    out List<Problem> problems,
    StatusType statusTypesToGet
);
```

Visual Basic

```
Public override Function GetDeviceStatusList(
    problems As out List<Problem>,
    statusTypesToGet As StatusType = StatusType.All
) As List<DeviceStatus>
```

引数

引数	説明
problems	プロセス中に発生した警告やエラーのリスト、または空のリスト。
statusTypesToGet	要求したステータスタイプを含むT:Hbm.Api.Common.Enums.StatusType(デフォルトはすべてのステータス情報)。

戻り値

指定したチャネルのデバイスステータスのリスト。

MgcDevice.GetPositionNumberOfSignal Method

信号の種類に関係なく、信号の位置番号を返します。

C++

```
int GetPositionNumberOfSignal(
    Signal signal
);
```

C#

```
public int GetPositionNumberOfSignal(
    Signal signal
);
```

Visual Basic

```
Public Function GetPositionNumberOfSignal(
    signal As Signal
) As Integer
```

引数

引数	説明
signal	位置を返す信号。

戻り値

信号の位置番号。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

MgcDevice.GetPossibleDataAcquisitionModes Method

デバイスの現在の同期設定に従って、可能なデータ収集モードを取得します。デバイスがハードウェア同期(通常は同期ケーブル経由で確立)を検出すると、返されるリストにはモードDataAcquisitionMode.HardwareSynchronizedを含みます。デバイスが時間ベースの同期(PTPまたはNTPなど)を使用している場合、返されるリストにはDataAcquisitionMode.TimestampSynchronizedモードを含みます。ほとんどのデバイスは非同期測定を実行できるため、返されるリストにはDataAcquisitionMode.Unsynchronizedモードを含む可能性が高くなります。デバイスが接続されていない場合、使用可能なデータ収集モードのリストは空になります。

C++

```
List<DataAcquisitionMode> GetPossibleDataAcquisitionModes();
```

C#

```
public override List<DataAcquisitionMode> GetPossibleDataAcquisitionModes();
```

Visual Basic

```
Public override Function GetPossibleDataAcquisitionModes() As List<DataAcquisitionMode>
```

戻り値

現在の同期設定に応じて、このデバイスに使用できるDataAcquisitionModeのリスト。

GetShunt Method

これは、GetShuntメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	MgcDevice.GetShunt (Connector)	指定したコネクタのShuntModeを取得します。
	MgcDevice.GetShunt (List<Connector>)	指定したコネクタのShuntModeを取得します。

MgcDevice.GetShunt Method (Connector)

指定したコネクタのShuntModeを取得します。

C++

```
ShuntMode GetShunt(
    Connector connector
);
```

C#

```
public override ShuntMode GetShunt(
    Connector connector
);
```

Visual Basic

```
Public override Function GetShunt(
   connector As Connector
) As ShuntMode
```

引数

引数	説明
connector	ShuntModeを取得するコネクタ。

戻り値

指定されたコネクタのShuntMode。

MgcDevice.GetShunt Method (List<Connector>)

指定したコネクタのShuntModeを取得します。

C++

```
List<ShuntMode> GetShunt(
    List<Connector> connectors
);
```

C#

```
public override List<ShuntMode> GetShunt(
    List<Connector> connectors
);
```

Visual Basic

```
Public override Function GetShunt(
   connectors As List<Connector>
) As List<ShuntMode>
```

引数

引数	説明
connectors	ShuntModeを取得するコネクタのリスト。

戻り値

指定されたコネクタのShuntModeのリスト。

MgcDevice.GetSignalOfSlotAndPosition Method

指定したスロットと位置に属する信号を検索します。

C++

```
Signal GetSignalOfSlotAndPosition(
    int slot,
    int position
);
```

C#

```
public Signal GetSignalOfSlotAndPosition(
    int slot,
    int position
);
```

Visual Basic

```
Public Function GetSignalOfSlotAndPosition(
    slot As Integer,
    position As Integer
) As Signal
```

引数

引数	説明
slot	信号が属しているスロット。
position	信号が属しているスロット内の位置。

戻り値

指定されたスロット/位置で見つかった信号、または対応する信号が見つからなかった場合はnull。

MgcDevice.GetSlotNumberOfSignal Method

信号の種類に関係なく、信号のスロット番号を返します。

C++

```
int GetSlotNumberOfSignal(
    Signal signal
);
```

C#

```
public int GetSlotNumberOfSignal(
    Signal signal
);
```

Visual Basic

```
Public Function GetSlotNumberOfSignal(
    signal As Signal
) As Integer
```

引数

引数	説明
signal	位置を返す信号。

戻り値

信号のスロット番号。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

MgcDevice.GetSupportedCanBitRates Method

サポートしているCANのビットレートを取得します。

C++

```
List<int> GetSupportedCanBitRates();
```

C#

```
public override List<int> GetSupportedCanBitRates();
```

Visual Basic

```
Public override Function GetSupportedCanBitRates() As List<int>
```

戻り値

サポートしている CANビットレートのリスト。

MgcDevice.GetSupportedFilterFrequencies Method

指定したフィルタのサポートされているフィルタ周波数のリストを取得します。

C++

```
List<decimal> GetSupportedFilterFrequencies(
    Filter Filter,
    bool isOffineQuery = false
);
```

C#

```
public override List<decimal> GetSupportedFilterFrequencies(
    Filter Filter,
    bool isOffineQuery
);
```

Visual Basic

```
Public override Function GetSupportedFilterFrequencies(
   Filter As Filter
   isOffineQuery As bool = false
) As List<decimal>
```

引数

引数	説明
filter	取得したいフィルタの対象となるフィルタ周波数。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なフィルタ周波数のリスト。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownFilterException	フィルタがこのデバイスに属していない場合にスローされます。

MgcDevice.GetSupportedFilters Method

指定した信号のサポートされているフィルタのリストを取得します。

C++

```
List<Filter> GetSupportedFilters(
    Signal signal,
    bool isOfflineQuery = false
);
```

C#

```
public override List<Filter> GetSupportedFilters(
    Signal signal,
    bool isOfflineQuery
);
```

Visual Basic

```
Public override Function GetSupportedFilters(
   signal As Signal
   isOfflineQuery As bool = false
) As List<Filter>
```

引数

引数	説明
signal	取得したいフィルタの対象となる信号。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なフィルタのリスト。

例外

例外	説明
System.ArgumentNullException	信号がnullです。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。

MgcDevice.GetSupportedJackType Method

指定した信号でサポートしているジャックの種類を取得します。

C++

```
string GetSupportedJackType(
    Signal signal,
    bool isOffineQuery = false
);
```

C#

```
public override string GetSupportedJackType(
    Signal signal,
    bool isOffineQuery
);
```

Visual Basic

```
Public override Function GetSupportedJackType(
   signal As Signal
   isOffineQuery As bool = false
) As List<Filter>
```

引数

引数	説明
signal	信号。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

ジャックの種類名。

現在位置: Supported devices -> Hbm.Api.Mgc Namespace ->
Hbm.Api.Mgc Namespace -> MgcDevice Class ->
MgcDevice Methods

MgcDevice.GetSupportedSampleRates Method

サポートしているサンプルレートのリストを取得します。

C++

```
List<decimal> GetSupportedSampleRates(
    Signal signal,
    bool isOfflineQuery = false
);
```

C#

```
public override List<decimal> GetSupportedSampleRates(
    Signal signal,
    bool isOfflineQuery
);
```

Visual Basic

```
Public override Function GetSupportedSampleRates(
   signal As Signal
   isOfflineQuery As bool = false
) As List<decimal>
```

引数

引数	説明
signal	取得したいサンプルレートの信号。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なサンプルレートのリスト。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.UnknownSignalException	信号がこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

MgcDevice.GetSupportedScalingTypes Method

指定したチャネルでサポートしているスケーリングタイプのリストを取得します。

C++

```
List<ScalingType> GetSupportedScalingTypes(
    Channel channel,
    bool isOffineQuery = false
);
```

C#

```
public override List<ScalingType> GetSupportedScalingTypes(
    Channel channel,
    bool isOffineQuery
);
```

Visual Basic

```
Public override Function GetSupportedScalingTypes(
    channel As Channel,
   isOffineQuery As bool = false
) As List<ScalingType>
```

引数

引数	説明
channel	取得したいスケーリングタイプのチャネル。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なスケーリングタイプのリスト。

戻り値

可能なスケーリングタイプは、チャネルの現在のセンサによっても異なります。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。

現在位置: Supported devices -> Hbm.Api.Mgc Namespace -> Hbm.Api.Mgc Namespace -> MgcDevice Class -> MgcDevice Methods

MgcDevice.GetSupportedSensorTypes Method

指定したチャネルでサポートしているセンサタイプのリストを取得します。

C++

```
List<SensorType> GetSupportedSensorTypes(
    Channel channel,
    bool isOfflineQuery = false
);
```

C#

```
public override List<SensorType> GetSupportedSensorTypes(
    Channel channel,
    bool isOfflineQuery
);
```

Visual Basic

```
Public override Function GetSupportedSensorTypes(
    channel As Channel,
   isOfflineQuery As bool = false
) As List<SensorType>
```

引数

引数	説明
channel	取得したいセンサタイプのチャネル。
isOfflineQuery	True、可能であれば、デバイスと通信せずにフィルタ周波数を返す必要があります(パフォーマンス上の理由など)。

戻り値

可能なセンサタイプのリスト。

例外

例外	説明
System.ArgumentNullException	チャネルがnullです。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

MgcDevice.GetTimeSourceQuality Method

オフセット(同期タイムソースとデバイスの時間の間のタイムスパン)をミリ秒単位で返し、同期の品質に関する詳細情報を含むTimeSourceに依存する文字列を返します。NtpTimeSourceの場合、この文字列にはパラメータ(カンマ区切り)が含まれます: "remote、refid、st、t、when、pol、reach、delay、offset、jitter"。PtpTimeSourceの場合、この文字列には次のパラメータ(カンマ区切り)が含まれます: "GrandmasterID、SyncMode"、TimeScaleFlags、UtcOffset、UtcOffsetused、MasterOffset"

C++

```
void GetTimeSourceQuality(
    out double offsetInMs,
    out string quality
);
```

C#

```
public override void GetTimeSourceQuality(
    out double offsetInMs,
    out string quality
);
```

Visual Basic

```
Public override Function GetTimeSourceQuality(
    offsetInMs As out double,
    quality As out string
) As void
```

引数

引数	説明
offsetInMs	タイムソースへのオフセット(ミリ秒単位)。オフセットを決定できない場合は double.MaxValue になります。
quality	TimeSourceタイプに依存する同期品質に関する情報。

MgcDevice.IsFirmwareVersionSupported Method

指定したファームウェアのバージョンをサポートしているかどうかを判断します。

C++

```
bool IsFirmwareVersionSupported(
    string firmwareVersion
);
```

C#

```
public override bool IsFirmwareVersionSupported(
    string firmwareVersion
);
```

Visual Basic

```
Public override Function IsFirmwareVersionSupported(
    firmwareVersion As string
) As bool
```

引数

引数	説明
firmwareVersion	確認するファームウェアのバージョン。

戻り値

指定したファームウェアバージョンがサポートされていない場合はfalseになります。

備考

この関数は、指定したファームウェアバージョンが最新の既知のバージョンよりも高い場合(APIがリリースされた時点)でもtrueを返します。

MgcDevice.ReadSingleMeasurementValue Method

DAQセッションを完全に初期化せずに、指定したSignalごとの単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。

C++

```
void ReadSingleMeasurementValue(
    List<Signal> measurementSignals
);
```

C#

```
public override void ReadSingleMeasurementValue(
    List<Signal> measurementSignals
);
```

Visual Basic

```
Public override Function ReadSingleMeasurementValue(
    measurementSignals As List<Signal>
) As void
```

引数

引数	説明
measurementSignals	測定値を取得する信号。

備考

デバイスが単一の測定値のタイムスタンプをサポートしていない場合、PCの現在の時刻を使用してタイムスタンプを生成します。

例外

例外	説明
System.ArgumentNullException	計測信号がnullです。
System.ArgumentException	計測信号の数が0の場合にスローされます
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

MgcDevice.ReadSingleMeasurementValueOfAllSignals Method

DAQセッションを完全に初期化せずに、デバイスのすべてのSignalに対して単一の測定値を取得します。Filter設定は無視します。測定値はSignalに内部的に保存され、Signal.GetSingleMeasurementValueを呼び出すことによって取得できます。

C++

```
void ReadSingleMeasurementValueOfAllSignals();
```

C#

```
public override void ReadSingleMeasurementValueOfAllSignals();
```

Visual Basic

```
Public override Function ReadSingleMeasurementValueOfAllSignals() As void
```

備考

デバイスが単一の測定値のタイムスタンプをサポートしていない場合、PCの現在の時刻を使用してタイムスタンプを生成します。

MgcDevice.ReadTeds Method

指定したコネクタに接続されているTEDSからセンサを読み取ります。

C++

```
Sensor ReadTeds(
    Connector connector,
    out string tid,
    out List<Problem> problems
);
```

C#

```
public override Sensor ReadTeds(
    Connector connector,
    out string tid,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function ReadTeds(
    connector As Connector,
    tid As out string,
    problems As out List<Problem>
) As Sensor
```

MgcDevice.RemoveInputSignalReference Method

指定したチャネルから入力信号のリファレンスを削除します。

C++

```
bool RemoveInputSignalReference(
    IInputSignalReference channel,
    out List<Problem> problems
);
```

C#

```
public override bool RemoveInputSignalReference(
    IInputSignalReference channel,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function RemoveInputSignalReference(
    channel As IInputSignalReference
    problems As out List<Problem>
) As bool
```

MgcDevice.Reset Method

デバイスを工場出荷時のデフォルト設定にリセットします。注意! この関数の呼び出し後に、デバイスは切断されます。これが必要なのはデバイスのパラメータ設定が変更され、デバイスがデバイスの関連オブジェクトモデルと一致しなくなったとき、工場出荷時のデフォルトをロードします(この関数を実装するときは、必ずデバイスを切断し、Messaging.MessageBroker.RaiseDeviceDisconnectedAsyncを呼び出して、デバイスが切断されたことを通知します。

C++

```
void Reset(
    out List<Problem> problems
);
```

C#

```
public override void Reset(
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function Reset(
    problems As out List<Problem>
) As void
```

引数

引数	説明
problems	デバイスのリセット中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

この関数の実行後にデバイスが切断され、一定時間アクセス不能になる可能性があることに注意してください。

MgcDevice.SetAnalogOut Method

アナログ出力信号を指定した値に設定します。

C++

```
void SetAnalogOut(
    AnalogOutSignal analogOutSignal,
    decimal value
);
```

C#

```
public override void SetAnalogOut(
    AnalogOutSignal analogOutSignal,
    decimal value
);
```

Visual Basic

```
Public override Function SetAnalogOut(
    analogOutSignal As AnalogOutSignal,
    value out decimal
) As void
```

引数

引数	説明
analogOutSignal	設定するアナログ出力信号。
value	設定する値。

例外

例外	説明
System.ArgumentNullException	アナログ出力信号が`null`です。
Hbm.Api.Common.Exceptions.UnknownSignalException	アナログ出力信号がこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

MgcDevice.SetCanBusParameters Method

すべての下層チャネル、センサなどを設定することなく、canconnector.supplyvoltage、canconnector.bitrateなどのCAN bus固有のプロパティを設定します。

C++

```
bool SetCanBusParameters(
    CanConnector canConnector,
    out List<Problem> problems
);
```

C#

```
public override bool SetCanBusParameters(
    CanConnector canConnector,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetCanBusParameters(
    canConnector As CanConnector,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
canConnector	プロパティを割り当てる必要がある既存の(変更された)CanConnector。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

MgcDevice.SetCanChannelActivation Method

提供されたチャネルのリストをアクティブまたは非アクティブにします。T:Hbm.Api.Common.Entities.Channels.CanInChannelおよびT:Hbm.Api.Common.Entities.Channels.CanOutChannelのみをアクティブ化または非アクティブ化できます。

C++

```
bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

C#

```
public override bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetDataRateDomain(
    dataRateDomain As DataRateDomainType,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
dataRateDomain	モジュールに設定するデータレートドメインタイプ。
problems	データレートドメインの設定中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

MgcDevice.SetChannelName Method

指定したチャネルにチャネル名を割り当てます。

C++

```
bool SetChannelName(
    Channel channel,
    string channelName,
    out List<Problem> problems
);
```

C#

```
public override bool SetChannelName(
    Channel channel,
    string channelName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetChannelName(
    channel As Channel,
    channelName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	変更するチャネルの名。
channelName	新しいチャネル名。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

実装では、チャネル名が正常に割り当て後にHbm.Api.Common.Messaging.MessageBroker.RaiseChannelNameChangedAsyncを呼び出す必要があります。

例外

例外	説明
System.ArgumentNullException	チャネルが`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

MgcDevice.SetDataRateDomain Method

dataRateDomainを設定します。

C++

```
bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

C#

```
public bool SetDataRateDomain(
    DataRateDomainType dataRateDomain,
    out List<Problem> problems
);
```

Visual Basic

```
Public Function SetDataRateDomain(
    dataRateDomain As DataRateDomainType,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
dataRateDomain	モジュールに設定するデータレートドメインタイプ。
problems	データレートドメインの設定中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

MgcDevice.SetDeviceName Method

デバイスの名前を設定します。MGCデバイスの名前を変更することはできません!

C++

```
bool SetDeviceName(
    string newName,
    out List<Problem> problems
);
```

C#

```
public override bool SetDeviceName(
    string newName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetDeviceName(
    newName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
newName	新しい名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

現在位置: Supported devices -> Hbm.Api.Mgc Namespace ->
Hbm.Api.Mgc Namespace -> MgcDevice Class ->
MgcDevice Methods

MgcDevice.SetDigitalIO Method

指定した値に従ってデジタル信号を設定します。

C++

```
void SetDigitalIO(
    DigitalSignal digitalSignal,
    DigitalValueType value
);
```

C#

```
public override void SetDigitalIO(
    DigitalSignal digitalSignal,
    DigitalValueType value
);
```

Visual Basic

```
Public override Function SetDigitalIO(
    digitalSignal As DigitalSignal,
    value As out DigitalValueType
) As void
```

引数

引数	説明
digitalSignal	調整するデジタル信号。
value	設定する値。

例外

例外	説明
System.ArgumentNullException	デジタル信号が`null`です。
System.InvalidOperationException	デジタル信号のコネクタが出力操作をサポートしていない場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownSignalException	デジタル信号がこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

MgcDevice.SetInputSignalReference Method

入力信号のリファレンスを指定したチャネルに割り当てます。リファレンスは、入力信号のデバイスと入力信号自体で構成されます。

C++

```
bool SetInputSignalReference(
    IInputSignalReference channel,
    Device inputSignalDevice,
    Signal inputSignal,
    out List<Problem> problems
);
```

C#

```
public override bool SetInputSignalReference(
    IInputSignalReference channel,
    Device inputSignalDevice,
    Signal inputSignal,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetInputSignalReference(
    channel As IInputSignalReference
    inputSignalDevice As Device
    inputSignal As Signal
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	入力信号リファレンスで設定される(出力)チャネル。
inputSignalDevice	入力信号のデバイス。
inputSignal	入力信号
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

例外

例外	説明
System.ArgumentNullException	channel、inputSignalDeviceまたはinputSignalが`null`です。
System.ArgumentException	inputSignalがDigitalSignalまたはAnalogOutSignalの場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownChannelException	チャネルがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.UnknownSignalException	inputSignalがこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

現在位置: Supported devices -> Hbm.Api.Mgc Namespace -> Hbm.Api.Mgc Namespace -> MgcDevice Class -> MgcDevice Methods

SetShunt Method

これは、SetShuntメソッドの多重定義の概要です。

多重定義リスト

	名前	説明
	MgcDevice.SetShunt (Connector, ShuntMode, out List<Problem>)	指定したコネクタのシャント出力の構成を設定します。
	MgcDevice.SetShunt (List<Connector>, ShuntMode, out List<Problem>)	指定したコネクタのシャント出力の構成を設定します。

MgcDevice.SetShunt Method (Connector, ShuntMode, out List<Problem>)

指定したコネクタのシャント出力の構成を設定します。

C++

```
bool SetShunt(
    Connector connector,
    ShuntMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool SetShunt(
    Connector connector,
    ShuntMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetShunt(
    connector As Connector,
    mode As ShuntMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	シャント出力を設定するコネクタ。
mode	指定したコネクタのShuntMode。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

MgcDevice.SetShunt Method (List<Connector>, ShuntMode, out List<Problem>)

指定したコネクタのシャント出力の構成を設定します。

C++

```
bool SetShunt(
    List<Connector> connectors,
    ShuntMode mode,
    out List<Problem> problems
);
```

C#

```
public override bool SetShunt(
    List<Connector> connectors,
    ShuntMode mode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetShunt(
    connectors As List<Connector>,
    mode As ShuntMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connectors	シャント出力を設定するコネクタ。
mode	指定したコネクタのShuntMode。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

MgcDevice.SetSignalName Method

指定した信号に信号名を割り当てます。

C++

```
bool SetSignalName(
    Signal signal,
    string signalName,
    out List<Problem> problems
);
```

C#

```
public override bool SetSignalName(
    Signal signal,
    string signalName,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetSignalName(
    signal As Signal,
    signalName As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
signal	名前を変更する信号。
signalName	信号の新しい名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

備考

実装では、信号名が正常に割り当てられた後にHbm.Api.Common.Messaging.MessageBroker.RaiseSignalNameChangedAsyncを呼び出す必要があります。

例外

例外	説明
System.ArgumentNullException	信号が`null`です。
Hbm.Api.Common.Exceptions.UnknownChannelException	信号がこのデバイスに属していない場合にスローされます。
Hbm.Api.Common.Exceptions.DeviceNotConnectedException	このデバイスが接続されていない場合にスローされます。

MgcDevice.SetTedsUsageMode Method

TEDS使用モードを設定し、コネクタ設定を再読み込みします。

C++

```
bool SetTedsUsageMode(
    Connector connector,
    TedsUsageMode tedsUsageMode,
    out List<Problem> problems
);
```

C#

```
public override bool SetTedsUsageMode(
    Connector connector,
    TedsUsageMode tedsUsageMode,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetTedsUsageMode(
    connector As Connector,
    tedsUsageMode As TedsUsageMode,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
connector	コネクタ。
tedsUsageMode	設定するTEDS使用モード。
problems	出力: 警告やエラーのリスト。何も発生しなかった場合は空です。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

MgcDevice.SetUnit Method

指定したチャネルの単位を割り当てます。

C++

```
bool SetUnit(
    Channel channel,
    string unit,
    out List<Problem> problems
);
```

C#

```
public override bool SetUnit(
    Channel channel,
    string unit,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function SetUnit(
    channel As Channel,
    unit As string,
    problems As out List<Problem>
) As bool
```

引数

引数	説明
channel	単位を設定するチャネル。
unit	信号の新しい名前。
problems	割り当てプロセス中に発生した警告とエラー。

戻り値

エラーが発生しなかった場合は`true`、それ以外の場合は`false`。

現在位置: Supported devices -> Hbm.Api.Mgc Namespace ->
Hbm.Api.Mgc Namespace -> MgcDevice Class ->
MgcDevice Methods

MgcDevice.WriteTeds Method

指定したコネクタに接続されたTEDSに指定したセンサを書き込みます。

C++

```
bool WriteTeds(
    Connector connector,
    out List<Problem> problems,
    Sensor sensor,
    string tid = null,
    string channelName = null
);
```

C#

```
public override bool WriteTeds(
    Connector connector,
    out List<Problem> problems,
    Sensor sensor,
    string tid,
    string channelName
);
```

Visual Basic

```
Public override Function WriteTeds(
    connector As Connector,
    problems As out List<Problem>,
    sensor As Sensor,
    tid As string = null,
    channelName As string = null
) As bool
```

MgcDevice Properties

MgcDeviceクラスのプロパティの一覧です。

Public Properties

	名前	説明
	AdditionalFeatures	MGCデバイスの追加機能。
R	FamilyName	このデバイスが属するファミリの名前。
	MeasurementSignalType	データ取得に使用する測定信号の種類。これはすべての測定タスクに有効です。測定の実行中にこのプロパティを変更しても、取得した測定値には影響しません。測定を準備して開始する前に、このプロパティを変更してください。デフォルトはMeasurementSignalType.NetSignalTypeです。デバイスを接続すると、MGCの風袋が常に0.0に設定されるため、両方の信号タイプ間で測定値に違いはありません。この機能は、実際には特定のケースでリンクリソースの競合を防ぐためにのみ使用されます(例: ML77はグロス信号を並列に要求します)。

MgcDevice.AdditionalFeatures Property

MGCデバイスの追加機能。

C++

```
[JsonIgnore]
__property MgcAdditionalFeatures AdditionalFeatures;
```

C#

```
[JsonIgnore]
public MgcAdditionalFeatures AdditionalFeatures;
```

Visual Basic

```
<JsonIgnore>
Public Property AdditionalFeatures() As MgcAdditionalFeatures
```

備考

追加機能は、Deviceクラスの抽象関数によって事前に定義されていないすべてのパブリック関数をカバーします。例: MgcDeviceのLoadParameterSet()関数はMgcDeviceのこれらの追加機能に属しますが、AssignConnector()はDeviceクラスの抽象関数であるため、追加機能には属しません。

MgcDevice.FamilyName Property

このデバイスが属するファミリの名前。

C++

```
__property string FamilyName;
```

C#

```
public override string FamilyName;
```

Visual Basic

```
Public override Property FamilyName() As string
```

備考

これは、クラスHbm.Api.Mgc.MgcDeviceFamilyのプロパティFamilyNameとまったく同じ文字列でなければなりません。

MgcDevice.MeasurementSignalType Propertys

データ取得に使用する測定信号の種類。これはすべての測定タスクに有効です。測定の実行中にこのプロパティを変更しても、取得した測定値には影響しません。測定を準備して開始する前に、このプロパティを変更してください。デフォルトはMeasurementSignalType.NetSignalTypeです。デバイスを接続すると、MGCの風袋が常に0.0に設定されるため、両方の信号タイプ間で測定値に違いはありません。この機能は、実際には特定のケースでリンクリソースの競合を防ぐためにのみ使用されます(例: ML77はグロス信号を並列に要求します)。

C++

```
__property MeasurementSignalType MeasurementSignalType;
```

C#

```
public MeasurementSignalType MeasurementSignalType;
```

Visual Basic

```
Public Property MeasurementSignalType() As MeasurementSignalType
```

MgcDeviceFamily Class

MGCデバイスの具体的なデバイスファミリの実装。



C++

```
class MgcDeviceFamily : public DeviceFamily;
```

C#

```
public class MgcDeviceFamily : DeviceFamily;
```

Visual Basic

```
Public Class MgcDeviceFamily
Inherits DeviceFamily
```

ファイル

MgcDeviceFamily.cs

MgcDeviceFamily Members

次の表は、MgcDeviceFamilyによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	Find	指定したConnectionInfoに従ってデバイスを検索します。
	GetPossibleDataAcquisitionModes	ファミリの指定したすべてのデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSourceは、ファミリのすべてのデバイスで使用するタイムソースに設定します(指定したデバイスでDataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関連します)。
	MgcDeviceFamily	MgcDeviceFamilyクラスの新しいインスタンスを初期化します。
	PrepareDataAcquisition	指定したDataAcquisitionModeに従って最終的なデバイス設定を行う関数。この関数を使用して、トリガマスタとトリガスレーブなどの設定、指定したデータ収集モードに関連する特定のタイムスタンプ設定の選択などを行います。測定に関与する特定のデバイスファミリのすべてのデバイスもこの関数に渡されるため、特定のデバイスをさまざまな方法で処理できます(たとえば、トリガマスタはスレーブと異なる設定/準備を取得します)。
	Scan	ネットワーク上のMGCデバイスをスキャンします。

Public Properties

	名前	説明
R	FamilyName	デバイスファミリの名前を取得します(読み取り専用)。
R	IsScanSupported	デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。
	SkillReader	SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

MgcDeviceFamily.MgcDeviceFamily Constructor

MgcDeviceFamilyクラスの新しいインスタンスを初期化します。

C++

```
MgcDeviceFamily();
```

C#

```
public MgcDeviceFamily();
```

Visual Basic

```
Public Sub New()
```

MgcDeviceFamily Methods

MgcDeviceFamilyクラスのメソッドの一覧です。

Public Methods

	名前	説明
	Find	指定したConnectionInfoに従ってデバイスを検索します。
	GetPossibleDataAcquisitionModes	ファミリの指定したすべてのデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSourceは、ファミリのすべてのデバイスで使用するタイムソースに設定します(指定したデバイスでDataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関連します)。
	PrepareDataAcquisition	指定したDataAcquisitionModeに従って最終的なデバイス設定を行う関数。この関数を使用して、トリガマスタとトリガスレーブなどの設定、指定したデータ収集モードに関連する特定のタイムスタンプ設定の選択などを行います。測定に関与する特定のデバイスファミリのすべてのデバイスもこの関数に渡されるため、特定のデバイスをさまざまな方法で処理できます(たとえば、トリガマスタはスレーブと異なる設定/準備を取得します)。
	Scan	ネットワーク上のMGCデバイスをスキャンします。

MgcDeviceFamily.Find Method

指定したConnectionInfoに従ってデバイスを検索します。

C++

```
List<Device> Find(
    List<ConnectionInfo> connectionInfos,
    out List<Problem> problems
);
```

C#

```
public override List<Device> Find(
    List<ConnectionInfo> connectionInfos,
    out List<Problem> problems
);
```

Visual Basic

```
Public override Function Find(
    connectionInfos As List<ConnectionInfo>,
    problems As out List<Problem>
) As List<Device>
```

引数

引数	説明
connectionInfos	接続情報。
problems	検索プロセス中に発生した警告とエラー。

戻り値

見つかったすべてのデバイスのリスト。

MgcDeviceFamily.GetPossibleDataAcquisitionModes Method

ファミリの指定したすべてのデバイスを考慮して、可能なデータ取得モードを取得します。CommonTimeSourceは、ファミリのすべてのデバイスで使用するタイムソースに設定します(指定したデバイスでDataAcquisitionMode.TimestampSynchronizedがサポートされている場合にのみ関連します)。

C++

```
List<DataAcquisitionMode> GetPossibleDataAcquisitionModes(
    List<Device> devices,
    out TimeSource commonTimeSource
);
```

C#

```
public override List<DataAcquisitionMode> GetPossibleDataAcquisitionModes(
    List<Device> devices,
    out TimeSource commonTimeSource
);
```

Visual Basic

```
Public override Function GetPossibleDataAcquisitionModes(
    devices As List<Device>,
    commonTimeSource As out TimeSource
) As List<DataAcquisitionMode>
```

引数

引数	説明
devices	ファミリに属し、測定に参加するデバイス。
commonTimeSource	すべてのデバイスでサポートされているTimeSource(通常はNtpTimeSourceなど)。

戻り値

指定したデバイスの可能性に応じた可能なDataAcquisitionMode。

MgcDeviceFamily.PrepareDataAcquisition Method

指定したDataAcquisitionModeに従って最終的なデバイス設定を行う関数。この関数を使用して、トリガマスタとトリガスレーブなどの設定、指定したデータ収集モードに関連する特定のタイムスタンプ設定の選択などを行います。測定に関与する特定のデバイスファミリのすべてのデバイスもこの関数に渡されるため、特定のデバイスをさまざまな方法で処理できます(たとえば、トリガマスタはスレーブと異なる設定/準備を取得します)。

C++

```
void PrepareDataAcquisition(
    DataAcquisitionMode dataAcquisitionMode,
    List<Device> devicesToPrepare
);
```

C#

```
public override void PrepareDataAcquisition(
    DataAcquisitionMode dataAcquisitionMode,
    List<Device> devicesToPrepare
);
```

Visual Basic

```
Public override Function PrepareDataAcquisition(
    dataAcquisitionMode As DataAcquisitionMode,
    devicesToPrepare As List<Device>
) As void
```

引数

引数	説明
dataAcquisitionMode	DataAcquisitionModeを使用して、測定を開始する方法を記述します。
devicesToPrepare	ファミリに属し、測定に参加するデバイス。

戻り値

DataAcquisitionMode.Autoではなく、関数 GetPossibleDataAcquisitionModes(List{Device}, out TimeSource)によって返されるDataAcquisitionModeのいずれかを使用します。

MgcDeviceFamily.Scan Method

ネットワーク上のMGCデバイスをスキャンします。

C++

```
List<Device> Scan();
```

C#

```
public override List<Device>) Scan;
```

Visual Basic

```
Public override Function Scan() As List<Device>
```

戻り値

見つかったMGCデバイスのリスト。

MgcDeviceFamily Properties

MgcDeviceFamilyクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
R	FamilyName	デバイスファミリの名前を取得します(読み取り専用)。
R	IsScanSupported	デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。
	SkillReader	SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

MgcDeviceFamily.FamilyName Property

デバイスファミリの名前を取得します(読み取り専用)。

C++

```
__property string FamilyName;
```

C#

```
public override string FamilyName;
```

Visual Basic

```
Public override Property FamilyName() As string
```

MgcDeviceFamily.IsScanSupported Property

デバイスファミリがスキャンメカニズムをサポートしているかどうかを示します。

C++

```
__property bool IsScanSupported;
```

C#

```
public override bool IsScanSupported;
```

Visual Basic

```
Public override Property IsScanSupported() As bool
```

MgcDeviceFamily.SkillReader Property

SkillReaderのデバイス固有の派生を含むアセンブリへのパス。

C++

```
__property ISkillReader SkillReader;
```

C#

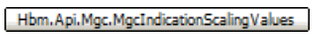
```
public override ISkillReader SkillReader;
```

Visual Basic

```
Public override Property SkillReader() As ISkillReader
```

MgcIndicationScalingValues Class

-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint。



C++

```
class MgcIndicationScalingValues;
```

C#

```
public class MgcIndicationScalingValues;
```

Visual Basic

```
Public Class MgcIndicationScalingValues
```

ファイル

MgcIndicationScalingValues.cs

MgcIndicationScalingValues Members

次の表は、MgcIndicationScalingValuesによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	InputRange	入力範囲。
	ZeroPoint	ゼロ値を表示します。

MgcIndicationScalingValues Properties

MgcIndicationScalingValuesクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	InputRange	入力範囲。
	ZeroPoint	ゼロ値を表示します。

MgcIndicationScalingValues.InputRange Property

入力範囲。

C++

```
__property double InputRange;
```

C#

```
public double InputRange;
```

Visual Basic

```
Public Property InputRange() As double
```

MgcIndicationScalingValues.ZeroPoint Property

ゼロ値を表示します。

C++

```
__property double ZeroPoint;
```

C#

```
public double ZeroPoint;
```

Visual Basic

```
Public Property ZeroPoint() As double
```

MgcLimitSwitch Class

リミットスイッチを記述するクラス。



C++

```
class MgcLimitSwitch;
```

C#

```
public class MgcLimitSwitch;
```

Visual Basic

```
Public Class MgcLimitSwitch
```

ファイル

MgcLimitSwitch.cs

MgcLimitSwitch Members

次の表は、MgcLimitSwitchによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	Enabled	リミット スイッチ操作がアクティブな場合はTrue。リミットスイッチがオフの場合はFalse。
	Hysteresis	ヒステリシス値。`OperationDirection`に依存し、バンドスパンを定義にも使用できます。
	IgnoreMeasurementValueStatus	trueの場合: リミットスイッチステータスの評価中に測定値のステータスは無視されます。falseの場合: 測定値のステータスがOKの場合にのみLimit switchが評価されます。それ以外の場合、リミットスイッチのステータスは固定されます。
	InputSignal	リミットスイッチを評価するために測定値が使用する信号。
	InvertResetBehaviour	trueの場合、定義された`ResetBehavior`が反転されます。
	Limit	制限値。`OperationDirection`に依存し、下限バンド値を定義するためにも使用できます。
	LimitSwitchNumber	リミットスイッチの番号。範囲は1～32です。
	OperatingDirection	リミットスイッチの動作方向。
	ResetBehaviorMask	リセット動作(ヒステリシス管理に使用できます)。すべてのデジタル入力とAND演算されるバイナリマスク。結果が>0 (true)の場合、このリミット スイッチはクリアされます。値の範囲: 0、1、2、4、8、16、...32768は、MGCの16個のデジタル入力に対応します。リセット動作は、`InvertResetBehavior`で反転できます。デフォルト設定は0です。

MgcLimitSwitch Properties

MgcLimitSwitchクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	Enabled	リミット スイッチ操作がアクティブな場合はTrue。リミットスイッチがオフの場合はFalse。
	Hysteresis	ヒステリシス値。`OperationDirection`に依存し、バンドスパンを定義にも使用できます。
	IgnoreMeasurementValueStatus	trueの場合: リミットスイッチステータスの評価中に測定値のステータスは無視されます。falseの場合: 測定値のステータスがOKの場合にのみLimit switchが評価されます。それ以外の場合、リミットスイッチのステータスは固定されます。
	InputSignal	リミットスイッチを評価するために測定値が使用する信号。
	InvertResetBehaviour	trueの場合、定義された`ResetBehavior`が反転されます。
	Limit	制限値。`OperationDirection`に依存し、下限バンド値を定義するためにも使用できます。
	LimitSwitchNumber	リミットスイッチの番号。範囲は1～32です。
	OperatingDirection	リミットスイッチの動作方向。
	ResetBehaviorMask	リセット動作(ヒステリシス管理に使用できます)。すべてのデジタル入力とAND演算されるバイナリマスク。結果が>0 (true)の場合、このリミット スイッチはクリアされます。値の範囲: 0、1、2、4、8、16、...32768は、MGCの16個のデジタル入力に対応します。リセット動作は、`InvertResetBehavior`で反転できます。デフォルト設定は0です。

MgcLimitSwitch.Enabled Property

リミット スイッチ操作がアクティブな場合はTrue。リミットスイッチがオフの場合はFalse。

C++

```
__property bool Enabled;
```

C#

```
public bool Enabled;
```

Visual Basic

```
Public Property Enabled() As bool
```

MgcLimitSwitch.Hysteresis Property

ヒステリシス値。OperationDirectionに依存し、バンドスパンを定義にも使用できます。

C++

```
__property double Hysteresis;
```

C#

```
public double Hysteresis;
```

Visual Basic

```
Public Property Hysteresis() As double
```

MgcLimitSwitch.IgnoreMeasurementValueStatus Property

trueの場合: リミットスイッチステータスの評価中に測定値のステータスは無視されます。falseの場合: 測定値のステータスがOKの場合にのみLimit switchが評価されます。それ以外の場合、リミットスイッチのステータスは固定されます。

C++

```
__property bool IgnoreMeasurementValueStatus;
```

C#

```
public bool IgnoreMeasurementValueStatus;
```

Visual Basic

```
Public Property IgnoreMeasurementValueStatus() As bool
```

MgcLimitSwitch.InputSignal Property

リミットスイッチを評価するために測定値が使用する信号。

C++

```
__property Signal InputSignal;
```

C#

```
public Signal InputSignal;
```

Visual Basic

```
Public Property InputSignal() As Signal
```

MgcLimitSwitch.InvertResetBehaviour Property

trueの場合、定義されたResetBehaviorが反転されます。

C++

```
__property bool InvertResetBehaviour;
```

C#

```
public bool InvertResetBehaviour;
```

Visual Basic

```
Public Property InvertResetBehaviour() As bool
```

MgcLimitSwitch.Limit Property

制限値。OperationDirectionに依存し、下限バンド値を定義するためにも使用できます。

C++

```
__property double Limit;
```

C#

```
public double Limit;
```

Visual Basic

```
Public Property Limit() As double
```

MgcLimitSwitch.LimitSwitchNumber Property

リミットスイッチの番号。範囲は1～32です。

C++

```
__property int LimitSwitchNumber;
```

C#

```
public int LimitSwitchNumber;
```

Visual Basic

```
Public Property LimitSwitchNumber() As Integer
```

MgcLimitSwitch.OperatingDirection Property

リミットスイッチの動作方向。

C++

```
__property LimitSwitchOperatingDirection OperatingDirection;
```

C#

```
public LimitSwitchOperatingDirection OperatingDirection;
```

Visual Basic

```
Public Property OperatingDirection() As LimitSwitchOperatingDirection
```

MgcLimitSwitch.ResetBehaviorMask Property

リセット動作(ヒステリシス管理に使用できます)。すべてのデジタル入力とAND演算されるバイナリマスク。結果が>0 (true)の場合、このリミット スイッチはクリアされます。値の範囲: 0、1、2、4、8、16、...32768は、MGCの16個のデジタル入力に対応します。リセット動作は、InvertResetBehaviorで反転できます。デフォルト設定は0です。

C++

```
__property int ResetBehaviorMask;
```

C#

```
public int ResetBehaviorMask;
```

Visual Basic

```
Public Property ResetBehaviorMask() As Integer
```

Hbm.Api.Mgc.Channels Namespace

これは名前空間Hbm.Api.Mgc.Channelsです。

クラス

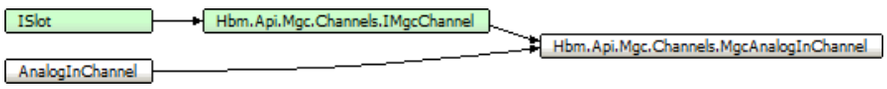
名前	説明
MgcAnalogInChannel	センサを接続するチャネルを表します。
MgcAnalogOutChannel	MGC固有のアナログ出力チャネルの実装。
MgcCanInChannel	MGC固有のCAN入力チャネルの実装。
MgcDigitalChannel	MGC固有のデジタルチャネルの実装。
MgcVirtualChannel	センサを接続しないチャネル(演算チャネルなど)を表します。

インタフェース

	名前	説明
	IMgcChannel	MgcChannelの追加プロパティのインタフェース。

MgcAnalogInChannel Class

センサを接続するチャネルを表します。



C++

```
class MgcAnalogInChannel : public AnalogInChannel, public IMgcChannel;
```

C#

```
public class MgcAnalogInChannel : AnalogInChannel, IMgcChannel;
```

Visual Basic

```
Public Class MgcAnalogInChannel
Inherits AnalogInChannel
Inherits IMgcChannel
```

ファイル

MgcAnalogInChannel.cs

MgcAnalogInChannel Members

次の表は、MgcAnalogInChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。
	TedsUsageMode	TEDS使用モードを取得または設定します。

MgcAnalogInChannel Methods

MgcAnalogInChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

MgcAnalogInChannel.GetUniqueID Method

チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。

MgcAnalogInChannel Properties

MgcAnalogInChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。
	TedsUsageMode	TEDS使用モードを取得または設定します。

MgcAnalogInChannel.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

MgcAnalogInChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

MgcAnalogInChannel.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

MgcAnalogInChannel.ConnectionBoardSerialNumber Property

チャネルが属する接続ボードのシリアル番号。

C++

```
__property string ConnectionBoardSerialNumber;
```

C#

```
public string ConnectionBoardSerialNumber;
```

Visual Basic

```
Public Property ConnectionBoardSerialNumber() As string
```

MgcAnalogInChannel.ConnectionBoardType Property

チャネルが属する接続ボードの種類。

C++

```
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
public ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
Public Property ConnectionBoardType() As ConnectionBoardType
```

MgcAnalogInChannel.Position Property

チャネルが属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcAnalogInChannel.Slot Property

チャネルが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcAnalogInChannel.TedsUsageMode Property

TEDS使用モードを取得または設定します。

C++

```
__property TedsUsageMode TedsUsageMode;
```

C#

```
public override TedsUsageMode TedsUsageMode;
```

Visual Basic

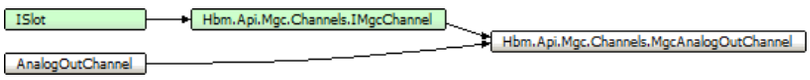
```
Public override Property TedsUsageMode() As TedsUsageMode
```

備考

TedsUsageMode.IgnoreおよびTedsUsageMode.UseManuallyのみをサポートしています。

MgcAnalogOutChannel Class

MGC固有のアナログ出力チャネルの実装。



C++

```
class MgcAnalogOutChannel : public AnalogOutChannel, public IMgcChannel;
```

C#

```
public class MgcAnalogOutChannel : AnalogOutChannel, IMgcChannel;
```

Visual Basic

```
Public Class MgcAnalogOutChannel
Inherits AnalogOutChannel
Inherits IMgcChannel
```

ファイル

MgcAnalogOutChannel.cs

MgcAnalogOutChannel Members

次の表は、MgcAnalogOutChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

MgcAnalogOutChannel Methods

MgcAnalogOutChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

MgcAnalogOutChannel.GetUniqueID Method

チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。

MgcAnalogOutChannel Properties

MgcAnalogOutChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

MgcAnalogOutChannel.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

MgcAnalogOutChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

MgcAnalogOutChannel.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

MgcAnalogOutChannel.ConnectionBoardSerialNumber Property

チャネルが属する接続ボードのシリアル番号。

C++

```
__property string ConnectionBoardSerialNumber;
```

C#

```
public string ConnectionBoardSerialNumber;
```

Visual Basic

```
Public Property ConnectionBoardSerialNumber() As string
```

MgcAnalogOutChannel.ConnectionBoardType Property

チャネルが属する接続ボードの種類。

C++

```
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
public ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
Public Property ConnectionBoardType() As ConnectionBoardType
```

MgcAnalogOutChannel.Position Property

チャネルが属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcAnalogOutChannel.Slot Property

チャネルが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

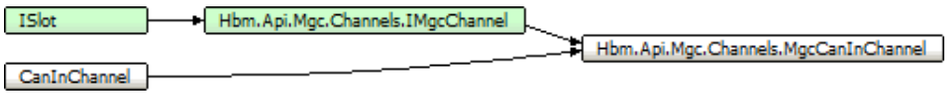
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcCanInChannel Class

MGC固有のCAN入力チャネルの実装。



C++

```
class MgcCanInChannel : public CanInChannel, public IMgcChannel;
```

C#

```
public class MgcCanInChannel : CanInChannel, IMgcChannel;
```

Visual Basic

```
Public Class MgcCanInChannel
Inherits CanInChannel
Inherits IMgcChannel
```

ファイル

MgcCanInChannel.cs

MgcCanInChannel Members

次の表は、MgcCanInChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

MgcCanInChannel Methods

MgcCanInChannelクラスのメソッドの一覧です。

Public Methods



	名前	説明
	GetUniqueID	チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

MgcCanInChannel.GetUniqueID Method

チャネルが属するデバイス内でチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。

MgcCanInChannel Properties

MgcCanInChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

MgcCanInChannel.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

MgcCanInChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

MgcCanInChannel.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

MgcCanInChannel.ConnectionBoardSerialNumber Property

チャネルが属する接続ボードのシリアル番号。

C++

```
__property string ConnectionBoardSerialNumber;
```

C#

```
public string ConnectionBoardSerialNumber;
```

Visual Basic

```
Public Property ConnectionBoardSerialNumber() As string
```

MgcCanInChannel.ConnectionBoardType Property

チャネルが属する接続ボードの種類。

C++

```
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
public ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
Public Property ConnectionBoardType() As ConnectionBoardType
```

MgcCanInChannel.Position Property

チャネルが属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcCanInChannel.Slot Property

チャネルが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcDigitalChannel Class

MGC固有のデジタルチャネルの実装。



C++

```
class MgcDigitalChannel : public DigitalChannel, public IMgcChannel;
```

C#

```
public class MgcDigitalChannel : DigitalChannel, IMgcChannel;
```

Visual Basic

```
Public Class MgcDigitalChannel
Inherits DigitalChannel
Inherits IMgcChannel
```

ファイル

MgcDigitalChannel.cs

MgcDigitalChannel Members

次の表は、MgcDigitalChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。
	Pin	このチャネルにマップされているデジタルピン。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

MgcDigitalChannel Methods

MgcDigitalChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

MgcDigitalChannel.GetUniqueID Method

デバイス内のチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。

MgcDigitalChannel Properties

MgcDigitalChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。
	Pin	このチャネルにマップされているデジタルピン。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

MgcDigitalChannel.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

MgcDigitalChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

MgcDigitalChannel.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

MgcDigitalChannel.ConnectionBoardSerialNumber Property

チャネルが属する接続ボードのシリアル番号。

C++

```
__property string ConnectionBoardSerialNumber;
```

C#

```
public string ConnectionBoardSerialNumber;
```

Visual Basic

```
Public Property ConnectionBoardSerialNumber() As string
```

MgcDigitalChannel.ConnectionBoardType Property

チャネルが属する接続ボードの種類。

C++

```
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
public ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
Public Property ConnectionBoardType() As ConnectionBoardType
```

MgcDigitalChannel.Pin Property

このチャネルにマップされているデジタルピン。

C++

```
[JsonProperty]
__property int Pin;
```

C#

```
[JsonProperty]
public int Pin;
```

Visual Basic

```
<JsonProperty>
Public Property Pin() As Integer
```

MgcDigitalChannel.Position Property

チャネルが属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcDigitalChannel.Slot Property
チャネルが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

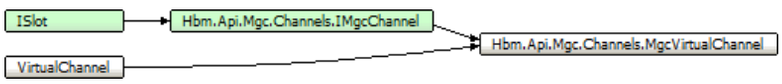
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcVirtualChannel Class

センサを接続しないチャネル(演算チャネルなど)を表します。



C++

```
class MgcVirtualChannel : public VirtualChannel, public IMgcChannel;
```

C#

```
public class MgcVirtualChannel : VirtualChannel, IMgcChannel;
```

Visual Basic

```
Public Class MgcVirtualChannel
Inherits VirtualChannel
Inherits IMgcChannel
```

ファイル

MgcVirtualChannel.cs

MgcVirtualChannel Members

次の表は、MgcVirtualChannelによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

MgcVirtualChannel Methods

PmxVirtualChannelクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のチャネルを識別する一意の文字列を返します。

MgcVirtualChannel.GetUniqueID Method

デバイス内のチャネルを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のチャネル識別子。

備考

この情報はスキャンま

MgcVirtualChannel Properties

MgcVirtualChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	チャネルが属するアンプのシリアル番号。
	AmplifierType	チャネルが属するアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。
	Position	チャネルが属するスロット内の位置。
	Slot	チャネルが属するデバイスのスロット番号。

MgcVirtualChannel.AmplifierSerialNumber Property

チャネルが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

MgcVirtualChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

MgcVirtualChannel.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

MgcVirtualChannel.ConnectionBoardSerialNumber Property

チャネルが属する接続ボードのシリアル番号。

C++

```
__property string ConnectionBoardSerialNumber;
```

C#

```
public string ConnectionBoardSerialNumber;
```

Visual Basic

```
Public Property ConnectionBoardSerialNumber() As string
```

MgcVirtualChannel.ConnectionBoardType Property

チャネルが属する接続ボードの種類。

C++

```
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
public ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
Public Property ConnectionBoardType() As ConnectionBoardType
```

MgcVirtualChannel.Position Property

チャネルが属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcVirtualChannel.Slot Property

チャネルが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

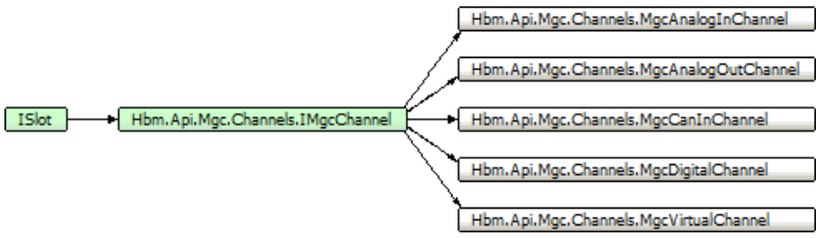
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

IMgcChannel Interface

MgcChannelの追加プロパティのインタフェース。



C++

```
__interface IMgcChannel : public ISlot;
```

C#

```
public interface IMgcChannel : ISlot;
```

Visual Basic

```
Public Interface IMgcChannel
Inherits ISlot
```

ファイル

IPmxChannel.cs

IMgcChannel Members

次の表は、IMgcChannelによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	AmplifierType	チャネルが属するアンプの種類。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。

IMgcChannel Properties

IMgcChannelクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierType	チャネルが属するアンプの種類。
	ConnectionBoardSerialNumber	チャネルが属する接続ボードのシリアル番号。
	ConnectionBoardType	チャネルが属する接続ボードの種類。

IMgcChannel.AmplifierType Property

チャネルが属するアンプの種類。

C++

```
[JsonProperty]
__property AmplifierType AmplifierType;
```

C#

```
[JsonProperty]
AmplifierType AmplifierType;
```

Visual Basic

```
<JsonProperty>
Property AmplifierType() As AmplifierType
```

IMgcChannel.ConnectionBoardSerialNumber Property

チャネルが属する接続ボードのシリアル番号。

C++

```
[JsonProperty]
__property string ConnectionBoardSerialNumber;
```

C#

```
[JsonProperty]
string ConnectionBoardSerialNumber;
```

Visual Basic

```
<JsonProperty>
Property ConnectionBoardSerialNumber() As string
```

IMgcChannel.ConnectionBoardType Property

チャネルが属する接続ボードの種類。

C++

```
[JsonProperty]
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
[JsonProperty]
ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
<JsonProperty>
Property ConnectionBoardType() As ConnectionBoardType
```

Hbm.Api.Mgc.Connectors Namespace

これは名前空間Hbm.Api.Mgc.Connectorsです。

クラス

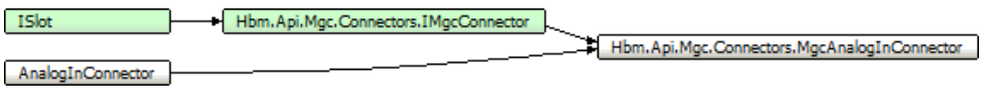
名前	説明
MgcAnalogInConnector	センサを接続するコネクタを表します。
MgcAnalogOutConnector	MGC固有のアナログ出力コネクタの実装。
MgcCanConnector	MGC固有のCANコネクタの実装。
MgcDigitalConnector	MGC固有のデジタルコネクタの実装。
MgcVirtualConnector	センサを接続しないコネクタ(演算チャネルを格納するコネクタなど)を表します。

インタフェース

	名前	説明
	IMgcConnector	MgcConnectorの追加プロパティのインタフェース。

MgcAnalogInConnector Class

センサを接続するコネクタを表します。



C++

```
class MgcAnalogInConnector : public AnalogInConnector, public IMgcConnector;
```

C#

```
public class MgcAnalogInConnector : AnalogInConnector, IMgcConnector;
```

Visual Basic

```
Public Class MgcAnalogInConnector
Inherits AnalogInConnector
Inherits IMgcConnector
```

ファイル

MgcAnalogInConnector.cs

MgcAnalogInConnector Members

次の表は、MgcAnalogInConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	コネクタが属するアンプのシリアル番号。
	AmplifierType	Slotで指定したスロットに差し込まれるアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。
	Position	アンプ側のコネクタの位置。
	Slot	コネクタが属するデバイスのスロット番号。

MgcAnalogInConnector Methods

MgcAnalogInConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のコネクタを識別する一意の文字列を返します。

MgcAnalogInConnector.GetUniqueID Method

デバイス内のコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタ識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。一意のコネクタIDが設定されていない場合、メソッド実装はcref="System.InvalidOperationException">をスローする必要があります。

例外

例外	説明
System.InvalidOperationException	値が含まれていない場合はスローされます。

MgcAnalogInConnector Properties

MgcAnalogInConnectorクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	コネクタが属するアンプのシリアル番号。
	AmplifierType	Slotで指定したスロットに差し込まれるアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。
	Position	アンプ側のコネクタの位置。
	Slot	コネクタが属するデバイスのスロット番号。

MgcAnalogInConnector.AmplifierSerialNumber Property

コネクタが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

MgcAnalogInConnector.AmplifierType Property

Slotで指定したスロットに差し込まれるアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

MgcAnalogInConnector.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

MgcAnalogInConnector.ConnectionBoardSerialNumber Property

コネクタが属する接続ボードのシリアル番号。

C++

```
__property string ConnectionBoardSerialNumber;
```

C#

```
public string ConnectionBoardSerialNumber;
```

Visual Basic

```
Public Property ConnectionBoardSerialNumber() As string
```

MgcAnalogInConnector.ConnectionBoardType Property

コネクタが属する接続ボードの種類。

C++

```
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
public ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
Public Property ConnectionBoardType() As ConnectionBoardType
```

MgcAnalogInConnector.Position Property

アンプ側のコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcAnalogInConnector.Slot Property

コネクタが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

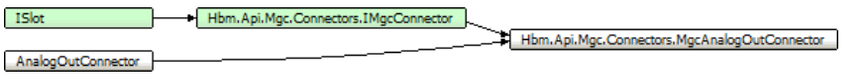
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcAnalogOutConnector Class

MGC固有のアナログ出力コネクタの実装。



C++

```
class MgcAnalogOutConnector : public AnalogOutConnector, public IMgcConnector;
```

C#

```
public class MgcAnalogOutConnector : AnalogOutConnector, IMgcConnector;
```

Visual Basic

```
Public Class MgcAnalogOutConnector
Inherits AnalogOutConnector
Inherits IMgcConnector
```

ファイル

MgcAnalogOutConnector.cs

MgcAnalogOutConnector Members

次の表は、MgcAnalogOutConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	コネクタが属するアンプのシリアル番号。
	AmplifierType	Slotで指定したスロットに差し込まれるアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。
	Position	アンプ側のコネクタの位置。
	Slot	コネクタが属するデバイスのスロット番号。

MgcAnalogOutConnector Methods

MgcAnalogOutConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のコネクタを識別する一意の文字列を返します。

MgcAnalogOutConnector.GetUniqueID Method
デバイス内のコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値
デバイス上の一意のコネクタ識別子。

備考
この情報はスキャンまたは接続中に取得する必要があります。一意のコネクタIDが設定されていない場合、メソッド実装はcref="System.InvalidOperationException">をスローする必要があります。

例外

例外	説明
System.InvalidOperationException	値が含まれていない場合はスローされます。

MgcAnalogOutConnector Properties

MgcAnalogOutConnectorクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	コネクタが属するアンプのシリアル番号。
	AmplifierType	Slotで指定したスロットに差し込まれるアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。
	Position	アンプ側のコネクタの位置。
	Slot	コネクタが属するデバイスのスロット番号。

MgcAnalogOutConnector.AmplifierSerialNumber Property

コネクタが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

MgcAnalogOutConnector.AmplifierType Property

Slotで指定したスロットに差し込まれるアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

MgcAnalogOutConnector.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

MgcAnalogOutConnector.ConnectionBoardSerialNumber Property

コネクタが属する接続ボードのシリアル番号。

C++

```
__property string ConnectionBoardSerialNumber;
```

C#

```
public string ConnectionBoardSerialNumber;
```

Visual Basic

```
Public Property ConnectionBoardSerialNumber() As string
```

MgcAnalogOutConnector.ConnectionBoardType Property

コネクタが属する接続ボードの種類。

C++

```
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
public ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
Public Property ConnectionBoardType() As ConnectionBoardType
```

MgcAnalogOutConnector.Position Property

アンプ側のコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcAnalogOutConnector.Slot Property

コネクタが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

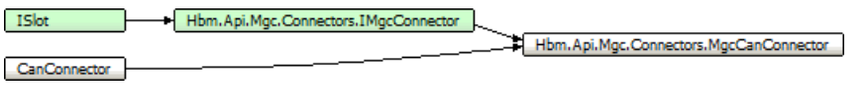
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcCanConnector Class

MGC固有のCANコネクタの実装。



C++

```
class MgcCanConnector : public CanConnector, public IMgcConnector;
```

C#

```
public class MgcCanConnector : CanConnector, IMgcConnector;
```

Visual Basic

```
Public Class MgcCanConnector
Inherits CanConnector
Inherits IMgcConnector
```

ファイル

MgcCanConnector.cs

MgcCanConnector Members

次の表は、MgcCanConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	コネクタが属するアンプのシリアル番号。
	AmplifierType	Slotで指定したスロットに差し込まれるアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。
	Position	MGCは、2つの異なるボーレートを持CANポートをサポートします。位置は1または2です。
	Slot	コネクタが属するデバイスのスロット番号。

MgcCanConnector Methods

MgcCanConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のコネクタを識別する一意の文字列を返します。

MgcCanConnector.GetUniqueID Method

デバイス内のコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタ識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。一意のコネクタIDが設定されていない場合、メソッド実装はcref="System.InvalidOperationException">をスローする必要があります。

例外

例外	説明
System.InvalidOperationException	値が含まれていない場合はスローされます。

MgcCanConnector Properties

MgcCanConnectorクラスのプロパティの一覧です。

Public Properties

	名前	説明
	AmplifierSerialNumber	コネクタが属するアンプのシリアル番号。
	AmplifierType	Slotで指定したスロットに差し込まれるアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。
	Position	MGCは、2つの異なるボーレートを持CANポートをサポートします。位置は1または2です。
	Slot	コネクタが属するデバイスのスロット番号。

MgcCanConnector.AmplifierSerialNumber Property

コネクタが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

MgcCanConnector.AmplifierType Property

Slotで指定したスロットに差し込まれるアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

MgcCanConnector.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

MgcCanConnector.ConnectionBoardSerialNumber Property

コネクタが属する接続ボードのシリアル番号。

C++

```
__property string ConnectionBoardSerialNumber;
```

C#

```
public string ConnectionBoardSerialNumber;
```

Visual Basic

```
Public Property ConnectionBoardSerialNumber() As string
```

MgcCanConnector.ConnectionBoardType Property

コネクタが属する接続ボードの種類。

C++

```
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
public ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
Public Property ConnectionBoardType() As ConnectionBoardType
```

MgcCanConnector.Position Property

MGCは、2つの異なるボーレートを持CANポートをサポートします。位置は1または2です。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcCanConnector.Slot Property

コネクタが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

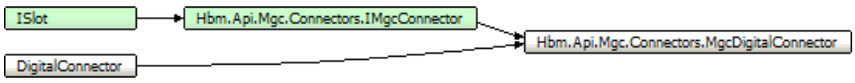
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcDigitalConnector Class

MGC固有のデジタルコネクタの実装。



C++

```
class MgcDigitalConnector : public DigitalConnector, public IMgcConnector;
```

C#

```
public class MgcDigitalConnector : DigitalConnector, IMgcConnector;
```

Visual Basic

```
Public Class MgcDigitalConnector
Inherits DigitalConnector
Inherits IMgcConnector
```

ファイル

MgcDigitalConnector.cs

MgcDigitalConnector Members

次の表は、MgcDigitalConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	コネクタが属するアンプのシリアル番号。
	AmplifierType	Slotで指定したスロットに差し込まれるアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。
	Pin	このコネクタにマップされているデジタルピン。
	Position	アンプ上のコネクタの位置。
	Slot	コネクタが属するデバイスのスロット番号。

MgcDigitalConnector Methods

MgcDigitalConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のコネクタを識別する一意の文字列を返します。

MgcDigitalConnector.GetUniqueID Method

デバイス内のコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタ識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。一意のコネクタIDが設定されていない場合、メソッド実装はcref="System.InvalidOperationException">をスローする必要があります。

例外

例外	説明
System.InvalidOperationException	値が含まれていない場合はスローされます。

現在位置: Supported devices -> Hbm.Api.Mgc Namespace -> Hbm.Api.Mgc.Connectors Namespace -> MgcDigitalConnector Methods -> MgcDigitalConnector Members

MgcDigitalConnector Properties

MgcDigitalConnectorクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	コネクタが属するアンプのシリアル番号。
	AmplifierType	Slotで指定したスロットに差し込まれるアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。
	Pin	このコネクタにマップされているデジタルピン。
	Position	アンプ上のコネクタの位置。
	Slot	コネクタが属するデバイスのスロット番号。

MgcDigitalConnector.AmplifierSerialNumber Property

コネクタが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

MgcDigitalConnector.AmplifierType Property

Slotで指定したスロットに差し込まれるアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

MgcDigitalConnector.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

MgcDigitalConnector.ConnectionBoardSerialNumber Property

コネクタが属する接続ボードのシリアル番号。

C++

```
__property string ConnectionBoardSerialNumber;
```

C#

```
public string ConnectionBoardSerialNumber;
```

Visual Basic

```
Public Property ConnectionBoardSerialNumber() As string
```

MgcDigitalConnector.ConnectionBoardType Property

コネクタが属する接続ボードの種類。

C++

```
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
public ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
Public Property ConnectionBoardType() As ConnectionBoardType
```

MgcDigitalConnector.Pin Property

このコネクタにマップされているデジタルピン。

C++

```
[JsonProperty]
__property int Pin;
```

C#

```
[JsonProperty]
public int Pin;
```

Visual Basic

```
<JsonProperty>
Public Property Pin() As Integer
```

MgcDigitalConnector.Position Property

アンプ上のコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcDigitalConnector.Slot Property

コネクタが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

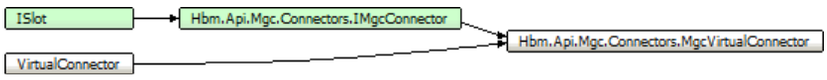
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcVirtualConnector Class

センサを接続しないコネクタ(演算チャネルを格納するコネクタなど)を表します。



C++

```
class MgcVirtualConnector : public VirtualConnector, public IMgcConnector;
```

C#

```
public class MgcVirtualConnector : VirtualConnector, IMgcConnector;
```

Visual Basic

```
Public Class MgcVirtualConnector
Inherits VirtualConnector
Inherits IMgcConnector
```

ファイル

MgcVirtualConnector.cs

MgcVirtualConnector Members

次の表は、MgcVirtualConnectorによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のコネクタを識別する一意の文字列を返します。

Public Properties

	名前	説明
	AmplifierSerialNumber	コネクタが属するアンプのシリアル番号。
	AmplifierType	Slotで指定したスロットに差し込まれるアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。
	Position	アンプ上のコネクタの位置。
	Slot	コネクタが属するデバイスのスロット番号。

MgcVirtualConnector Methods

MgcVirtualConnectorクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	デバイス内のコネクタを識別する一意の文字列を返します。

MgcVirtualConnector.GetUniqueID Method

デバイス内のコネクタを識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意のコネクタ識別子。

備考

この情報はスキャンまたは接続中に取得する必要があります。一意のコネクタIDが設定されていない場合、メソッド実装はcref="System.InvalidOperationException">をスローする必要があります。

例外

例外	説明
System.InvalidOperationException	値が含まれていない場合はスローされます。

MgcVirtualConnector Properties

MgcVirtualConnectorクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierSerialNumber	コネクタが属するアンプのシリアル番号。
	AmplifierType	Slotで指定したスロットに差し込まれるアンプの種類。
R	AmplifierTypeName	アンプの種類名。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。
	Position	アンプ上のコネクタの位置。
	Slot	コネクタが属するデバイスのスロット番号。

MgcVirtualConnector.AmplifierSerialNumber Property

コネクタが属するアンプのシリアル番号。

C++

```
__property string AmplifierSerialNumber;
```

C#

```
public string AmplifierSerialNumber;
```

Visual Basic

```
Public Property AmplifierSerialNumber() As string
```

MgcVirtualConnector.AmplifierType Property

Slotで指定したスロットに差し込まれるアンプの種類。

C++

```
__property AmplifierType AmplifierType;
```

C#

```
public AmplifierType AmplifierType;
```

Visual Basic

```
Public Property AmplifierType() As AmplifierType
```

MgcVirtualConnector.AmplifierTypeName Property

アンプの種類名。

C++

```
__property string AmplifierTypeName;
```

C#

```
public string AmplifierTypeName;
```

Visual Basic

```
Public Property AmplifierTypeName() As string
```

MgcVirtualConnector.ConnectionBoardSerialNumber Property

コネクタが属する接続ボードのシリアル番号。

C++

```
__property string ConnectionBoardSerialNumber;
```

C#

```
public string ConnectionBoardSerialNumber;
```

Visual Basic

```
Public Property ConnectionBoardSerialNumber() As string
```

MgcVirtualConnector.ConnectionBoardType Property

コネクタが属する接続ボードの種類。

C++

```
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
public ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
Public Property ConnectionBoardType() As ConnectionBoardType
```

MgcVirtualConnector.Position Property

アンプ上のコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcVirtualConnector.Slot Property

コネクタが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

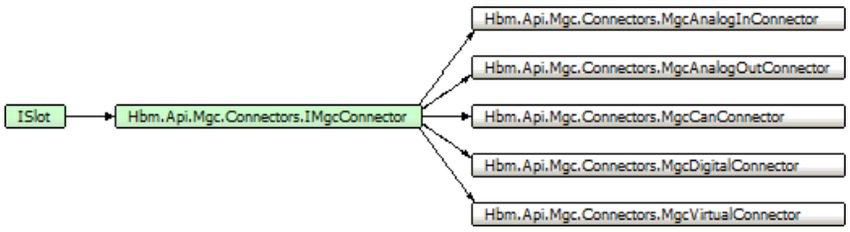
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

IMgcConnector Interface

MgcConnectorの追加プロパティのインタフェース。



C++

```
__interface IMgcConnector : public ISlot;
```

C#

```
public interface IMgcConnector : ISlot;
```

Visual Basic

```
Public Interface IMgcConnector
Inherits ISlot
```

ファイル

IMgcConnector.cs

IMgcConnector Members

次の表は、IMgcConnectorによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	AmplifierType	スロットに接続するアンプの種類。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。

IMgcConnector Properties

IMgcConnectorクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AmplifierType	スロットに接続するアンプの種類。
	ConnectionBoardSerialNumber	コネクタが属する接続ボードのシリアル番号。
	ConnectionBoardType	コネクタが属する接続ボードの種類。

IMgcConnector.AmplifierType Property

スロットに接続するアンプの種類。

C++

```
[JsonProperty]
__property AmplifierType AmplifierType;
```

C#

```
[JsonProperty]
AmplifierType AmplifierType;
```

Visual Basic

```
<JsonProperty>
Property AmplifierType() As AmplifierType
```

IMgcConnector.ConnectionBoardSerialNumber Property

コネクタが属する接続ボードのシリアル番号。

C++

```
[JsonProperty]
__property string ConnectionBoardSerialNumber;
```

C#

```
[JsonProperty]
string ConnectionBoardSerialNumber;
```

Visual Basic

```
<JsonProperty>
Property ConnectionBoardSerialNumber() As string
```

IMgcConnector.ConnectionBoardType Property

コネクタが属する接続ボードの種類。

C++

```
[JsonProperty]
__property ConnectionBoardType ConnectionBoardType;
```

C#

```
[JsonProperty]
ConnectionBoardType ConnectionBoardType;
```

Visual Basic

```
<JsonProperty>
Property ConnectionBoardType() As ConnectionBoardType
```

Hbm.Api.Mgc.Enums Namespace

これは名前空間Hbm.Api.Mgc.Enumsです。

列挙型

	名前	説明
	AmplifierType	MGCアンプの種類。
	AnalogOutBesselFrequency	MGCがサポートするアナログ出力ベッセルフィルタ周波数。
	AnalogOutButterworthFrequency	MGCがサポートするアナログ出力バターワースフィルタ周波。
	AnalogOutMode	アナログ出力チャネルのモード。
	BaudRate	MGCがサポートするCAN busボー レート。
	ByteOrder	MGC CAN信号のバイトオーダー。
	ConnectionBoardType	MGC接続ボードの種類。
	DataFormat	MGC CAN信号のデータ形式。
	DeviceSynchronization	CP52のデバイス同期の種類。
	DigitalOutMode	デジタル信号のモード(コネクタが出力方向のデジタル信号にのみ有効)。
	DischargeTimeType	放電時間の種類。
	ExcitationAmplitude	MGCがサポートする励起振幅。
	FilterCharacteristic	MGCがサポートするフィルタ特性。
	FilterFrequency	MGCがサポートするフィルタ周波数(Hz)。
	FrameFormat	MGC CANセンサのフレームフォーマット。
	InputType	入力の種類。
	LimitSwitchOperatingDirection	MGCリミットスイッチの動作方向。
	MeasurementSignalType	測定信号の種類。MGCを接続すると、風袋が0.0に設定されなくなることに注意してください! したがって、信号の種類によって測定値が異なる場合があります(これを使用して、たとえばグロス信号を要求するML77によって発生する可能性のあるリンク リソースエラーを解決します...)。
	MgcDeviceStatus	Deviceのステータス。
	MgcSensorType	センサの種類。
	ModeSignalByteOrder	MGC CAN信号のモード信号のバイトオーダー。
	SampleRate	周波数(Hz)。
	SignalType	MGCがサポートする信号の種類。
	TimeChannelType	時間チャネルの形式。注意!!! 現在サポートされている種類はNtp64BitTimeのみです。他の種類はすべて網羅性のためだけのものです...
	TransducerSensitivity	MGCがサポートするトランスデューサの感度。
	TriggerOperationMode	測定開始時のMGCの動作を定義するトリガ操作モード。
	Wiring	MGCの配線の種類。

Hbm.Api.Mgc.Enums.AmplifierType Enumeration

MGCアンプの種類。

C++

```
enum AmplifierType {
  Unknown = -1,
  ML30 = 5000,
  ML50 = 5001,
  ML01 = 5002,
  ML55 = 5003,
  ML60 = 5004,
  ML35 = 5005,
  ML10 = 5006,
  ML51 = 5007,
  ML55S6 = 5008,
  ML38 = 5009,
  ML85_3 = 5010,
  ML85_1 = 5011,
  ML70 = 5012,
  ML71 = 5013,
  ML77 = 5014,
  ML801 = 5015,
  ML78 = 5016,
  ML71S6 = 5017,
  ML460 = 5018,
  ML74 = 5019,
  ML455 = 5020
};
```

C#

```
public enum AmplifierType {
  Unknown = -1,
  ML30 = 5000,
  ML50 = 5001,
  ML01 = 5002,
  ML55 = 5003,
  ML60 = 5004,
  ML35 = 5005,
  ML10 = 5006,
  ML51 = 5007,
  ML55S6 = 5008,
  ML38 = 5009,
  ML85_3 = 5010,
  ML85_1 = 5011,
  ML70 = 5012,
  ML71 = 5013,
  ML77 = 5014,
  ML801 = 5015,
  ML78 = 5016,
  ML71S6 = 5017,
  ML460 = 5018,
  ML74 = 5019,
  ML455 = 5020
}
```

Visual Basic

```
Public Enum AmplifierType
  Unknown = -1
  ML30 = 5000
  ML50 = 5001
  ML01 = 5002
  ML55 = 5003
  ML60 = 5004
  ML35 = 5005
  ML10 = 5006
  ML51 = 5007
  ML55S6 = 5008
  ML38 = 5009
  ML85_3 = 5010
  ML85_1 = 5011
  ML70 = 5012
  ML71 = 5013
  ML77 = 5014
  ML801 = 5015
  ML78 = 5016
  ML71S6 = 5017
  ML460 = 5018
  ML74 = 5019
  ML455 = 5020
End Enum
```

ファイル

AmplifierType.cs

メンバー

メンバー	説明
Unknown	不明なアンプ。
ML30	ML30アンプ。
ML50	ML50アンプ。
ML01	ML01アンプ。
ML55	ML55アンプ。
ML60	ML60アンプ。
ML35	ML35アンプ。
ML10	ML10アンプ。
ML51	ML51アンプ。
ML55S6	ML55S6アンプ。
ML38	ML38アンプ。
ML85_3	ML85_3アンプ。
ML85_1	ML85_1アンプ。
ML70	ML70アンプ。
ML71	ML71アンプ。
ML77	ML77アンプ。
ML801	ML801アンプ。
ML78	ML78アンプ。
ML71S6	ML71S6アンプ。
ML460	ML460アンプ。
ML74	ML74アンプ。

備考

列挙項目の値は、デバイスのクエリ言語によって返されるアンプの種類番号と同じです。

Hbm.Api.Mgc.Enums.AnalogOutBesselFrequency Enumeration

MGCがサポートするアナログ出力ベッセルフィルタ周波数。

C++

```
[AdditionalValueType(typeof(decimal))]
enum AnalogOutBesselFrequency : public int {
  Hz0_05 = 908,
  Hz0_1 = 914,
  Hz0_2 = 917,
  Hz0_5 = 921,
  Hz1_25 = 929,
  Hz2_5 = 932,
  Hz5_0 = 935,
  Hz10 = 941,
  Hz20_0 = 945,
  Hz40_0 = 948,
  Hz100 = 955,
  Hz200 = 958,
  Hz400 = 961,
  Hz900 = 968
};
```

C#

```
[AdditionalValueType(typeof(decimal))]
internal enum AnalogOutBesselFrequency : int {
  Hz0_05 = 908,
  Hz0_1 = 914,
  Hz0_2 = 917,
  Hz0_5 = 921,
  Hz1_25 = 929,
  Hz2_5 = 932,
  Hz5_0 = 935,
  Hz10 = 941,
  Hz20_0 = 945,
  Hz40_0 = 948,
  Hz100 = 955,
  Hz200 = 958,
  Hz400 = 961,
  Hz900 = 968
}
```

Visual Basic

```
<AdditionalValueType(typeof(decimal))>
Friend Enum AnalogOutBesselFrequency
Inherits int
  Hz0_05 = 908
  Hz0_1 = 914
  Hz0_2 = 917
  Hz0_5 = 921
  Hz1_25 = 929
  Hz2_5 = 932
  Hz5_0 = 935
  Hz10 = 941
  Hz20_0 = 945
  Hz40_0 = 948
  Hz100 = 955
  Hz200 = 958
  Hz400 = 961
  Hz900 = 968
End Enum
```

ファイル

AnalogOutBesselFrequency.cs

Hbm.Api.Mgc.Enums.AnalogOutButterworthFrequency Enumeration

MGCがサポートするアナログ出力バターワースフィルタ周波。

C++

```
[AdditionalValueType(typeof(decimal))]
enum AnalogOutButterworthFrequency : public int {
  Hz10 = 941,
  Hz20_0 = 945,
  Hz40_0 = 948,
  Hz80 = 953,
  Hz250 = 959,
  Hz500 = 962,
  Hz1000 = 969,
  Hz1500 = 972
};
```

C#

```
[AdditionalValueType(typeof(decimal))]
internal enum AnalogOutButterworthFrequency : int {
  Hz10 = 941,
  Hz20_0 = 945,
  Hz40_0 = 948,
  Hz80 = 953,
  Hz250 = 959,
  Hz500 = 962,
  Hz1000 = 969,
  Hz1500 = 972
}
```

Visual Basic

```
<AdditionalValueType(typeof(decimal))>
Friend Enum AnalogOutButterworthFrequency
Inherits int
  Hz10 = 941
  Hz20_0 = 945
  Hz40_0 = 948
  Hz80 = 953
  Hz250 = 959
  Hz500 = 962
  Hz1000 = 969
  Hz1500 = 972
End Enum
```

ファイル

AnalogOutButterworthFrequency.cs

Hbm.Api.Mgc.Enums.AnalogOutMode Enumeration

アナログ出力チャネルのモード。

C++

```
enum AnalogOutMode {
  Fixed0V = 1908,
  Adjustable = 1909,
  LinkedToSignal = 1910,
  LinkedToFunctionGenerator1 = 1911,
  LinkedToFunctionGenerator2 = 1912,
  LinkedToFunctionGenerator3 = 1913,
  LinkedToFunctionGenerator4 = 1914,
  LinkedToFunctionGenerator5 = 1915,
  LinkedToFunctionGenerator6 = 1916,
  LinkedToFunctionGenerator7 = 1917,
  LinkedToFunctionGenerator8 = 1918,
  LinkedToFunctionGenerator9 = 1919,
  LinkedToFunctionGenerator10 = 1920
};
```

C#

```
public enum AnalogOutMode {
  Fixed0V = 1908,
  Adjustable = 1909,
  LinkedToSignal = 1910,
  LinkedToFunctionGenerator1 = 1911,
  LinkedToFunctionGenerator2 = 1912,
  LinkedToFunctionGenerator3 = 1913,
  LinkedToFunctionGenerator4 = 1914,
  LinkedToFunctionGenerator5 = 1915,
  LinkedToFunctionGenerator6 = 1916,
  LinkedToFunctionGenerator7 = 1917,
  LinkedToFunctionGenerator8 = 1918,
  LinkedToFunctionGenerator9 = 1919,
  LinkedToFunctionGenerator10 = 1920
}
```

Visual Basic

```
Public Enum AnalogOutMode
  Fixed0V = 1908
  Adjustable = 1909
  LinkedToSignal = 1910
  LinkedToFunctionGenerator1 = 1911
  LinkedToFunctionGenerator2 = 1912
  LinkedToFunctionGenerator3 = 1913
  LinkedToFunctionGenerator4 = 1914
  LinkedToFunctionGenerator5 = 1915
  LinkedToFunctionGenerator6 = 1916
  LinkedToFunctionGenerator7 = 1917
  LinkedToFunctionGenerator8 = 1918
  LinkedToFunctionGenerator9 = 1919
  LinkedToFunctionGenerator10 = 1920
End Enum
```

ファイル

AnalogOutMode.cs

メンバー

メンバー	説明
Fixed0V	0V固定出力。
Adjustable	調整可能-10..10V出力。
LinkedToSignal	出力はデバイスの信号にリンク。
LinkedToFunctionGenerator1	出力はファンクションジェネレータ1によって生成。
LinkedToFunctionGenerator2	出力はファンクションジェネレータ2によって生成。
LinkedToFunctionGenerator3	出力はファンクションジェネレータ3によって生成。
LinkedToFunctionGenerator4	出力はファンクションジェネレータ4によって生成。
LinkedToFunctionGenerator5	出力はファンクションジェネレータ5によって生成。
LinkedToFunctionGenerator6	出力はファンクションジェネレータ6によって生成。
LinkedToFunctionGenerator7	出力はファンクションジェネレータ7によって生成。
LinkedToFunctionGenerator8	出力はファンクションジェネレータ8によって生成。
LinkedToFunctionGenerator9	出力はファンクションジェネレータ9によって生成。
LinkedToFunctionGenerator10	出力はファンクションジェネレータ10によって生成。

Hbm.Api.Mgc.Enums.BaudRate Enumeration

MGCがサポートするCAN busボー レート。

C++

```
[AdditionalValueType(typeof(int))]
enum BaudRate {
  Bd_10000 = 1409,
  Bd_20000 = 1411,
  Bd_50000 = 1413,
  Bd_100000 = 1427,
  Bd_125000 = 1417,
  Bd_250000 = 1419,
  Bd_500000 = 1421,
  Bd_1000000 = 1424
};
```

C#

```
[AdditionalValueType(typeof(int))]
internal enum BaudRate {
  Bd_10000 = 1409,
  Bd_20000 = 1411,
  Bd_50000 = 1413,
  Bd_100000 = 1427,
  Bd_125000 = 1417,
  Bd_250000 = 1419,
  Bd_500000 = 1421,
  Bd_1000000 = 1424
}
```

Visual Basic

```
<AdditionalValueType(typeof(int))>
Friend Enum BaudRate
  Bd_10000 = 1409
  Bd_20000 = 1411
  Bd_50000 = 1413
  Bd_100000 = 1427
  Bd_125000 = 1417
  Bd_250000 = 1419
  Bd_500000 = 1421
  Bd_1000000 = 1424
End Enum
```

ファイル

BaudRate.cs

備考

列挙項目の値は、デバイスのクエリ言語によって返される CAN busのボーレート番号と同じです。

Hbm.Api.Mgc.Enums.ByteOrder Enumeration

MGC CAN信号のバイトオーダー。

C++

```
enum ByteOrder {
  Intel = 0,
  Motorola = 1
};
```

C#

```
public enum ByteOrder {
  Intel = 0,
  Motorola = 1
}
```

Visual Basic

```
Public Enum ByteOrder
  Intel = 0
  Motorola = 1
End Enum
```

ファイル

ByteOrder.cs

メンバー

メンバー	説明
Intel	リトルエンディアン(最下位バイトは最下位のメモリ位置に格納されます)。
Motorola	ビッグエンディアン(最上位バイトは最下位のメモリ位置に格納されます)。

Hbm.Api.Mgc.Enums.ConnectionBoardType Enumeration

MGC接続ボードの種類。

C++

```
enum ConnectionBoardType {
  Unknown = -1,
  NoAP = 0,
  AP01 = 5500,
  AP0506 = 5501,
  AP08 = 5502,
  AP14 = 5503,
  AP18 = 5504,
  AP09 = 5505,
  AP801 = 5506,
  AP809 = 5507,
  AP835 = 5508,
  AP409 = 5509,
  AP4092 = 5510,
  AP810 = 5511,
  AP814 = 5512,
  AP71 = 5513,
  AP72 = 5514,
  AP75 = 5515,
  AP77 = 5516,
  AP78 = 5517,
  AP815 = 5518,
  AP401 = 5519,
  AP4012 = 5520,
  AP836 = 5521,
  AP460 = 5522,
  AP801S6 = 5523,
  AP814B = 5525,
  AP801S41 = 5526,
  AP418 = 5527,
  AP4182 = 5528,
  AP17 = 5529,
  AP01i = 5530,
  AP03i = 5531,
  AP11i = 5532,
  AP74 = 5533,
  AP18i = 5535,
  AP13i = 5536,
  AP455 = 5538,
  AP402 = 5539,
  AP836i = 5541,
  AP816 = 5542,
  APTEST = 5550,
  APEINMESS = 5551,
  APWRONG = 5555
};
```

C#

```
public enum ConnectionBoardType {
  Unknown = -1,
  NoAP = 0,
  AP01 = 5500,
  AP0506 = 5501,
  AP08 = 5502,
  AP14 = 5503,
  AP18 = 5504,
  AP09 = 5505,
  AP801 = 5506,
  AP809 = 5507,
  AP835 = 5508,
  AP409 = 5509,
  AP4092 = 5510,
  AP810 = 5511,
  AP814 = 5512,
  AP71 = 5513,
  AP72 = 5514,
  AP75 = 5515,
  AP77 = 5516,
  AP78 = 5517,
  AP815 = 5518,
  AP401 = 5519,
  AP4012 = 5520,
  AP836 = 5521,
  AP460 = 5522,
  AP801S6 = 5523,
  AP814B = 5525,
  AP801S41 = 5526,
  AP418 = 5527,
  AP4182 = 5528,
  AP17 = 5529,
  AP01i = 5530,
  AP03i = 5531,
  AP11i = 5532,
  AP74 = 5533,
  AP18i = 5535,
  AP13i = 5536,
  AP455 = 5538,
  AP402 = 5539,
  AP836i = 5541,
  AP816 = 5542,
  APTEST = 5550,
  APEINMESS = 5551,
  APWRONG = 5555
}
```

Visual Basic

```
Public Enum ConnectionBoardType
  Unknown = -1
  NoAP = 0
  AP01 = 5500
  AP0506 = 5501
  AP08 = 5502
  AP14 = 5503
  AP18 = 5504
  AP09 = 5505
  AP801 = 5506
  AP809 = 5507
  AP835 = 5508
  AP409 = 5509
  AP4092 = 5510
  AP810 = 5511
  AP814 = 5512
  AP71 = 5513
  AP72 = 5514
  AP75 = 5515
  AP77 = 5516
  AP78 = 5517
  AP815 = 5518
  AP401 = 5519
  AP4012 = 5520
  AP836 = 5521
  AP460 = 5522
  AP801S6 = 5523
  AP814B = 5525
  AP801S41 = 5526
  AP418 = 5527
  AP4182 = 5528
  AP17 = 5529
  AP01i = 5530
  AP03i = 5531
  AP11i = 5532
  AP74 = 5533
  AP18i = 5535
  AP13i = 5536
  AP455 = 5538
  AP402 = 5539
  AP836i = 5541
  AP816 = 5542
  APTEST = 5550
  APEINMESS = 5551
  APWRONG = 5555
End Enum
```

ファイル

ConnectionBoardType.cs

メンバー

メンバー	説明
Unknown	不明な接続ボード。
NoAP	接続ボードなし(ML78接続ボードの2つのアナログ出力のみ使用されます)。
AP01	AP01接続ボード。
AP0506	AP0506接続ボード。
AP08	AP08接続ボード。
AP14	AP14接続ボード。
AP18	AP18接続ボード。
AP09	AP09接続ボード。
AP801	AP801接続ボード。
AP809	AP809接続ボード。
AP835	AP835接続ボード。
AP409	AP409接続ボード。
AP4092	AP4092接続ボード。
AP810	AP810接続ボード。
AP814	AP814接続ボード。
AP71	AP71接続ボード。
AP72	AP72接続ボード。
AP75	AP75接続ボード。
AP77	AP77接続ボード。
AP78	AP78接続ボード。
AP815	AP815接続ボード。
AP401	AP401接続ボード。
AP4012	AP4012接続ボード。
AP836	AP836接続ボード。
AP460	AP460接続ボード。
AP801S6	AP801S6接続ボード。
AP814B	AP814またはAP815。
AP801S41	AP801(0.1V)。
AP418	AP418: ピエゾ。
AP4182	2xAP418: ピエゾ。
AP17	AP17接続ボード。
AP01i	AP01i接続ボード。
AP03i	AP03i接続ボード。
AP11i	AP11i接続ボード。
AP74	AP74接続ボード。
AP18i	AP18i接続ボード。
AP13i	AP13i接続ボード。
AP455	AP455接続ボード。
AP402	AP402接続ボード。
AP836i	AP836i接続ボード。
AP816	AP816接続ボード。
APTEST	テスト接続ボード。
APEINMESS	校正接続ボード。
APWRONG	接続ボードが間違っています(アンプと接続ボードの組み合わせが無効です)。

備考

列挙項目の値は、デバイスのクエリ言語によって返される 接続ボードの種類番号と同じです。

Hbm.Api.Mgc.Enums.DataFormat Enumeration

MGC CAN信号のデータ形式。

C++

```
enum DataFormat {
  SignedInt = 1,
  UnsignedInt = 2,
  Float = 4
};
```

C#

```
public enum DataFormat {
  SignedInt = 1,
  UnsignedInt = 2,
  Float = 4
}
```

Visual Basic

```
Public Enum DataFormat
  SignedInt = 1
  UnsignedInt = 2
  Float = 4
End Enum
```

ファイル

DataFormat.cs

メンバー

メンバー	説明
SignedInt	符号付き整数。
UnsignedInt	符号なし整数。
Float	浮動小数点数。

Hbm.Api.Mgc.Enums.DeviceSynchronization Enumeration

CP52のデバイス同期の種類。

C++

```
enum DeviceSynchronization {
  NotSynchrnonized = 6700,
  CableSynchronized = 6701,
  PtpSynchonized = 6702,
  IrigSynchronized = 6703,
  NtpSynchronized = 6704
};
```

C#

```
public enum DeviceSynchronization {
  NotSynchrnonized = 6700,
  CableSynchronized = 6701,
  PtpSynchonized = 6702,
  IrigSynchronized = 6703,
  NtpSynchronized = 6704
}
```

Visual Basic

```
Public Enum DeviceSynchronization
  NotSynchrnonized = 6700
  CableSynchronized = 6701
  PtpSynchonized = 6702
  IrigSynchronized = 6703
  NtpSynchronized = 6704
End Enum
```

ファイル

DeviceSynchronization.cs

メンバー

メンバー	説明
NotSynchrnonized	非同期。
CableSynchronized	ケーブル経由で同期。
PtpSynchonized	PTPで同期。
IrigSynchronized	IRIGチャネル経由で同期。
NtpSynchronized	NTPで同期。

Hbm.Api.Mgc.Enums.DigitalOutMode Enumeration

デジタル信号のモード(コネクタが出力方向のデジタル信号にのみ有効)。

C++

```
enum DigitalOutMode {
  LogicalCombination = 1940,
  Adjustable = 1942,
  LinkedToSignal = 1941
};
```

C#

```
public enum DigitalOutMode {
  LogicalCombination = 1940,
  Adjustable = 1942,
  LinkedToSignal = 1941
}
```

Visual Basic

```
Public Enum DigitalOutMode
  LogicalCombination = 1940
  Adjustable = 1942
  LinkedToSignal = 1941
End Enum
```

ファイル

DigitalOutMode.cs

メンバー

メンバー	説明
LogicalCombination	制限値スイッチと他のボードのオーバーフローステータスの論理的な組み合わせ。
Adjustable	出力はHighまたはLowに調整可能。
LinkedToSignal	出力はデバイスの信号のステータスにリンク。

Hbm.Api.Mgc.Enums.DischargeTimeType Enumeration

放電時間の種類。

C++

```
enum DischargeTimeType {
  Slow = 31,
  Medium = 32,
  Fast = 33
};
```

C#

```
internal enum DischargeTimeType {
  Slow = 31,
  Medium = 32,
  Fast = 33
}
```

Visual Basic

```
Friend Enum DischargeTimeType
  Slow = 31
  Medium = 32
  Fast = 33
End Enum
```

ファイル

DischargeTimeType.cs

メンバー

メンバー	説明
Slow	遅い放電時間。
Medium	中程度の放電時間。
Fast	早い放電時間。

Hbm.Api.Mgc.Enums.ExcitationAmplitude Enumeration

MGCがサポートする励起振幅。

C++

```
[AdditionalValueType(typeof(decimal))]
enum ExcitationAmplitude {
  NoSupply = 10,
  V1_0 = 11,
  V1_25 = 12,
  V2_5 = 13,
  V5_0 = 14,
  V10_0 = 15,
  Current = 16,
  V0_2 = 17,
  V0_5 = 18
};
```

C#

```
[AdditionalValueType(typeof(decimal))]
internal enum ExcitationAmplitude {
  NoSupply = 10,
  V1_0 = 11,
  V1_25 = 12,
  V2_5 = 13,
  V5_0 = 14,
  V10_0 = 15,
  Current = 16,
  V0_2 = 17,
  V0_5 = 18
}
```

Visual Basic

```
<AdditionalValueType(typeof(decimal))>
Friend Enum ExcitationAmplitude
  NoSupply = 10
  V1_0 = 11
  V1_25 = 12
  V2_5 = 13
  V5_0 = 14
  V10_0 = 15
  Current = 16
  V0_2 = 17
  V0_5 = 18
End Enum
```

ファイル

ExcitationAmplitude.cs

メンバー

メンバー	説明
Current	非常に特殊なML10/AP18(ピエゾ アクティブセンサ)トランスデューサの励起はVではなくmAです!!!

備考

列挙項目の値は、デバイスのクエリ言語によって返される励起振幅数値と同じです。

Hbm.Api.Mgc.Enums.FilterCharacteristic Enumeration

MGCがサポートするフィルタ特性。

C++

```
enum FilterCharacteristic {
  NoFilter = 140,
  Butterworth = 141,
  Bessel = 142
};
```

C#

```
internal enum FilterCharacteristic {
  NoFilter = 140,
  Butterworth = 141,
  Bessel = 142
}
```

Visual Basic

```
Friend Enum FilterCharacteristic
  NoFilter = 140
  Butterworth = 141
  Bessel = 142
End Enum
```

ファイル

FilterCharacteristic.cs

備考

列挙項目の値は、デバイスのクエリ言語によって返されるフィルタ特性番号と同じです。

Hbm.Api.Mgc.Enums.FilterFrequency Enumeration

MGCがサポートするフィルタ周波数(Hz)。

C++

```
[AdditionalValueType(typeof(decimal))]
enum FilterFrequency : public int {
  Hz0_01 = 901,
  Hz0_012 = 902,
  Hz0_015 = 903,
  Hz0_02 = 904,
  Hz0_025 = 905,
  Hz0_03 = 906,
  Hz0_04 = 907,
  Hz0_05 = 908,
  Hz0_06 = 909,
  Hz0_07 = 910,
  Hz0_075 = 911,
  Hz0_08 = 912,
  Hz0_09 = 913,
  Hz0_1 = 914,
  Hz0_12 = 915,
  Hz0_15 = 916,
  Hz0_2 = 917,
  Hz0_25 = 918,
  Hz0_3 = 919,
  Hz0_4 = 920,
  Hz0_5 = 921,
  Hz0_6 = 922,
  Hz0_7 = 923,
  Hz0_75 = 924,
  Hz0_80 = 925,
  Hz0_90 = 926,
  Hz1_0 = 927,
  Hz1_2 = 928,
  Hz1_25 = 929,
  Hz1_5 = 930,
  Hz2_0 = 931,
  Hz2_5 = 932,
  Hz3_0 = 933,
  Hz4_0 = 934,
  Hz5_0 = 935,
  Hz6_0 = 936,
  Hz7_0 = 937,
  Hz7_5 = 938,
  Hz8_0 = 939,
  Hz9_0 = 940,
  Hz10 = 941,
  Hz12 = 942,
  Hz15 = 943,
  Hz17_50 = 944,
  Hz20_0 = 945,
  Hz25_0 = 946,
  Hz30_0 = 947,
  Hz40_0 = 948,
  Hz50_0 = 949,
  Hz60_0 = 950,
  Hz70_0 = 951,
  Hz75_0 = 952,
```

```
  Hz80_0 = 953,
  Hz90_0 = 954,
  Hz100 = 955,
  Hz120 = 956,
  Hz150 = 957,
  Hz200 = 958,
  Hz250 = 959,
  Hz300 = 960,
  Hz400 = 961,
  Hz500 = 962,
  Hz600 = 964,
  Hz700 = 965,
  Hz750 = 966,
  Hz800 = 967,
  Hz900 = 968,
  Hz1000 = 969,
  Hz1100 = 970,
  Hz1200 = 971,
  Hz1500 = 972,
  Hz2000 = 973,
  Hz2400 = 974,
  Hz2500 = 975,
  Hz3000 = 976,
  Hz4000 = 977,
  Hz5000 = 978,
  Hz6000 = 979,
  Hz7000 = 980,
  Hz7500 = 981,
  Hz8000 = 982,
  Hz9000 = 983,
  Hz10000 = 1100,
  Hz50000 = 1140,
  Hz100000 = 1150
};
```

C#

```
[AdditionalValueType(typeof(decimal))]
internal enum FilterFrequency {
  Hz0_01 = 901,
  Hz0_012 = 902,
  Hz0_015 = 903,
  Hz0_02 = 904,
  Hz0_025 = 905,
  Hz0_03 = 906,
  Hz0_04 = 907,
  Hz0_05 = 908,
  Hz0_06 = 909,
  Hz0_07 = 910,
  Hz0_075 = 911,
  Hz0_08 = 912,
  Hz0_09 = 913,
  Hz0_1 = 914,
  Hz0_12 = 915,
  Hz0_15 = 916,
  Hz0_2 = 917,
  Hz0_25 = 918,
  Hz0_3 = 919,
  Hz0_4 = 920,
```


現在位置: Supported devices -> Hbm.Api.Mgc Namespace ->
Hbm.Api.Mgc.Enums Namespace

```
    Hz0_5 = 921,
    Hz0_6 = 922,
    Hz0_7 = 923,
    Hz0_75 = 924,
    Hz0_80 = 925,
    Hz0_90 = 926,
    Hz1_0 = 927,
    Hz1_2 = 928,
    Hz1_25 = 929,
    Hz1_5 = 930,
    Hz2_0 = 931,
    Hz2_5 = 932,
    Hz3_0 = 933,
    Hz4_0 = 934,
    Hz5_0 = 935,
    Hz6_0 = 936,
    Hz7_0 = 937,
    Hz7_5 = 938,
    Hz8_0 = 939,
    Hz9_0 = 940,
    Hz10 = 941,
    Hz12 = 942,
    Hz15 = 943,
    Hz17_50 = 944,
    Hz20_0 = 945,
    Hz25_0 = 946,
    Hz30_0 = 947,
    Hz40_0 = 948,
    Hz50_0 = 949,
    Hz60_0 = 950,
    Hz70_0 = 951,
    Hz75_0 = 952,
    Hz80_0 = 953,
    Hz90_0 = 954,
    Hz100 = 955,
    Hz120 = 956,
    Hz150 = 957,
    Hz200 = 958,
    Hz250 = 959,
    Hz300 = 960,
    Hz400 = 961,
    Hz500 = 962,
    Hz600 = 964,
    Hz700 = 965,
    Hz750 = 966,
    Hz800 = 967,
    Hz900 = 968,
    Hz1000 = 969,
    Hz1100 = 970,
    Hz1200 = 971,
    Hz90_0 = 954,
    Hz100 = 955,
    Hz120 = 956,
    Hz150 = 957,
    Hz200 = 958,
    Hz250 = 959,
    Hz300 = 960,
    Hz400 = 961,
    Hz500 = 962,
    Hz600 = 964,
```

```
  Hz700 = 965,
  Hz750 = 966,
  Hz800 = 967,
  Hz900 = 968,
  Hz1000 = 969,
  Hz1100 = 970,
  Hz1200 = 971,
  Hz1500 = 972,
  Hz2000 = 973,
  Hz2400 = 974,
  Hz2500 = 975,
  Hz3000 = 976,
  Hz4000 = 977,
  Hz5000 = 978,
  Hz6000 = 979,
  Hz7000 = 980,
  Hz7500 = 981,
  Hz8000 = 982,
  Hz9000 = 983,
  Hz10000 = 1100,
  Hz50000 = 1140,
  Hz100000 = 1150
}
```

Visual Basic

```
<AdditionalValueType(typeof(decimal))>
Friend Enum FilterFrequency
  Hz0_01 = 901,
  Hz0_012 = 902,
  Hz0_015 = 903,
  Hz0_02 = 904,
  Hz0_025 = 905,
  Hz0_03 = 906,
  Hz0_04 = 907,
  Hz0_05 = 908,
  Hz0_06 = 909,
  Hz0_07 = 910,
  Hz0_075 = 911,
  Hz0_08 = 912,
  Hz0_09 = 913,
  Hz0_1 = 914,
  Hz0_12 = 915,
  Hz0_15 = 916,
  Hz0_2 = 917,
  Hz0_25 = 918,
  Hz0_3 = 919,
  Hz0_4 = 920,
  Hz0_5 = 921,
  Hz0_6 = 922,
  Hz0_7 = 923,
  Hz0_75 = 924,
  Hz0_80 = 925,
  Hz0_90 = 926,
  Hz1_0 = 927,
  Hz1_2 = 928,
  Hz1_25 = 929,
  Hz1_5 = 930,
  Hz2_0 = 931,
```

```
    Hz2_5 = 932,
    Hz3_0 = 933,
    Hz4_0 = 934,
    Hz5_0 = 935,
    Hz6_0 = 936,
    Hz7_0 = 937,
    Hz7_5 = 938,
    Hz8_0 = 939,
    Hz9_0 = 940,
    Hz10 = 941,
    Hz12 = 942,
    Hz15 = 943,
    Hz17_50 = 944,
    Hz20_0 = 945,
    Hz25_0 = 946,
    Hz30_0 = 947,
    Hz40_0 = 948,
    Hz50_0 = 949,
    Hz60_0 = 950,
    Hz70_0 = 951,
    Hz75_0 = 952,
    Hz80_0 = 953,
    Hz90_0 = 954,
    Hz100 = 955,
    Hz120 = 956,
    Hz150 = 957,
    Hz200 = 958,
    Hz250 = 959,
    Hz300 = 960,
    Hz400 = 961,
    Hz500 = 962,
    Hz600 = 964,
    Hz700 = 965,
    Hz750 = 966,
    Hz800 = 967,
    Hz900 = 968,
    Hz1000 = 969,
    Hz1100 = 970,
    Hz1200 = 971,
    Hz90_0 = 954,
    Hz100 = 955,
    Hz120 = 956,
    Hz150 = 957,
    Hz200 = 958,
    Hz250 = 959,
    Hz300 = 960,
    Hz400 = 961,
    Hz500 = 962,
    Hz600 = 964,
    Hz700 = 965,
    Hz750 = 966,
    Hz800 = 967,
    Hz900 = 968,
    Hz1000 = 969,
    Hz1100 = 970,
    Hz1200 = 971,
    Hz1500 = 972,
    Hz2000 = 973,
    Hz2400 = 974,
    Hz2500 = 975,
    Hz3000 = 976,
```

```
  Hz4000 = 977,
  Hz5000 = 978,
  Hz6000 = 979,
  Hz7000 = 980,
  Hz7500 = 981,
  Hz8000 = 982,
  Hz9000 = 983,
  Hz10000 = 1100,
  Hz50000 = 1140,
  Hz100000 = 1150
End Enum
```

ファイル
FilterFrequency.cs

備考
列挙項目の値は、デバイスのクエリ言語によって返されるフィルタ周波数番号と同じです。

Hbm.Api.Mgc.Enums.FrameFormat Enumeration

MGC CANセンサのフレームフォーマット。

C++

```
enum FrameFormat {
  Standard_11BitID = 0,
  Extended_29BitID = 1
};
```

C#

```
public enum FrameFormat {
  Standard_11BitID = 0,
  Extended_29BitID = 1
}
```

Visual Basic

```
Public Enum FrameFormat
  Standard_11BitID = 0,
  Extended_29BitID = 1
End Enum
```

ファイル

FrameFormat.cs

メンバー

メンバー	説明
Standard_11BitID	IDに11ビットを使用する標準フレーム形式。
Extended_29BitID	IDに29ビットを使用する拡張フレーム形式。

Hbm.Api.Mgc.Enums.InputType Enumeration

入力の種類。

C++

```
enum InputType {
  In5V = 21,
  In100mV = 22,
  Direct = 23,
  Inductive = 24

};
```

C#

```
internal enum InputType {
  In5V = 21,
  In100mV = 22,
  Direct = 23,
  Inductive = 24
}
```

Visual Basic

```
Friend Enum InputType
  In5V = 21
  In100mV = 22
  Direct = 23
  Inductive = 24
End Enum
```

ファイル

InputType.cs

Hbm.Api.Mgc.Enums.LimitSwitchOperatingDirection Enumeration

MGCリミットスイッチの動作方向。

C++

```
enum LimitSwitchOperatingDirection {
  Off = -1,
  AboveLimit = 130,
  BelowLimit = 131,
  InBand = 132,
  OutOfBand = 133
};
```

C#

```
public enum LimitSwitchOperatingDirection {
  Off = -1,
  AboveLimit = 130,
  BelowLimit = 131,
  InBand = 132,
  OutOfBand = 133
}
```

Visual Basic

```
Public Enum LimitSwitchOperatingDirection
  Off = -1
  AboveLimit = 130
  BelowLimit = 131
  InBand = 132
  OutOfBand = 133
End Enum
```

ファイル

LimitSwitchOperatingDirection.cs

メンバー

メンバー	説明
Off	リミットスイッチは評価しません。
AboveLimit	リミットスイッチは指定値を超えるとトリガされます。
BelowLimit	リミットスイッチは指定値を下回るとトリガされます。
InBand	リミット スイッチは指定した帯域幅内でトリガされます。
OutOfBand	リミット スイッチは、指定した帯域幅を超えるとトリガされます。

備考

列挙項目の値は、デバイスのクエリ言語によって返される値と同じです。

Hbm.Api.Mgc.Enums.MeasurementSignalType Enumeration

測定信号の種類。MGCを接続すると、風袋が0.0に設定されなくなることに注意してください! したがって、信号の種類によって測定値が異なる場合があります(これを使用して、たとえばグロス信号を要求するML77によって発生する可能性のあるリンク リソースエラーを解決します...)。

C++

```
enum MeasurementSignalType {
  GrossSignalType = 214,
  NetSignalType = 215
};
```

C#

```
public enum MeasurementSignalType {
  GrossSignalType = 214,
  NetSignalType = 215
}
```

Visual Basic

```
Public Enum MeasurementSignalType
  GrossSignalType = 214
  NetSignalType = 215
End Enum
```

ファイル

MeasurementSignalType.cs

メンバー

メンバー	説明
GrossSignalType	グロス信号。
NetSignalType	ネット信号。

Hbm.Api.Mgc.Enums.MgcDeviceStatus Enumeration
Deviceのステータス。

C++

```
[Flags()]
enum MgcDeviceStatus {
  NoError = 0,
  DeviceDependentError = 8,
  ParameterDefective = 16,
  UnknownCommand = 32
};
```

C#

```
[Flags()]
public enum MgcDeviceStatus {
  NoError = 0,
  DeviceDependentError = 8,
  ParameterDefective = 16,
  UnknownCommand = 32
}
```

Visual Basic

```
<Flags()>
Public Enum MgcDeviceStatus
  NoError = 0
  DeviceDependentError = 8
  ParameterDefective = 16
  UnknownCommand = 32
End Enum
```

ファイル
DeviceStatus.cs

メンバー

メンバー	説明
NoError	エラーなし。
DeviceDependentError	デバイス依存のエラーが発生。
ParameterDefective	間違った、または不適切なパラメータをデバイスに送信。
UnknownCommand	不明なコマンドがデバイスをデバイスに送信。

Hbm.Api.Mgc.Enums.MgcSensorType Enumeration

センサの種類。

C++

```
enum MgcSensorType {
  Blocked = 4,
  Inactive = 6,
  Can = 0,
  FullBridge = 350,
  HalfBridge = 351,
  QuaterBridge = 352,
  StrainGaugeFullBridge = 353,
  StrainGaugeHalfBridge = 354,
  StrainGaugeQuarterBridge = 355,
  InductiveFullBridge = 356,
  InductiveHalfBridge = 357,
  FullBridgeLowLevel = 358,
  HalfBridgeLowLevel = 359,
  FullBridgeHighLevel = 360,
  HalfBridgeHighLevel = 361,
  StrainGaugeFullBridge120 = 362,
  StrainGaugeFullBridge350 = 363,
  StrainGaugeFullBridge700 = 364,
  StrainGaugeHalfBridge120 = 365,
  StrainGaugeHalfBridge350 = 366,
  StrainGaugeHalfBridge700 = 367,
  QuaterBridge120Ohm4L = 368,
  QuaterBridge350Ohm4L = 369,
  QuaterBridge700Ohm4L = 370,
  QuaterBridge1000Ohm4L = 376,
  QuaterBridge120Ohm3L = 371,
  QuaterBridge350Ohm3L = 372,
  QuaterBridge700Ohm3L = 373,
  QuaterBridge1000Ohm3L = 374,
  LVDT = 380,
  Potentiometer = 385,
  DCV = 420,
  DCA = 421,
  DC75mV = 425,
  DC10V = 426,
  DC20mA = 427,
  DC2V = 428,
  DC100mV = 429,
  DC60V = 433,
  TypeJ = 450,
  TypeK = 451,
  TypeT = 452,
  TypeS = 453,
  TypeB = 454,
  TypeE = 455,
  TypeR = 456,
  TypeN = 457,
  Resistance = 475,
  Ohm500 = 476,
  Ohm5000 = 477,
  Ohm2500 = 478,
  Ohm50 = 479,
  Pt10 = 500,
  Pt100 = 501,
```


現在位置: Supported devices -> Hbm.Api.Mgc Namespace -> Hbm.Api.Mgc.Enums Namespace

```
  Pt500 = 502,
  Pt1000 = 503,
  InternalTemperature = 510,
  Frequency = 520,
  Period = 521,
  PulseCounting = 525,
  Pwm = 527,
  Freq2kHz = 530,
  Freq20kHz = 531,
  Freq200kHz = 532,
  Freq2MHz = 537,
  Freq500kHz = 538,
  Deltatron01V = 550,
  Deltatron1V = 551,
  Deltatron10V = 552,
  Charge = 570,
  Charge01nC = 571,
  Charge1nC = 572,
  Charge10nC = 573,
  Charge100nC = 574,
  VirtualSensor = 575,
  NoSensor = 599
};
```

C#

```
internal enum MgcSensorType {
  Blocked = 4,
  Inactive = 6,
  Can = 0,
  FullBridge = 350,
  HalfBridge = 351,
  QuaterBridge = 352,
  StrainGaugeFullBridge = 353,
  StrainGaugeHalfBridge = 354,
  StrainGaugeQuarterBridge = 355,
  InductiveFullBridge = 356,
  InductiveHalfBridge = 357,
  FullBridgeLowLevel = 358,
  HalfBridgeLowLevel = 359,
  FullBridgeHighLevel = 360,
  HalfBridgeHighLevel = 361,
  StrainGaugeFullBridge120 = 362,
  StrainGaugeFullBridge350 = 363,
  StrainGaugeFullBridge700 = 364,
  StrainGaugeHalfBridge120 = 365,
  StrainGaugeHalfBridge350 = 366,
  StrainGaugeHalfBridge700 = 367,
  QuaterBridge120Ohm4L = 368,
  QuaterBridge350Ohm4L = 369,
  QuaterBridge700Ohm4L = 370,
  QuaterBridge1000Ohm4L = 376,
  QuaterBridge120Ohm3L = 371,
  QuaterBridge350Ohm3L = 372,
  QuaterBridge700Ohm3L = 373,
  QuaterBridge1000Ohm3L = 374,
  LVDT = 380,
  Potentiometer = 385,
  DCV = 420,
  DCA = 421,
```

```
  DC75mV = 425,
  DC10V = 426,
  DC20mA = 427,
  DC2V = 428,
  DC100mV = 429,
  DC60V = 433,
  TypeJ = 450,
  TypeK = 451,
  TypeT = 452,
  TypeS = 453,
  TypeB = 454,
  TypeE = 455,
  TypeR = 456,
  TypeN = 457,
  Resistance = 475,
  Ohm500 = 476,
  Ohm5000 = 477,
  Ohm2500 = 478,
  Ohm50 = 479,
  Pt10 = 500,
  Pt100 = 501,
  Pt500 = 502,
  Pt1000 = 503,
  InternalTemperature = 510,
  Frequency = 520,
  Period = 521,
  PulseCounting = 525,
  Pwm = 527,
  Freq2kHz = 530,
  Freq20kHz = 531,
  Freq200kHz = 532,
  Freq2MHz = 537,
  Freq500kHz = 538,
  Deltatron01V = 550,
  Deltatron1V = 551,
  Deltatron10V = 552,
  Charge = 570,
  Charge01nC = 571,
  Charge1nC = 572,
  Charge10nC = 573,
  Charge100nC = 574,
  VirtualSensor = 575,
  NoSensor = 599
}
```

Visual Basic

```
internal enum MgcSensorType {
  Blocked = 4
  Inactive = 6
  Can = 0
  FullBridge = 350
  HalfBridge = 351
  QuaterBridge = 352
  StrainGaugeFullBridge = 353
  StrainGaugeHalfBridge = 354
  StrainGaugeQuarterBridge = 355
  InductiveFullBridge = 356
  InductiveHalfBridge = 357
```

```
    FullBridgeLowLevel = 358
    HalfBridgeLowLevel = 359
    FullBridgeHighLevel = 360
    HalfBridgeHighLevel = 361
    StrainGaugeFullBridge120 = 362
    StrainGaugeFullBridge350 = 363
    StrainGaugeFullBridge700 = 364
    StrainGaugeHalfBridge120 = 365
    StrainGaugeHalfBridge350 = 366
    StrainGaugeHalfBridge700 = 367
    QuaterBridge120Ohm4L = 368
    QuaterBridge350Ohm4L = 369
    QuaterBridge700Ohm4L = 370
    QuaterBridge1000Ohm4L = 376
    QuaterBridge120Ohm3L = 371
    QuaterBridge350Ohm3L = 372
    QuaterBridge700Ohm3L = 373
    QuaterBridge1000Ohm3L = 374
    LVDT = 380
    Potentiometer = 385
    DCV = 420
    DCA = 421
    DC75mV = 425
    DC10V = 426
    DC20mA = 427
    DC2V = 428
    DC100mV = 429
    DC60V = 433
    TypeJ = 450
    TypeK = 451
    TypeT = 452
    TypeS = 453
    TypeB = 454
    TypeE = 455
    TypeR = 456
    TypeN = 457
    Resistance = 475
    Ohm500 = 476
    Ohm5000 = 477
    Ohm2500 = 478
    Ohm50 = 479
    Pt10 = 500
    Pt100 = 501
    Pt500 = 502
    Pt1000 = 503
    InternalTemperature = 510
    Frequency = 520
    Period = 521
    PulseCounting = 525
    Pwm = 527
    Freq2kHz = 530
    Freq20kHz = 531
    Freq200kHz = 532
    Freq2MHz = 537
    Freq500kHz = 538
    Deltatron01V = 550
    Deltatron1V = 551
    Deltatron10V = 552
    Charge = 570
    Charge01nC = 571
```

```
  Charge1nC = 572
  Charge10nC = 573
  Charge100nC = 574
  VirtualSensor = 575
  NoSensor = 599
End Enum
```

ファイル
MgcSensorType.cs

メンバー

メンバー	説明
Can	実際にはCANには特定の種類はありませんが、ML71は"SAD?,?"を介して可能なセンサ種類をクエリすると0を返します!

備考
列挙項目の値は、デバイスのクエリ言語によって返されるSensorType番号と同じです。

Hbm.Api.Mgc.Enums.ModeSignalByteOrder Enumeration

MGC CAN信号のモード信号のバイトオーダー。

C++

```
enum ModeSignalByteOrder {
  Intel = 0,
  Motorola = 1
};
```

C#

```
public enum ModeSignalByteOrder : int {
  Intel = 0,
  Motorola = 1
}
```

Visual Basic

```
Public Enum ModeSignalByteOrder
  Intel = 0
  Motorola = 1
End Enum
```

ファイル

ModeSignalByteOrder.cs

メンバー

メンバー	説明
Intel	リトルエンディアン(最下位バイトは最下位のメモリ位置に格納されます)。
Motorola	ビッグエンディアン(最上位バイトは最下位のメモリ位置に格納されます)。

Hbm.Api.Mgc.Enums.SampleRate Enumeration

MGCがサポートするサンプルレート。

C++

```
[AdditionalValueType(typeof(decimal))]
enum SampleRate {
  _1Hz = 6300,
  _2Hz = 6301,
  _5Hz = 6302,
  _10Hz = 6303,
  _20Hz = 6326,
  _25Hz = 6304,
  _50Hz = 6305,
  _75Hz = 6307,
  _100Hz = 6308,
  _150Hz = 6309,
  _200Hz = 6310,
  _300Hz = 6311,
  _600Hz = 6313,
  _1200Hz = 6315,
  _2400Hz = 6317,
  _4800Hz = 6319,
  _9600Hz = 6320,
  _19200Hz = 6345,
};
```

C#

```
[AdditionalValueType(typeof(decimal))]
internal enum SampleRate {
  _1Hz = 6300,
  _2Hz = 6301,
  _5Hz = 6302,
  _10Hz = 6303,
  _20Hz = 6326,
  _25Hz = 6304,
  _50Hz = 6305,
  _75Hz = 6307,
  _100Hz = 6308,
  _150Hz = 6309,
  _200Hz = 6310,
  _300Hz = 6311,
  _600Hz = 6313,
  _1200Hz = 6315,
  _2400Hz = 6317,
  _4800Hz = 6319,
  _9600Hz = 6320,
  _19200Hz = 6345,
}
```

Visual Basic

```
<AdditionalValueType(typeof(decimal))>
Friend Enum SampleRate
  _1Hz = 6300
  _2Hz = 6301
  _5Hz = 6302
  _10Hz = 6303
  _20Hz = 6326
  _25Hz = 6304
  _50Hz = 6305
  _75Hz = 6307
  _100Hz = 6308
  _150Hz = 6309
  _200Hz = 6310
  _300Hz = 6311
  _600Hz = 6313
  _1200Hz = 6315
  _2400Hz = 6317
  _4800Hz = 6319
  _9600Hz = 6320
  _19200Hz = 6345
End Enum
```

ファイル

SampleRate.cs

備考

列挙項目の値は、デバイスのクエリ言語によって返されるサンプル レートと同じです。

Hbm.Api.Mgc.Enums.SignalType Enumeration

MGCがサポートする信号の種類。

C++

```
enum SignalType {
  StandardSignal = 16,
  ModeSignal = 32,
  ModeDependentSignal = 64
};
```

C#

```
public enum SignalType {
  StandardSignal = 16,
  ModeSignal = 32,
  ModeDependentSignal = 64
}
```

Visual Basic

```
Public Enum SignalType
  StandardSignal = 16
  ModeSignal = 32
  ModeDependentSignal = 64
End Enum
```

ファイル

SignalType.cs

メンバー

メンバー	説明
StandardSignal	標準信号。
ModeSignal	モード信号。
ModeDependentSignal	モード依存信号

Hbm.Api.Mgc.Enums.TimeChannelType Enumeration

時間チャネルの形式。注意!!! 現在サポートされている種類はNtp64BitTimeのみです。他の種類はすべて網羅性のためだけのものです...

C++

```
enum TimeChannelType {
  Ntp64BitTime = 7002
};
```

C#

```
public enum TimeChannelType {
  Ntp64BitTime = 7002
}
```

Visual Basic

```
Public Enum TimeChannelType
  Ntp64BitTime = 7002
End Enum
```

ファイル

TimeChannelType.cs

メンバー

メンバー	説明
Ntp64BitTime	Ntp64ビットタイムチャネル形式。

Hbm.Api.Mgc.Enums.TransducerSensitivity Enumeration

MGCがサポートするトランスデューサの感度。

C++

```
enum TransducerSensitivity {
  _4mVV = 778,
  _100mVV  = 774,
  _1000mVV  = 776
};
```

C#

```
internal enum TransducerSensitivity {
  _4mVV = 778,
  _100mVV  = 774,
  _1000mVV  = 776
}
```

Visual Basic

```
Friend Enum TransducerSensitivity
  _4mVV = 778
  _100mVV  = 774
  _1000mVV  = 776
End Enum
```

ファイル

TransducerSensitivity.cs

備考

列挙項目の値は、デバイスのクエリ言語によって返される励起振幅数値と同じです。

Hbm.Api.Mgc.Enums.TriggerOperationMode Enumeration

測定開始時のMGCの動作を定義するトリガ操作モード。

C++

```
enum TriggerOperationMode {
  StandAlone = 6713,
  TriggerSlave  = 6712,
  TriggerMaster  = 6711
};
```

C#

```
public enum TriggerOperationMode {
  StandAlone = 6713,
  TriggerSlave  = 6712,
  TriggerMaster  = 6711
}
```

Visual Basic

```
Public Enum TriggerOperationMode
  StandAlone = 6713
  TriggerSlave  = 6712
  TriggerMaster  = 6711
End Enum
```

ファイル

TriggerOperationMode.cs

メンバー

メンバー	説明
StandAlone	スタンドアロン(デバイスは他のデバイスとのハードウェア同期を使用しません)。
TriggerSlave	スレーブ(デバイスは測定を開始するために開始トリガをリッスンします)。
TriggerMaster	マスタ(デバイスが他のデバイスをトリガして測定を開始します)。

Hbm.Api.Mgc.Enums.Wiring Enumeration

MGCの配線の種類。

C++

```
enum Wiring {
  Wiring3L = 25,
  Wiring4L = 26
};
```

C#

```
public enum Wiring {
  Wiring3L = 25,
  Wiring4L = 26
}
```

Visual Basic

```
Public Enum Wiring
  Wiring3L = 25
  Wiring4L = 26
End Enum
```

ファイル

Wiring.cs

備考

列挙項目の値は、デバイスのクエリ言語によって返される配線の種類と同じです。

Hbm.Api.Mgc.Signals Namespace

これは名前空間Hbm.Api.Mgc.Signalsです。

クラス

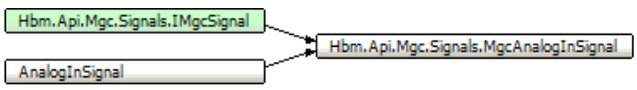
名前	説明
MgcAnalogInSignal	物理センサから送られる測定信号を表します。
MgcAnalogOutSignal	MGC固有のアナログ出力信号を表します。
MgcCanInSignal	MGC固有のCAN信号の実装。
MgcDigitalSignal	マスクされたビットに応じて、デジタル信号の一部(実際にはMGCの"ネイティブ"デジタル入力またはデジタル出力信号の1ビット)を表します。
MgcVirtualSignal	センサを接続しない信号(演算信号など)を表します。

インタフェース

	名前	説明
	IMgcSignal	MgcSignalの追加プロパティのインタフェース。

MgcAnalogInSignal Class

物理センサから送られる測定信号を表します。



C++

```
class MgcAnalogInSignal : public AnalogInSignal, public IMgcSignal;
```

C#

```
public class MgcAnalogInSignal : AnalogInSignal, IMgcSignal;
```

Visual Basic

```
Public Class MgcAnalogInSignal
Inherits AnalogInSignal
Inherits IMgcSignal
```

ファイル

MgcAnalogInSignal.cs

現在位置: Supported devices -> Hbm.Api.Mgc Namespace ->
Hbm.Api.Mgc.Signals Namespace

MgcAnalogInSignal Members

次の表は、MgcAnalogInSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Position	アンプのコネクタの位置。
	Slot	コネクタが属するスロット。

MgcAnalogInSignal Methods

MgcAnalogInSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

MgcAnalogInSignal.GetUniqueID Method

信号が属するデバイス内で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別子。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェア モジュール上の信号を識別するためにストリーミングで使用する情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。

MgcAnalogInSignal Properties

MgcAnalogInSignalクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Position	アンプのコネクタの位置。
	Slot	コネクタが属するスロット。

MgcAnalogInSignal.IndicationScaling Property

-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。

C++

```
__property MgcIndicationScalingValues IndicationScaling;
```

C#

```
public MgcIndicationScalingValues IndicationScaling;
```

Visual Basic

```
Public Property IndicationScaling() As MgcIndicationScalingValues
```

MgcAnalogInSignal.Position Property

アンプのコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcAnalogInSignal.Slot Property

コネクタが属するスロット。

C++

```
__property int Slot;
```

C#

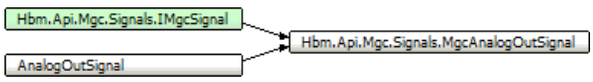
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcAnalogOutSignal Class

MGC固有のアナログ出力信号を表します。



C++

```
class MgcAnalogOutSignal : public AnalogOutSignal, public IMgcSignal;
```

C#

```
public class MgcAnalogOutSignal : AnalogOutSignal, IMgcSignal;
```

Visual Basic

```
Public Class MgcAnalogOutSignal
Inherits AnalogOutSignal
Inherits IMgcSignal
```

ファイル

MgcAnalogOutSignal.cs

備考

この種の信号を測定に含めることはできません。

MgcAnalogOutSignal Members

次の表は、MgcAnalogOutSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	AnalogOutMode	アナログ出力のソースを表すモード。
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Position	信号が属するスロット内の位置。
	Slot	コネクタが属するデバイスのスロット番号。

MgcAnalogOutSignal Methods

MgcAnalogOutSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

MgcAnalogOutSignal.GetUniqueID Method

信号が属するデバイス内で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別子。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェア モジュール上の信号を識別するためにストリーミングで使用する情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。

MgcAnalogOutSignal Properties

MgcAnalogOutSignalクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	AnalogOutMode	アナログ出力のソースを表すモード。
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Position	信号が属するスロット内の位置。
	Slot	コネクタが属するデバイスのスロット番号。

MgcAnalogOutSignal.AnalogOutMode Property

アナログ出力のソースを表すモード。

C++

```
[JsonProperty]
__property AnalogOutMode AnalogOutMode;
```

C#

```
[JsonProperty]
public AnalogOutMod AnalogOutMode;
```

Visual Basic

```
<JsonProperty>
Public Property AnalogOutMode() As AnalogOutMode
```

MgcAnalogOutSignal.IndicationScaling Property

-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。

C++

```
__property MgcIndicationScalingValues IndicationScaling;
```

C#

```
public MgcIndicationScalingValues IndicationScaling;
```

Visual Basic

```
Public Property IndicationScaling() As MgcIndicationScalingValues
```

MgcAnalogOutSignal.Position Property

信号が属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcAnalogOutSignal.Slot Property

コネクタが属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

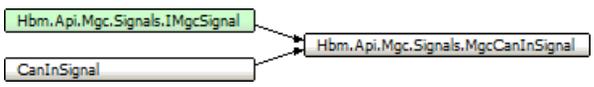
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcCanInSignal Class

MGC固有のCAN信号の実装。



C++

```
class MgcCanInSignal : public CanInSignal, public IMgcSignal;
```

C#

```
public class MgcCanInSignal : CanInSignal, IMgcSignal;
```

Visual Basic

```
Public Class MgcCanInSignal
Inherits CanInSignal
Inherits IMgcSignal
```

ファイル

MgcCanInSignal.cs

MgcCanInSignal Members

次の表は、MgcCanInSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Position	アンプ上のコネクタの位置。
	Slot	コネクタが属するスロット。

MgcCanInSignal Methods

MgcCanInSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

MgcCanInSignal.GetUniqueID Method

信号が属するデバイス内で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別子。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェア モジュール上の信号を識別するためにストリーミングで使用する情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。

MgcCanInSignal Properties

MgcCanInSignalクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Position	アンプ上のコネクタの位置。
	Slot	コネクタが属するスロット。

MgcCanInSignal.IndicationScaling Property

-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。

C++

```
__property MgcIndicationScalingValues IndicationScaling;
```

C#

```
public MgcIndicationScalingValues IndicationScaling;
```

Visual Basic

```
Public Property IndicationScaling() As MgcIndicationScalingValues
```

MgcCanInSignal.Position Property

アンプ上のコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcCanInSignal.Slot Property

コネクタが属するスロット。

C++

```
__property int Slot;
```

C#

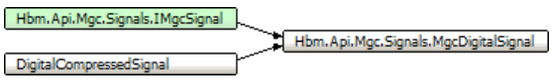
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcDigitalSignal Class

マスクされたビットに応じて、デジタル信号の一部(実際にはMGCの"ネイティブ"デジタル入力またはデジタル出力信号の1ビット)を表します。



C++

```
class MgcDigitalSignal : public DigitalCompressedSignal, public IMgcSignal;
```

C#

```
public class MgcDigitalSignal : DigitalCompressedSignal, IMgcSignal;
```

Visual Basic

```
Public Class MgcDigitalSignal
Inherits DigitalCompressedSignal
Inherits IMgcSignal
```

ファイル

MgcDigitalSignal.cs

MgcDigitalSignal Members

次の表は、MgcDigitalSignalによって公開されるメンバーの一覧です。

Internal Properties

	名前	説明
	DigitalOutMode	デジタル出力のソースを記述するモード。デジタル出力ピンをhighまたはlowに設定する前に、追加の(時間のかかる)コマンドをmgcに送信する必要があるかどうかを決定するために使用します。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Pin	この信号にマップされているデジタルピン。
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

MgcDigitalSignal Methods

MgcDigitalSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

MgcDigitalSignal.GetUniqueID Method

信号が属するデバイス内で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別子。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェア モジュール上の信号を識別するためにストリーミングで使用する情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。

MgcDigitalSignal Properties

MgcDigitalSignalクラスのプロパティは次のとおりです。

Internal Properties

	名前	説明
	DigitalOutMode	デジタル出力のソースを記述するモード。デジタル出力ピンをhighまたはlowに設定する前に、追加の(時間のかかる)コマンドをmgcに送信する必要があるかどうかを決定するために使用します。

Public Properties

	名前	説明
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Pin	この信号にマップされているデジタルピン。
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

MgcDigitalSignal.DigitalOutMode Property

デジタル出力のソースを記述するモード。デジタル出力ピンをhighまたはlowに設定する前に、追加の(時間のかかる)コマンドをmgcに送信する必要があるかどうかを決定するために使用します。

C++

```
[JsonProperty]
__property DigitalOutMode DigitalOutMode;
```

C#

```
[JsonProperty]
internal DigitalOutMode DigitalOutMode;
```

Visual Basic

```
<JsonProperty>
Friend Property DigitalOutMode() As DigitalOutMode
```

MgcDigitalSignal.IndicationScaling Property

-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。

C++

```
__property MgcIndicationScalingValues IndicationScaling;
```

C#

```
public MgcIndicationScalingValues IndicationScaling;
```

Visual Basic

```
Public Property IndicationScaling() As MgcIndicationScalingValues
```

MgcDigitalSignal.Pin Property

この信号にマップされているデジタルピン。

C++

```
[JsonProperty]
__property int Pin;
```

C#

```
[JsonProperty]
public int Pin;
```

Visual Basic

```
<JsonProperty>
Public Property Pin() As Integer
```

MgcDigitalSignal.Position Property

信号が属するスロット内の位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcDigitalSignal.Slot Property

信号が属するデバイスのスロット番号。

C++

```
__property int Slot;
```

C#

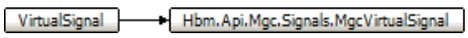
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

MgcVirtualSignal Class

センサを接続しない信号(演算信号など)を表します。



C++

```
class MgcVirtualSignal : public VirtualSignal;
```

C#

```
public class MgcVirtualSignal : VirtualSignal;
```

Visual Basic

```
Public Class MgcVirtualSignal
Inherits VirtualSignal
```

ファイル

MgcVirtualSignal.cs

MgcVirtualSignal Members

次の表は、MgcVirtualSignalによって公開されるメンバーの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

Public Properties

	名前	説明
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Position	アンプ上のコネクタの位置。
	Slot	コネクタが属するスロット。

MgcVirtualSignal Methods

MgcVirtualSignalクラスのメソッドの一覧です。

Public Methods

	名前	説明
	GetUniqueID	信号が属するデバイス内で信号を識別する一意の文字列を返します。

MgcVirtualSignal.GetUniqueID Method

信号が属するデバイス内で信号を識別する一意の文字列を返します。

C++

```
string GetUniqueID();
```

C#

```
public override string GetUniqueID();
```

Visual Basic

```
Public override Function GetUniqueID() As string
```

戻り値

デバイス上の一意の信号識別子。

備考

ストリーミング可能な信号の場合、このメソッドは、ハードウェア モジュール上の信号を識別するためにストリーミングで使用する情報を返す必要があります。この情報は、スキャンまたは接続中に取得する必要があります。

MgcVirtualSignal Properties

MgcVirtualSignalクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Position	アンプ上のコネクタの位置。
	Slot	コネクタが属するスロット。

MgcVirtualSignal.IndicationScaling Property

-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。

C++

```
__property MgcIndicationScalingValues IndicationScaling;
```

C#

```
public MgcIndicationScalingValues IndicationScaling;
```

Visual Basic

```
Public Property IndicationScaling() As MgcIndicationScalingValues
```

MgcVirtualSignal.Position Property

アンプ上のコネクタの位置。

C++

```
__property int Position;
```

C#

```
public int Position;
```

Visual Basic

```
Public Property Position() As Integer
```

MgcVirtualSignal.Slot Property

コネクタが属するスロット。

C++

```
__property int Slot;
```

C#

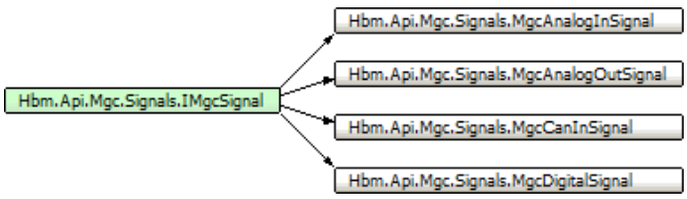
```
public int Slot;
```

Visual Basic

```
Public Property Slot() As Integer
```

IMgcSignal Interface

MgcSignalの追加プロパティのインタフェース。



C++

```
__interface IMgcSignal;
```

C#

```
public interface IMgcSignal;
```

Visual Basic

```
Public Interface IMgcSignal
```

ファイル

IMgcSignal.cs

IMgcSignal Members

次の表は、IMgcSignalによって公開されるメンバーの一覧です。

Public Properties

	名前	説明
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

IMgcSignal Properties

IMgcSignalクラスのプロパティは次のとおりです。

Public Properties

	名前	説明
	IndicationScaling	-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。
	Position	信号が属するスロット内の位置。
	Slot	信号が属するデバイスのスロット番号。

IMgcSignal.IndicationScaling Property

-7680000から7680000間で受信したバイナリ値に基づいて測定値を計算する値。Measurement value = (binary value/7680000)*InputRange - ZeroPoint 。

C++

```
[JsonProperty]
__property MgcIndicationScalingValues IndicationScaling;
```

C#

```
[JsonProperty]
int MgcIndicationScalingValues IndicationScaling;
```

Visual Basic

```
<JsonProperty>
Property IndicationScaling() As MgcIndicationScalingValues
```

IMgcSignal.Position Property

信号が属するスロット内の位置。

C++

```
[JsonProperty]
__property int Position;
```

C#

```
[JsonProperty]
int Position;
```

Visual Basic

```
<JsonProperty>
Property Position() As Integer
```

IMgcSignal.Slot Property

信号が属するデバイスのスロット番号。

C++

```
[JsonProperty]
__property int Slot;
```

C#

```
[JsonProperty]
int Slot;
```

Visual Basic

```
<JsonProperty>
Property Slot() As Integer
```

www.hbm.com

HBM Test and Measurement
Tel. +49 6151 803-0
Fax +49 6151 803-9100
info@hbm.com

アドバン オートメーション株式会社
Tel. 03-5282-7047
Fax. 03-5282-0808
info-advan@adv-auto.co.jp

measure and predict with confidence

HBM