



Micronet.Co,

マイクロネット
Micronet

INDUSTRIAL REALTIME EDGE COMPUTERS

RT-edge

API リファレンスマニュアル




株式会社マイクロネット

<http://www.mnc.co.jp>

TEL: +81(0)299-90-1733

FAX: +81(0)299-92-8557

本書で使用するマークについて

	ノート：操作方法や手順等の補足情報や注釈を説明しています。
	情報：製品を利用する上で有効な豆知識となる説明をしています。
	警告：製品仕様上注意が必要な事象について説明しています。

目次

1. はじめに	6
1.1. 用語解説	6
2. 開発環境設定	8
2.1. 開発用ファイル	8
2.2. 開発プロジェクト設定	8
INtime(C 言語)アプリケーションにおける方法	9
C#アプリケーションにおける方法	12
3. RT-edge API	14
3.1. 関数リスト	14
3.1.1. フレームワーク API 機能	14
3.1.2. メッセージ通信機能	14
3.1.3. EgTag 関連	14
3.1.4. EgCollector 関連	16
3.1.5. EgDataset 関連	16
3.1.6. EgService 関連	16
3.1.7. EgTagTrigger 関連	16
3.1.8. 保守関連	17
3.1.9. その他	17
3.2. API 関数(C/C++)	18
EgInit	18
EgFinalize	20
EgMailboxCreateEx	21
EgMailboxDelete	23
EgMailboxOpen	24
EgMailboxOpenSC	25
EgMailboxClose	26
EgMailboxSend	27
EgMailboxReceive	28
EgMailboxGetInfo	29
EgMailboxGetProperty	31
EgMailboxGetPropertyByIndex	32
EgMailboxGetCount	33
EgMailBoxGetEnum	34
EgTagCreateEx	35
EgTagCreateSp	37
EgTagWrite	39
EgTagRead	40
EgTagGetProperty	41
EgTagGetCount	43
EgTagGetEnum	44
EgTagWriteByIndex	45
EgTagReadByIndex	46
EgTagGetPropertyByIndex	47
EgTagWriteByEntry	49
EgTagReadByEntry	50
EgTagCreateSegment	51

EgTagWriteSegment.....	53
EgTagReadSegment	54
EgTagWriteSegmentByIndex	55
EgTagReadSegmentByIndex	56
EgTagWriteSegmentByEntry	57
EgTagReadSegmentByEntry	58
EgTagCreateFIFO	59
EgTagEnqueueFIFO	61
EgTagEnqueueFIFOByIndex	62
EgTagEnqueueFIFOByEntry	63
EgTagEnqueueRingFIFO.....	64
EgTagEnqueueRingFIFOByIndex	65
EgTagEnqueueRingFIFOByEntry	66
EgTagDequeueFIFO.....	67
EgTagDequeueFIFOByIndex	68
EgTagDequeueFIFOByEntry	69
EgTagClearFIFO	70
EgTagClearFIFOByIndex	71
EgTagClearFIFOByEntry	72
EgTagGetRemainFIFO.....	73
EgTagGetRemainFIFOByIndex	74
EgTagGetRemainFIFOByEntry	75
EgTagSetDefaultValue	76
EgTagSetDefaultValueByGroup	77
EgCollectorGetCount	78
EgCollectorGetEnum	79
EgCollectorGetProperty	80
EgCollectorGetPropertyByIndex	81
EgDatasetCreate	82
EgDatasetAddTag	83
EgDatasetGetSize.....	84
EgDatasetGetBinary	85
EgDatasetGetFirst	86
EgDatasetGetNext.....	87
EgDatasetGetCount	88
EgDatasetGetEnum	89
EgDatasetGetPropertyByIndex	90
EgServiceGetCount.....	91
EgServiceGetEnum	92
EgServiceGetPropertyByIndex	93
EgFW_AddTagTrigger.....	94
EgDecode	95
EgEncode.....	96
EgMakeAddress	97
EgParseAddress	98
3.3. API 関数(.Net).....	99
edge_API Class.....	99
EgInit	100
EgFinalize	102
EgMailboxCreateEx.....	103
EgMailboxDelete	105

EgMailboxOpen	106
EgMailboxClose	106
EgMailboxSend	107
EgMailboxSendSC	108
EgMailboxReceive	109
EgMailboxGetInfo	109
EgMailboxGetProperty	110
EgMailboxGetPropertyByIndex	111
EgMailboxGetCount	112
EgMailBoxGetEnum	113
EgTagCreateEx	114
EgTagCreateSp	116
EgTagWrite	118
EgTagRead	119
EgTagGetProperty	120
EgTagGetCount	121
EgTagGetEnum	122
EgTagWriteByIndex	123
EgTagReadByIndex	124
EgTagGetPropertyByIndex	125
EgTagWriteByEntry	127
EgTagReadByEntry	128
EgTagCreateSegment	129
EgTagWriteSegment	131
EgTagReadSegment	132
EgTagWriteSegmentByIndex	133
EgTagReadSegmentByIndex	134
EgTagWriteSegmentByEntry	135
EgTagReadSegmentByEntry	136
EgTagCreateFIFO	137
EgTagEnqueueFIFO	139
EgTagEnqueueFIFOByIndex	140
EgTagEnqueueFIFOByEntry	141
EgTagEnqueueRingFIFO	142
EgTagEnqueueRingFIFOByIndex	143
EgTagEnqueueRingFIFOByEntry	144
EgTagDequeueFIFO	145
EgTagDequeueFIFOByIndex	146
EgTagDequeueFIFOByEntry	147
EgTagClearFIFO	148
EgTagClearFIFOByIndex	149
EgTagClearFIFOByEntry	150
EgTagGetRemainFIFO	151
EgTagGetRemainFIFOByIndex	152
EgTagGetRemainFIFOByEntry	153
EgTagSetDefaultValue	154
EgTagSetDefaultValueByGroup	155
EgCollectorGetCount	156
EgCollectorGetEnum	157
EgCollectorGetProperty	158
EgCollectorGetPropertyByIndex	159
EgDatasetCreate	160

EgDatasetAddTag	161
EgDatasetGetSize.....	161
EgDatasetGetBinary	161
EgDatasetGetFirst	162
EgDatasetGetNext.....	163
EgDatasetGetCount	164
EgDatasetGetEnum	165
EgDatasetGetPropertyByIndex	166
EgServiceGetCount.....	167
EgServiceGetEnum	168
EgServiceGetPropertyByIndex	169
EgFW_AddTagTrigger	170
EgGatherExecute	171
EgMDiagExecute_PfmCntAdd.....	172
EgMDiagExecute_PfmCntGet	173
EgMDiagExecute_DiskInfo	174
EgMDiagGetSMARTDiskName	175
EgMDiagGetSMARTValue.....	176
EgDecode	177
EgEncode.....	178
EgMakeAddress	179
EgParseAddress	180
3.4. エラーコードリスト.....	181
4. 付録.....	185
4.1. メッセージ通信用システムメッセージ一覧.....	185
4.2. Address 表記	187
4.3. その他構造体	188

1. はじめに

本マニュアルでは、RT-edge API リファレンスについて説明しています。

その他の説明に関しては以下マニュアルを参照してください。

表 1 関連マニュアル

名称	ファイル名	内容
RT-edge ユーザーズマニュアル	DOCRTEEDGEUSR.pdf	RT-edge システム全般的な内容・基本仕様の説明が記載されています。
RT-edge サービス作成マニュアル	DOCRTEDEGESRV.pdf	RT-edge サービスの作成方法、サービス作成用テンプレート、サンプルシステムについて説明が記載されています。
インストール手順書	インストール手順書.pdf	RT-edge 実行/開発環境のインストール手順が記載されています。

1.1. 用語解説

本ドキュメントにおいて使用される用語・略称について説明します：

表 2 用語集

用語	説明
INtime	INtime for Windows: Windows と協調動作可能なリアルタイムカーネル拡張ソフトウェアです。 INtime Distributed RTOS(dRTOS): Windows OS を必要とせず、スタンドアロンで動作するリアルタイム OS です。
RTA	RealTime Application : リアルタイムアプリケーションの略称。INtime 上で動作するローダブルプロセスの拡張子です。INtime 上で動作するローダブルアプリケーションは、RTA という拡張子を持ちます。
RSL	Realtime Shared Library : リアルタイム共有ライブラリの略称。INtime 上でアプリケーションがロード可能なライブラリです。Windows 上で使用される DLL(Dynamic Link Library)のようなものです。RTA から使用されるライブラリインタフェース等は、こちらを使用して作成することができます。
API	Application Programming Interface : アプリケーションプログラミングインタフェースの略称。RT-edge ではデバイスへのアクセスインタフェースとして API ライブラリを提供しています。
NTX	INtime's Windows NT extension API: INtime 用 Windows NT 拡張 API の略称。 NTX 関数は Windows プログラムが INtime リアルタイム環境上で実行するリアルタイムプログラムと通信を可能とする関数セットです。
サービスコンテナ/EgService	RT-edge システムを構成する機能プロセス(rta/exe)です。
タグ/EgTag	瞬時値データ値 1 つを示すオブジェクトです。ユニーク名とグローバルなスコープを持ち、全ての EgService から読み書きが許されたオブジェクトです。タグは生成時にデータ型が確定され変更はできません。
リンクタグ	同一名称のタグを重複生成した場合に自動的に別名称で生成されるタグを指します。通常のタグと同様、グローバルなスコープを持ち、全ての EgService から読み書きが許されたオブジェクトです。一つのタグに対し、異なるプロパティ情報を定義したい場合に使用します。

用語	説明
データセット/EgDataset	タグ 1 つ以上の組み合わせでデータ並び順(データ構造)を定義する名前付きオブジェクトです。
コレクタ/EgCollector	データセットに定義されたデータ構造に従って、同時刻のバイナリデータ列で生成し、データレコードとしてメールボックスに送信するオブジェクト (スレッド) です。
メールボックス/EgMailBox	時系列なデータセット、または時系列メッセージを FIFO で蓄えることができ、また受信イベントとして処理できるオブジェクトです。
タグ参照/TagRef	タグの参照として使用するオブジェクトです。タグの名前を保持し値は保持しません。サービスコンフィグファイルでデータセットの収集用タグとして定義することや、サービスコンテナ内のオブジェクトとして定義することでサービスコンテナのメンバ変数として使用することができます。
コレクタ参照/CollectorRef	コレクタの参照として使用するオブジェクトです。コレクタの名前を保持しそれ以外のオブジェクトは保持しません。サービスコンフィグファイルでサービスコンテナ内のオブジェクトとして定義することでサービスコンテナ内のメンバ変数として使用することができます。
タグトリガー/TagTrigger	タグトリガーとは特定のタグの値が変化した場合に、サービスコンテナ側でメッセージ通知を受けることができる機能です。サービスコンテナの ECI ファイルでタグトリガーとして登録したいタグ名を記載することで、タグ値の変更通知を受け取ることができます。
メッセージ	メールボックスで扱われる 1 レコード分のデータ、またはサービスコンテナ間のコマンド、応答の電文です。
フレームワーク	フレームワークは、アプリケーションが API を組み合わせて実装するよくある処理についてマクロ化、自動化したものでサービスコンフィグファイルの記述により自動処理させることができます。
サービスコンフィグファイル	RT-edge フレームワークが、タグやメールボックスなどの RT-edge オブジェクトの生成やサービスコンテナの起動を自動処理するための定義を記述した XML 形式のファイルです。
入力	RT-edge システムを中心に見た場合、外部の情報を RT-edge システムへ取り込む方向性のデータの流れを意味します。
出力	RT-edge システムを中心に見た場合、RT-edge システムが持つデータを外部に書き出す方向性のデータの流れを意味します。
RTCD	Realtime Common Data の略称。RT-edge システム上で最もベースとなる共有データ構造機能です。
RT-edge Object	RT-edge システム上で使用可能なオブジェクト群 (機能群) の総称です。 例えば、センサーや装置から収集したデータをアプリケーション間で受け渡しを行う場合に使用するタグ、アプリケーション間でメッセージのやり取りを行う場合のメールボックス等、アプリケーション間でデータの受け渡しを行うケースにおいて利用されるオブジェクトです。 RT-edge Object は Windows アプリケーション間、INtime®アプリケーション間、Windows-INtime®アプリケーション間いずれの場合も利用可能です。
S.M.A.R.T.情報	Self-Monitoring Analysis and Reporting Technology の略で、ハードディスクドライブ (HDD) やソリッドステートドライブ (SSD) などのストレージデバイスに内蔵されている自己診断機能のこと。
blittable 型	.NET Framework で扱うデータ型のうち、マネージ側(C#実装コード上など)とネイティブ側(C/C++実装コード上など)とで内部表現が同じになるものを表します。 具体的には、char を除く任意の整数型・浮動小数点型、列挙型と、前述の型のみを含む構造体・レイアウト指定されたクラス・1 次元固定長配列が該当します。

2. 開発環境設定

2.1. 開発用ファイル

製品セットアップで作成された以下開発環境コンポーネントを使用します。



必要に応じて開発プロジェクト配下にコピー、または下記展開ディレクトリを環境変数に登録し、参照できるようにします。本マニュアルではCドライブ直下に開発環境コンポーネントを展開し、下記の環境変数を登録した例を記載します。

環境変数 : 「RTEDGE」 パス : 「C¥RT-edge¥」

表 3 開発環境コンポーネント

フォルダ 階層	ファイル名	説明	使用有無		
			共通	RT	WIN
RT-edge¥Library¥	edgeAPI_RTCD.h	RT-edge 開発環境 API ヘッダ	○		
RT-edge¥Library¥	egAPI.h	RT-edge 開発環境 API ヘッダ	○		
RT-edge¥Library¥RT¥	eghgapi.lib	RT-edge 開発環境 API ライブラリ(INTime)		○	
RT-edge¥Library¥RT¥	egosdep.lib	RT-edge 開発環境 API ライブラリ(INTime)		○	
RT-edge¥Library¥RT¥	egRTCD.lib	RT-edge 開発環境 API ライブラリ(INTime)		○	
RT-edge¥Library¥WIN¥	eghgapiWin.lib	RT-edge 開発環境 API ライブラリ(Windows)	○		
RT-edge¥Library¥WIN¥	egosdepntx.lib	RT-edge 開発環境 API ライブラリ(Windows)		○	
RT-edge¥Library¥WIN¥	egosdepwin.lib	RT-edge 開発環境 API ライブラリ(Windows)			○
RT-edge¥Library¥WIN¥	egRTCDntx.lib	RT-edge 開発環境 API ライブラリ(Windows)		○	
RT-edge¥Library¥WIN¥	egRTCDwin.lib	RT-edge 開発環境 API ライブラリ(Windows)			○

・ RT : INtime 環境で使用

・ WIN : Windows (INTime 未インストール) 環境で使用

2.2. 開発プロジェクト設定

RT-edge 開発ライブラリを使用し RT-edge サービスコンテナを作成する場合、アプリケーションビルド用の開発プロジェクトを生成後、ライブラリリンク設定を行う必要があります。RT-edge サービスコンテナには、従来のコンピュータ言語で記述されるネイティブコード(Windows C/C++言語 MFC, INtime C/C++言語)で作成する方法と.NET Framework 上で動作するオブジェクトコード(C#)で作成する方法があります。以下では、INTime アプリケーション(C言語)を使用した開発プロジェクト、C#(.NET)を使用した開発プロジェクト生成方法について説明します：

INtime(C 言語)アプリケーションにおける方法

以下の手順に沿ってプロジェクト設定をおこなってください（例では Visual Studio 2017 を用いています）：

手順 1

Microsoft Visual Studio を使用し、プロジェクトを作成します。

プロジェクト生成には、INtime Projects を選択後、Application Wizard 等を指定しプロジェクトワークスペースを生成します：

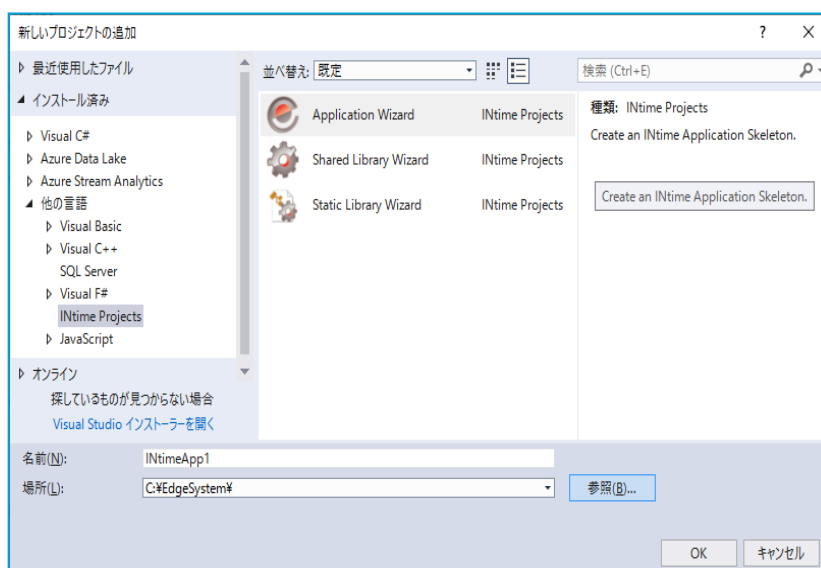


図 1. プロジェクトワークスペース生成(C/C++)

手順 2

ヘッダファイルの参照設定を行います。

[プロジェクト]メニューから[プロパティ]を選択し、プロジェクトのプロパティダイアログを開きます。[構成プロパティ]-[C/C++]-[全般]の[追加のインクルードディレクトリ]に、開発用ヘッダ定義ファイルのパスを指定します。

例： 開発ライブラリのパス（「RTEDGE」が環境変数に登録してある場合）

\$(RTEDGE)Library

[追加のインクルードディレクトリ]欄に直接追加するか、右端リストのボタンから[編集]を選択し、追加のインクルードディレクトリダイアログからパスを設定してください：

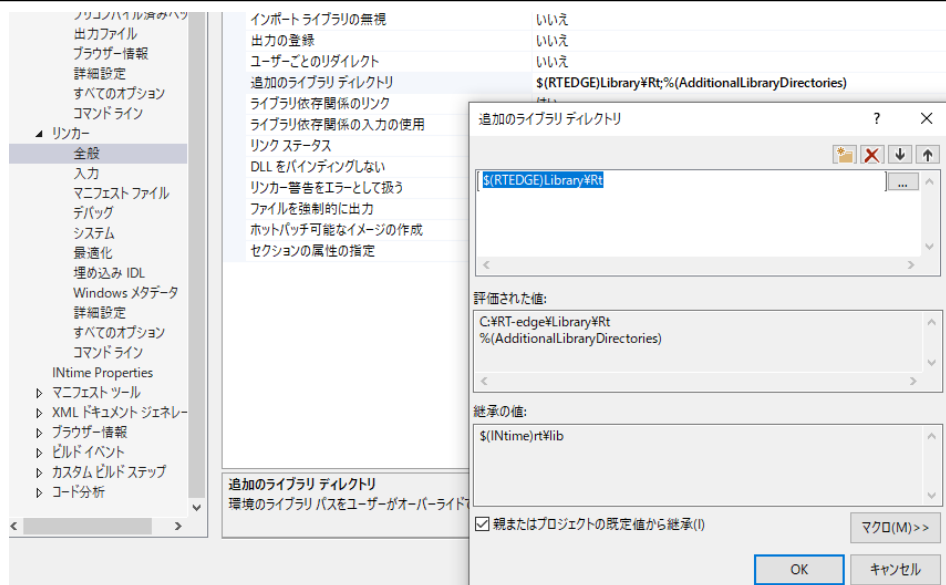


図 2.インクルードディレクトリ設定



参照パスを追加する場合、追加のインクルードディレクトリには、あらかじめウィザードにより設定されている設定情報があります。この設定は消さないでください。ライブラリパスの設定は、最後に追加し、既に設定されているパスの中間に設定したり、先頭に配置したりしないでください。



RT-Edge API を使用する場合、ライブラリ関数定義が行われている egAPI.h をユーザープログラムの先頭で #include してください。

手順 3

リンクライブラリ設定を行います。

ヘッダ参照設定と同様、[プロジェクト]メニューから[プロパティ]を選択し、プロジェクトのプロパティダイアログを開きます。[構成プロパティ]-[リンカー]-[全般]の[追加のライブラリディレクトリ]に、開発リンクライブラリのパスを指定します。

例：（「RTEDGE」が環境変数に登録してある場合）

`$(RTEDGE)Library\Rt`

[追加のライブラリディレクトリ]欄に直接入力するか、右端リストのボタンから[編集]を選択し、追加のライブラリディレクトリダイアログからパスを設定してください：

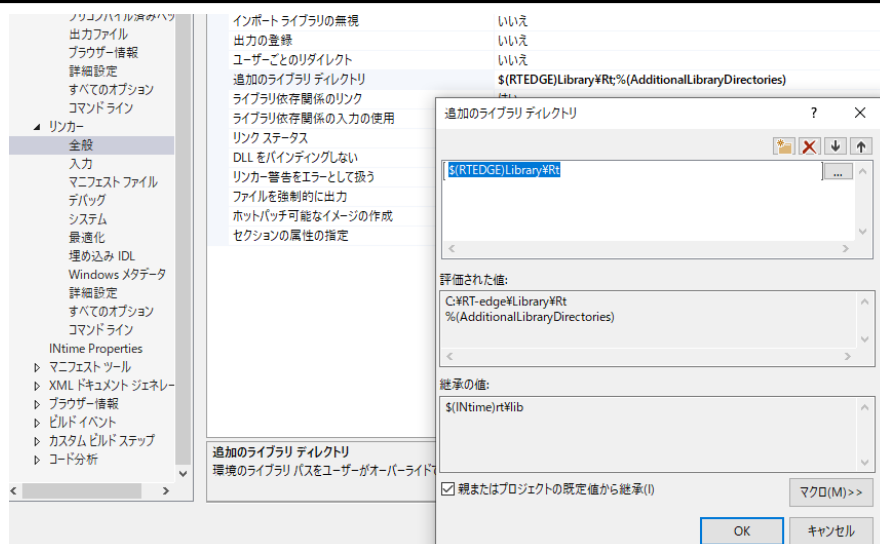


図 3.リンクライブラリ設定



参照パスを追加する場合、追加のライブラリディレクトリには、あらかじめウィザードにより設定されている設定情報があります。この設定は消さないでください。ライブラリパスの設定は、最後に追加し、既に設定されているパスの中間に設定したり、先頭に配置したりしないでください。

次に、**[構成プロパティ]-[リンカー]-[入力]**の**[追加の依存ファイル]**に製品リンクライブラリ (eghgapi.lib)を設定します。



追加の依存ファイル欄には、既に追加されているエントリがあります。リンクライブラリ名を追加する際、先に追加されているエントリを削除せず、最後に追加するようにしてください。

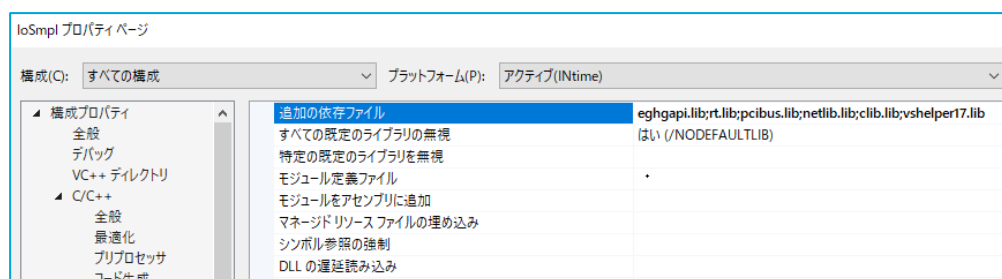


図 4. 追加の依存ファイル入力

C#アプリケーションにおける方法

以下の手順に沿ってプロジェクト設定をおこなってください:

手順 1

Microsoft Visual Studio を使用し、プロジェクトを作成します。Visual C#のプロジェクトから、プロジェクトワークスペースを生成します:

Visual C#

- Windows フォームアプリケーション
- WPF アプリケーション
- コンソールアプリケーション

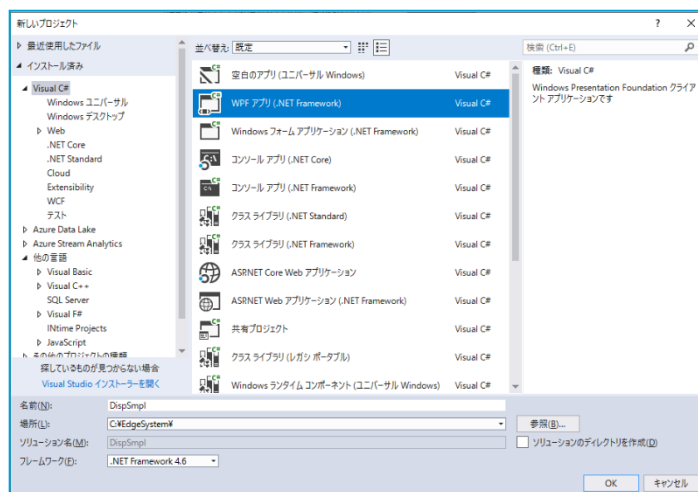


図 5. プロジェクトワークスペース生成(VB#/VB.NET)

手順 2

アセンブリの参照設定を行います。

[プロジェクト]メニューから[参照設定]を選択し、参照マネージャを起動します。

[ブラウズ]-[参照]から、ファイル選択ダイアログにて、インストールを行ったフォルダの「egapiWrap.dll」を選択します。

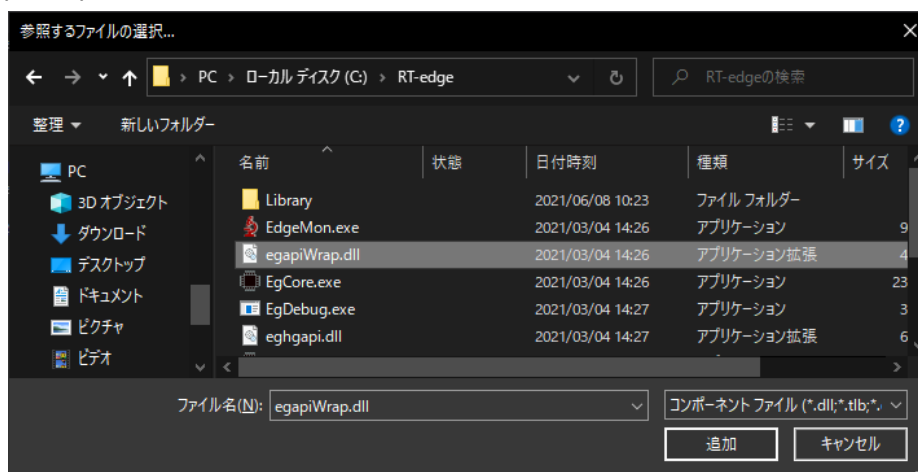


図 6. 参照するアセンブリ選択

手順 3

アセンブリ追加の確認。

ソリューションエクスプローラから、参照設定を開き、追加したアセンブリが表示されていることを確認します。

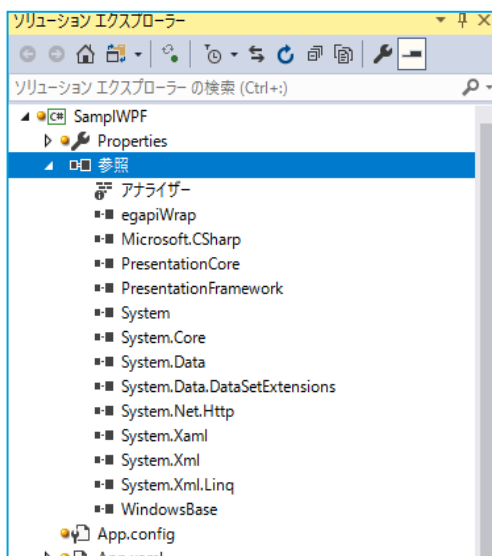


図 7. 参照設定確認

手順 4

プラットフォームターゲットの設定。

プロジェクトのプロパティ設定を開き、プラットフォームターゲットを「x64」とします。

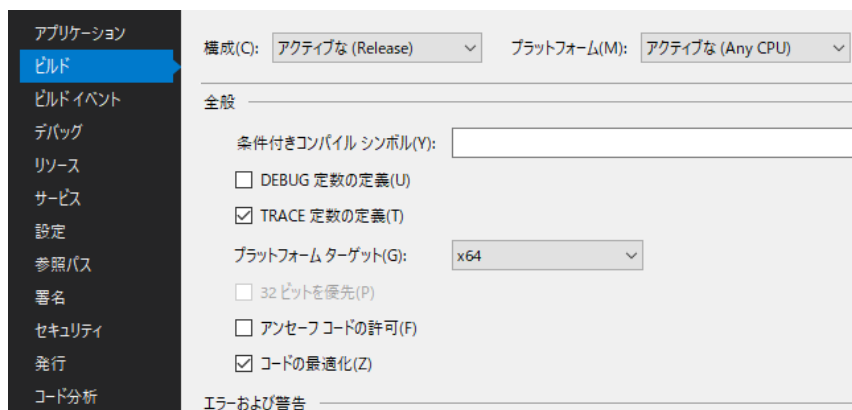


図 8. プラットフォームターゲットの設定

3. RT-edge API

3.1. 関数リスト

以下リストの API を提供します:

3.1.1. フレームワーク API 機能

API 関数	説明	C/C++	.NET
EgInit	Edge フレームワークの初期化・開始処理	○	○
EgFinalize	Edge フレームワーク終了処理	○	○



サービスコンテナ作成方法、フレームワークを用いたサービスコンテナ作成方法は、別マニュアル「RT-edge コンテナ作成マニュアル」を参照してください。

3.1.2. メッセージ通信機能

API 関数	説明	C/C++	.NET
EgMailboxCreateEx	メッセージ通信オブジェクト生成	○	○
EgMailboxDelete	メッセージ通信オブジェクト削除	○	○
EgMailboxOpen	メッセージ通信オブジェクトのオープン	○	×
EgMailboxOpenSC	コンテナ間通信オブジェクトのオープン (サービスコンテナ名指定)	○	×
EgMailboxClose	メッセージ通信オブジェクトのクローズ	○	×
EgMailboxSend	メッセージ送信	○	○
EgMailboxSendSC	コンテナ間通信オブジェクト専用メッセージ送信	×	○
EgMailboxReceive	メッセージ受信	○	○
EgMailboxGetInfo	メッセージ通信オブジェクト情報の取得	○	○
EgMailboxGetProperty	EgMailbox のプロパティ取得	○	○
EgMailboxGetPropertyByIndex	EgMailbox のプロパティ取得(Index 指定)	○	○
EgMailboxGetCount	EgMailbox の登録数取得	○	○
EgMailBoxGetEnum	EgMailbox の一覧取得	○	○

3.1.3. EgTag 関連

API 関数	説明	C/C++	.NET
EgTagCreateEx	EgTag を作成する	○	○
EgTagCreateSp	EgTag を作成する(タグ名重複対応版)	○	○
EgTagWrite	EgTag にデータを書き込む	○	○

API 関数	説明	C/C++	.NET
EgTagRead	EgTag のデータを読み込む	○	○
EgTagGetProperty	EgTag の Property を取得	○	○
EgTagGetCount	EgTag の登録数取得	○	○
EgTagGetEnum	EgTag の一覧取得	○	○
EgTagWriteByIndex	EgTag にデータを書き込む(Index 指定)	○	○
EgTagReadByIndex	EgTag のデータを読み込む(Index 指定)	○	○
EgTagGetPropertyByIndex	EgTag の Property を取得(Index 指定)	○	○
EgTagWriteByEntry	EgTag にデータを書き込む(Entry 指定)	○	○
EgTagReadByEntry	EgTag のデータを読み込む(Entry 指定)	○	○
EgTagCreateSegment	EgTag を作成する(文字列、バイト配列対応版)	○	○
EgTagWriteSegment	EgTag に配列データを書き込む	○	○
EgTagReadSegment	EgTag の配列データを読み込む	○	○
EgTagWriteSegmentByIndex	EgTag に配列データを書き込む(Index 指定)	○	○
EgTagReadSegmentByIndex	EgTag の配列データを読み込む(Index 指定)	○	○
EgTagWriteSegmentByEntry	EgTag に配列データを書き込む(Entry 指定)	○	○
EgTagReadSegmentByEntry	EgTag の配列データを読み込む(Entry 指定)	○	○
EgTagCreateFIFO	EgTag を作成する(FIFO 対応版)	○	○
EgTagEnqueueFIFO	EgTag(FIFO)にデータ登録	○	○
EgTagEnqueueFIFOByIndex	EgTag(FIFO)にデータ登録(Index 指定)	○	○
EgTagEnqueueFIFOByEntry	EgTag(FIFO)にデータ登録(Entry 指定)	○	○
EgTagEnqueueRingFIFO	EgTag(FIFO)にデータ登録(Ring 動作)	○	○
EgTagEnqueueRingFIFOByIndex	EgTag(FIFO)にデータ登録(Ring/Index 指定)	○	○
EgTagEnqueueRingFIFOByEntry	EgTag(FIFO)にデータ登録(Ring/Entry 指定)	○	○
EgTagDequeueFIFO	EgTag(FIFO)からデータ取得	○	○
EgTagDequeueFIFOByIndex	EgTag(FIFO)からデータ取得(Index 指定)	○	○
EgTagDequeueFIFOByEntry	EgTag(FIFO)からデータ取得(Entry 指定)	○	○
EgTagClearFIFO	EgTag(FIFO)上のデータ消去	○	○
EgTagClearFIFOByIndex	EgTag(FIFO)上のデータ消去(Entry 指定)	○	○
EgTagClearFIFOByEntry	EgTag(FIFO)上のデータ消去(Index 指定)	○	○
EgTagGetRemainFIFO	EgTag(FIFO)上のデータ数取得	○	○
EgTagGetRemainFIFOByIndex	EgTag(FIFO)上のデータ数取得(Entry 指定)	○	○
EgTagGetRemainFIFOByEntry	EgTag(FIFO)上のデータ数取得(Index 指定)	○	○
EgTagSetDefaultValue	Tag の値を ECI で指定した初期値に更新(ECI	○	○

API 関数	説明	C/C++	.NET
	で定義した全 Tag)		
EgTagSetDefaultValueByGroup	Tag の値を ECI で指定した初期値に更新(ECI で InitGroup を指定した Tag)	○	○

3.1.4. EgCollector 関連

API 関数	説明	C/C++	.NET
EgCollectorGetProperty	EgCollector の Property を取得	○	○
EgCollectorGetCount	EgCollector の登録数取得	○	○
EgCollectorGetEnum	EgCollector の一覧取得	○	○
EgCollectorGetPropertyByIndex	EgCollector の Property を取得(Index 指定)	○	○

3.1.5. EgDataset 関連

API 関数	説明	C/C++	.NET
EgDatasetCreate	EgDataset を生成する	○	○
EgDatasetAddTag	EgDataset に EgTag を追加する	○	○
EgDatasetGetSize	EgDataset 内の全ての Tag の Size を加算し、1 レコードのバイナリバイトサイズを返す	○	×
EgDatasetGetBinary	EgDataset 内の全ての Tag の Value を取得し、1 レコードのバイナリ値として連結作成	○	×
EgDatasetGetFirst	EgDataset の最初のタグを取得	○	○
EgDatasetGetNext	EgDataset の次のタグを取得	○	○
EgDatasetGetCount	EgDataset の登録数取得	○	○
EgDatasetGetEnum	EgDataset の一覧取得	○	○
EgDatasetGetPropertyByIndex	EgDataset の Property を取得(Index 指定)	○	○

3.1.6. EgService 関連

API 関数	説明	C/C++	.NET
EgServiceGetCount	EgService の登録数取得	○	○
EgServiceGetEnum	EgService の一覧取得	○	○
EgServiceGetPropertyByIndex	EgService の Property を取得(Index 指定)	○	○

3.1.7. EgTagTrigger 関連

API 関数	説明	C/C++	.NET
EgFW_AddTagTrigger	サービスコンテナに EgTagTrigger を登録	○	○

3.1.8. 保守関連

API 関数	説明	C/C++	.NET
EgGatherExecute	【収集】 リストに基づいたログなどのファイルを指定のフォルダへコピー後、zip 化する	×	○
EgMDiagExecute_PfmCntAdd	【診断・通知】 パフォーマンスカウンター追加	×	○
EgMDiagExecute_PfmCntGet	【診断・通知】 パフォーマンスカウンター値を取得	×	○
EgMDiagExecute_DiskInfo	【診断・通知】 S.M.A.R.T.情報を取得	×	○
EgMDiagGetSMARTDiskName	【診断・通知】 「EgMDiagExecute_DiskInfo」で取得した S.M.A.R.T 情報から指定のディスクのモデル名を取得	×	○
EgMDiagGetSMARTValue	【診断・通知】 「EgMDiagExecute_DiskInfo」で取得した S.M.A.R.T 情報から指定のディスク・検査項目 ID に対応する現在値、ステータス値を取得	×	○

3.1.9. その他

API 関数	説明	C/C++	.NET
EgDecode	バイナリデータ(バイト配列)から指定のフォーマットに基づいた値を取得する	○	○
EgEncode	指定のフォーマットに基づいてバイナリデータ列の所定位置にデータを書き出す	○	○
EgMakeAddress	オフセット値・サイズからアドレス情報を作成する	○	○
EgParseAddress	アドレス情報からオフセットとサイズに分解する	○	○

3.2. API 関数(C/C++)

EgInit

API ライブラリの初期化をします:

シンタックス

```
int32_t EgInit ( EDGECONFIG *Config );
```

引 数

***Config** [IN] edge フレームワークの動作設定構造体
デフォルト設定で良い場合は NULL、または EDGE_CONFIG_DEFAULT を使用します。

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

EDGECONFIG 構造体

```
typedef struct tEdgeConfig
{
    uint8_t      bDebug;                // デバッグフラグ(デフォルト false)
    uint8_t      bReadXmlF;            // サービスコンフィグファイルを
                                        // 0:読まない, 1:読む(デフォルト)
    uint16_t     shPriority;            // メールボックススレッド プライオリティ
                                        // (デフォルト 150)
    FUNC_POINTER fpUserMessHdlFunc;    // ユーザメッセージハンドラ関数
    char         Path[EG_MAX_SERVICEPATH+1]; // サービスプロセスファイルパス
    char         _nil00;               // 予約
```

EDGECONFIG 構造体 フィールド

型	シンボル	内容
uint8_t	bDebug	デバッグフラグ(デフォルト false)
uint8_t	bReadXmlF	(内部で使用する為使用不可)
uint16_t	shPriority	サービスコンテナ間通信用メールボックスのプライオリティ指定 (1~254)(デフォルト 150)
FUNC_POINTER	fpUserMessHdlFunc	ユーザメッセージハンドラコールバック関数
char	Path	サービスプロセスファイルパス
char	_nil00	(予約 使用不可)

ユーザメッセージハンドラ(コールバック関数)指定

コールバック関数とは、他の関数に引数として渡される関数で、外側の関数で何らかの処理やアクションを実行します。このコールバック関数の原理を利用して、ユーザメッセージハンドラ関数(ユーザ定義関数)を EDGECONFIG 構造体変数に指定します。ユーザメッセージハンドラの作成方法については、「RT-edge コンテナ作成マニュアル」を参照してください。

コールサンプル

EgInit コール例

```
int32_t    result;
EDGECONFIG config = EDGE_CONFIG_DEFAULT;
config.fpUserMessHdlFunc= myMsgHandler;
result = EgInit( &config );
if( result == EDGE_SUCCESS )
    printf("成功");
```

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- EgInit を使用したサービスプロセスは終了時に EgFinalize をコールする必要があります。

EgFinalize

edge フレームワークを終了します:

シンタックス

```
int32_t EgFinalize ( void );
```

引 数

なし

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_FW_INIT_ERROR	0x80001001	フレームワークとして動作不可

コールサンプル

EgFinalize コール例

```
int32_t result;
EDGECONFIG config = EDGE_CONFIG_DEFAULT;
result = EgInit( &config );
:
result = EgFinalize( );
```

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- EgFinalize は edge フレームワーク処理スレッドを削除します。このため edge フレームワークが処理を行っている edge メッセージハンドラ関数内で EgFinalize をコールすることはできませんのでご注意ください。edge メッセージハンドラ処理が停止してしまいます。この関数はアプリケーションの終了時にコールされる必要があります。

EgMailboxCreateEx

EgMailBox を生成します:

シンタックス

```
void EgMailboxCreateEx ( const char *pName,
                        uint32_t dwColSize,
                        uint32_t dwRowSize,
                        RTCDMEMINFO *pRtcdInfo );
```

引 数

*pName	[IN] EgMailBox の名前(半角英数字 8 文字まで)
dwColSize	[IN] レコード長(バイト) 設定範囲:1 以上
dwRowSize	[IN] レコード数 設定範囲:2 以上
*pRtcdInfo	[OUT] メールボックスメモリ情報

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

RTCDMEMINFO 構造体

```
typedef union
{
    struct {
        BYTE *pRtcd;           // メモリポインタ(本体)
        OBJHANDLE mem_handle;   // メモリハンドル
        OBJHANDLE sem1_handle;  // 排他用セマフォハンドル
        OBJHANDLE sem2_handle;  // 書込通知用セマフォハンドル
        char name[12];          // 名称(最大 10 文字まで)
    };
    BYTE reserve[28];
} EDGE_RTCD_MEM_INFO, RTCDMEMINFO;
```

RTCDMEMINFO 構造体 フィールド

型	シンボル	内容
BYTE*	pRtcd	メモリポインタ
OBJHANDLE	mem_handle	(内部で使用 メモリハンドル)
OBJHANDLE	sem1_handle	(内部で使用 排他用セマフォハンドル)
OBJHANDLE	sem2_handle	(内部で使用 書込通知用セマフォハンドル)
char	Name	名称(最大 10 文字まで)

OBJHANDLE は unsigned long long と同様です。

コールサンプル

EgMailboxCreateEx コール例

```
int32_t result;  
RTCDMEMINFO rtcdi = {0};  
result = EgMailboxCreateEx( "myMbox" ,1024 ,1024 ,&rtcdi );  
if ( result == EDGE_SUCCESS )  
    printf("成功");
```

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgMailboxDelete

EgMailBox を削除します:

シンタックス

```
int32_t EgMailboxDelete ( RTCDMEMINFO *pRtcdInfo );
```

引 数

***pRtcdInfo** [IN] メールボックスメモリ情報
(EgMailboxCreateEx にて取得したメモリ情報構造体)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_FW_INIT_ERROR	0x80001001	フレームワークとして動作不可

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxDelete コール例

```
int32_t result;  
RTCDMEMINFO rtcdi = {0};  
result = EgMailboxCreateEx( "myMbox" ,1024 ,1024 ,&rtcdi );  
:  
result = EgMailboxDelete( &rtcdi );
```


EgMailboxOpen

EgMailBox をオープンします:

シンタックス

```
int32_t EgMailboxOpen ( const char *pName
                        uint32_t   dwAccess,
                        RTCDHANDLE *pHandle );
```

引 数

***pName** [IN] EgMailBox の名前(半角英数字 8 文字まで)

dwAccess [IN] EgMailBox にアクセスするモードを指定

定数	値	説明
RTCD_READ	0x0001	読込アクセス
RTCD_WRITE	0x0002	書込みアクセス
RTCD_READWRITE	0x0003	読書きアクセス

***pHandle** [OUT] EgmailBox ハンドル構造体
成功すると本構造体に有効なデータを返す

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxOpen コール例

```
int32_t result;
RTCDMEMINFO rtdi = {0};
RTCDHANDLE hMbox;
result = EgMailboxCreateEx( "myMbox" ,1024 ,1024 ,&rtdi );
:
result = EgMailboxOpen( "myMbox" ,RTCD_WRITE ,&hMbox);
if( result == EDGE_SUCCESS )
    printf("成功");
```

EgMailboxOpenSC

コンテナ間通信用 EgMailBox をオープンします:

シンタックス

```
int32_t EgMailboxOpenSC ( const char *pName
                          uint32_t   dwAccess,
                          RTCDHANDLE *pHandle );
```

引 数

***pName**

[IN] サービスコンテナ名

dwAccess

[IN] EgMailBox にアクセスするモードを指定

定数	値	説明
RTCD_READ	0x0001	読込アクセス
RTCD_WRITE	0x0002	書込みアクセス
RTCD_READWRITE	0x0003	読書きアクセス

***pHandle**

[OUT] EgmailBox ハンドル構造体

成功すると本構造体に有効なデータを返す

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxOpen コール例

```
int32_t result;
RTCDMEMINFO rtcdi = {0};
RTCDHANDLE hMbox;
result = EgMailboxOpen( "ServiceName" ,RTCD_WRITE ,&hMbox);
if( result == EDGE_SUCCESS )
    printf("成功");
```

EgMailboxClose

EgMailBox をクローズします:

シンタックス

```
int32_t EgMailboxClose( RTCDHANDLE *pHandle );
```

引 数

***pHandle** [IN] EgMailBox ハンドル

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxClose コール例

```
int32_t result;
RTCDMEMINFO rtcdi = {0};
RTCDHANDLE hMbox;
result = EgMailboxCreateEx( "myMbox", 1024, 1024, &rtcdi );
:
result = EgMailboxOpen( "myMbox", RTCD_WRITE, &hMbox );
:
result = EgMailboxClose( &hMbox );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgMailboxSend

EgMailBox にメッセージを送信します。他サービスコンテナにメッセージを送る場合に使用します:

シンタックス

```
int32_t EgMailboxSend( RTCDHANDLE *pHandle,
                      const char *SenderName,
                      int32_t MessageNo,
                      void *pData,
                      uint32_t dwSize);
```

引 数

*pHandle	[IN] EgMailBox ハンドル
*SenderName	[IN] 送信 EgMailBox 名
MessageNo	[IN] 送信するメッセージ番号
*pData	[IN] 送信するメッセージ引数・データ等
dwSize	[IN] 送信するデータサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxSend コール例

```
int32_t result;
RTCDMEMINFO rtcdi = {0};
RTCDHANDLE hMbox;
result = EgMailboxCreateEx( "tgtServ", 1024, 1024, &rtcdi );
:
result = EgMailboxOpen( "tgtServ", RTCD_WRITE, &hMbox );
:
BYTE data[] = { 3, 4, 3, 5, 3, 128 };
result = EgMailboxSend( &hMbox, "myServ", 10000, data, sizeof(data));
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgMailboxReceive

EgMailBox からメッセージを受信します:

シンタックス

```
int32_t EgMailboxReceive( RTCDHANDLE *pHandle,
                          const char *SenderName,
                          int32_t MessageNo,
                          void *pData,
                          uint32_t dwTimeout);
```

引 数

*pHandle	[IN] 受信用 EgMailBox ハンドル
*SenderName	[OUT] 送信元 EgMailBox 名が格納されるバッファ
MessageNo	[OUT] 受信メッセージ番号が格納される場バッファ
*pData	[OUT] 受信メッセージデータが格納されるバッファ
dwTimeout	[IN] タイムアウト指定(ミリ秒)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxReceive コール例

```
int32_t result;
RTCDMEMINFO rtdi = {0};
RTCDHANDLE hMbox;
result = EgMailboxCreateEx( "myMbox", 1024, 1024, &rtdi );
:
result = EgMailboxOpen( "myMbox", RTCD_READ, &hMbox);    // read mode
:
BYTE *data = (BYTE*)malloc(1024);
char *from = (char*)malloc(9);
int32_t MsgNo;
uint32_t Timeout = 5000;
result = EgMailboxReceive( &hMbox, from, &MsgNo, data, Timeout );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgMailboxGetInfo

EgMailBox 情報を取得します:

シンタックス

```
int32_t EgMailboxGetInfo( RTCDHANDLE      *pHandle,
                          EDGE_RTCD_HEADER *pInfo);
```

引 数

***pHandle** [IN] 受信用 EgMailBox ハンドル

***pInfo** [OUT] EgMailbox ヘッダ情報構造体

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxGetInfo コール例

```
int32_t result;
RTCDMEMINFO rtcdi = {0};
RTCDHANDLE hMbox;
result = EgMailboxCreateEx( "tgtServ", 1024, 1024, &rtcdi );
:
result = EgMailboxOpen( "tgtServ", RTCD_WRITE, &hMbox );
:
EDGE_RTCD_HEADER info;
result = EgMailboxGetInfo( &hMbox, &info );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EDGE_RTCD_HEADER 構造体

```
typedef union
{
    struct {
        DWORD record_num;           // レコード数
        DWORD record_size;          // レコードサイズ(バイト数)
        DWORD read_pos;             // 読み位置
        DWORD write_pos;            // 書き位置
        DWORD write_mode;           // 書きモード 1:シングル 2:マルチ
        DWORD read_open_count;      // 読みモードオープンカウンター
        DWORD write_open_count;     // 書きモードオープンカウンター
        DWORD lost_count;           // 欠落カウンター
        OBJHANDLE mem_handle;       // メモリハンドル値
        OBJHANDLE lock_sem_handle;  // 排他用セマフォハンドル値
        OBJHANDLE notify_sem_handle; // 排他用セマフォハンドル値
        char name[12];              // 名称(最大 10 文字まで)
    };
    BYTE reserve[28];
} EDGE_RTCD_HEADER;
```

RTCDMEMINFO 構造体 フィールド

型	シンボル	内容
DWORD	record_num	レコード数
DWORD	record_size	レコードサイズ(バイト数)
DWORD	read_pos	(内部、DEBUG 用に使用 読み位置)
DWORD	write_pos	(内部、DEBUG 用に使用 書き位置)
DWORD	write_mode	書きモード 1:シングル 2:マルチ
DWORD	read_open_count	読みモードオープンカウンター
DWORD	write_open_count	書きモードオープンカウンター
DWORD	lost_count	欠落カウンター
OBJHANDLE	mem_handle	メモリハンドル値
OBJHANDLE	lock_sem_handle	排他用セマフォハンドル値
OBJHANDLE	notify_sem_handle	書き通知用セマフォハンドル値
char	Name	名称(最大 10 文字まで)

OBJHANDLE は unsigned long long と同様です。

EgMailboxGetProperty

EgMailbox の属性を読み込みます:

シンタックス

```
int32_t EgMailboxGetProperty ( const char    *Name,
                              const char    *propName,
                              void          *Value,
                              int32_t      dwSize);
```

引 数

*Name	[IN] EgTag 名
*propName	[IN] 要求する属性名
*Value	[OUT] 取得した属性値
dwSize	[IN] value バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxGetProperty コール例

```
uint32_t colSize; // ColSize を読み出す
result = EgMailboxGetProperty ("MyApp" , "ColSize" ,colSize ,sizeof(colSize) );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

注釈

- 指定可能な属性名は以下のとおりです。

PropName	型	内容
"ColSize"	uint32_t	データ格納メールボックス 1 レコードサイズ
"RowSize"	uint32_t	データ格納メールボックスレコード数

EgMailboxGetPropertyByIndex

EgMailbox の属性を読み込みます(インデックス指定版):

シンタックス

```
int32_t EgMailboxGetPropertyByIndex ( EDGE_INDEX *index,
                                     const char *propName,
                                     void *Value,
                                     int32_t dwSize);
```

引 数

<i>*index</i>	[IN] EgTag のインデックス
<i>*propName</i>	[IN] 要求する属性名
<i>*Value</i>	[OUT] 取得した属性値
<i>dwSize</i>	[IN] value バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定したインデックスは登録されていない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxGetPropertyByIndex コール例

```
uint32_t colSize; // ColSize を読み出す
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください
result = EgMailboxGetPropertyByIndex (&index, "ColSize", colSize, sizeof(colSize));
if ( result == EDGE_SUCCESS )
    printf("成功");
```

注釈

- 指定可能な属性名は以下のとおりです。

PropName	型	内容
"ColSize"	uint32_t	データ格納メールボックス 1 レコードサイズ
"RowSize"	uint32_t	データ格納メールボックスレコード数

EgMailboxGetCount

EgMailbox の登録数を取得します:

シンタックス

```
int32_t EgMailboxGetCount ( uint32_t *pItemNum );
```

引 数

***pItemNum** [OUT] EgMailbox の登録数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxGetCount コール例

```
uint32_t itemNum;  
result = EgMailboxGetCount (&itemNum );  
if ( result == EDGE_SUCCESS )  
    printf("成功");
```

EgMailBoxGetEnum

EgMailbox の一覧を取得します:

シンタックス

```
int32_t EgMailBoxGetEnum ( EDGE_INDEX *pBuffer,
                           uint32_t   dwSize,
                           uint32_t   *pItemNum);
```

引 数

***pBuffer** [OUT] 取得したインデックス一覧

dwSize [IN] バッファサイズ

***pItemNum** [OUT] インデックス一覧の取得数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailBoxGetEnum コール例

```
uint32_t itemNum;
EDGE_INDEX *pBuffer;
uint32_t size;
uint32_t getNum;
EgMailboxGetCount (&itemNum );
size = sizeof(EDGE_INDEX) * itemNum;
pBuffer = malloc(size);
result = EgMailboxGetEnum (pBuffer, size, &getNum );
if ( result == EDGE_SUCCESS )
    printf("成功");
free(pBuffer);
```

EgTagCreateEx

EgTag を生成します:

シンタックス

```
int32_t EgTagCreateEx ( const char    *Name,
                        uint16_t    Type,
                        const char    *Source,
                        const char    *Comment );
```

引 数

*Name	[IN] EgTag 名(半角英数時 48 バイトまで)
Type	[IN] 型(後述)
*Source	[IN] データソース
*Comment	[IN] タグの説明文(全角半角英数 48 バイトまで)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_RTCD_ARG_DATA_SIZE_ERROR	0x80000300	指定した引数の変数サイズが既定のサイズより大きい,または小さいためバッファを壊す
EDGE_TTABLE_MEMORRY_ERROR	0x80000400	メモリの作成ができない(INtime の場合はカーネルが起動していない等)
EDGE_TTABLE_ISEXIST_TAGNAME	0x80000409	同名登録済み
EDGE_TTABLE_NOEMPTY	0x8000040E	テーブルに空きがありません。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 既に登録されているか内部的にチェックを行い、登録されている場合は作成しません。
- この API を使用して、文字列型/バイト配列型 EgTag を作成することはできません。これらのタグを作成する場合は、EgTagCreateSegment API を使用してください。
- この API を使用して、FIFO 型 EgTag を作成することはできません。この型のタグを作成する場合は、EgTagCreateFIFO API を使用してください。

コールサンプル

EgTagCreateEx コール例

```
int32_t result;  
result = EgTagCreateEx ( "MyApp.Value1" ,EgVtUInt32 ,"%D0" ,"Sample tag 01" );  
if ( result == EDGE_SUCCESS )  
    printf("成功");
```

EgTagCreateSp

EgTag を生成します。既に生成済みのタグ名を指定した場合は、別名でリンクタグを生成します。:

シンタックス

```
int32_t EgTagCreateSp ( const char    *Name,
                        uint16_t      Type,
                        const char    *Source,
                        const char    *Comment
                        char           *regName
                        uint16_t      regNameSize
                        uint16_t      *isLink );
```

引 数

*Name	[IN] EgTag 名(半角英数時 48 バイトまで)
Type	[IN] 型(後述)
*Source	[IN] データソース
*Comment	[IN] タグの説明文(全角半角英数 48 バイトまで)
*regName	[OUT]実際に登録されたタグ名
regNameSize	[IN]regName のバッファサイズ
*isLink	[OUT]リンクタグフラグ(0:通常タグ、1:リンクタグ)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_DUPLICATE_ERROR	0x80000418	Tag の重複登録失敗
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_RTCD_ARG_DATA_SIZE_ERROR	0x80000300	指定した引数の変数サイズが既定のサイズより大きい,または小さいためバッファを壊す
EDGE_TTABLE_MEMORRY_ERROR	0x80000400	メモリの作成ができない(INtime の場合はカーネルが起動していない等)
EDGE_TTABLE_NOEMPTY	0x8000040E	テーブルに空きがありません。

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- リンクタグとして生成される場合、タグ名先頭に"#*"の2文字が自動的に付与されます。
(*は0~9の数字が順番に割り当てられます)
- リンクタグは最大10個まで生成可。
- このAPIを使用して、文字列型/バイト配列型EgTagを作成することはできません。これらのタグを作成する場合は、EgTagCreateSegment API を使用してください。
- このAPIを使用して、FIFO型EgTagを作成することはできません。この型のタグを作成す

る場合は、EgTagCreateFIFO API を使用してください。

コールサンプル

EgTagCreateSp コール例

```
int32_t result;  
char regName[EG_MAX_TAGNAME + 1]; // 実際に登録されたタグ名を取得する  
uint16_t isLink;  
result = EgTagCreateSp ( "MyApp.Value1" ,EgVtUInt32 ,"%D0" ,"Sample tag 01",  
regName, sizeof(regName), &isLink );  
if ( result == EDGE_SUCCESS )  
printf("成功");
```

EgTagWrite

EgTag にデータを書き込みます:

シンタックス

```
void EgTagWrite ( const char    *Name,
                  void          *pBuffer,
                  uint16_t      wSize );
```

引 数

*Name	[IN] EgTag 名
*pBuffer	[IN] 書き込みデータ
wSize	[IN] 書き込みサイズ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定された値を Tag 構造体内の値に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagWrite コール例

```
int32_t result;
result = EgTagCreate ( "MyApp.Value1" ,EgVtUInt32 ,4 ,"%D0" ,"Sample tag 01" );
:
uint32_t dwValue1 = 0x12345678;
result = EgTagWrite ( "MyApp.Value1" ,&dwValue1 ,sizeof(dwValue1) );
if ( result == EDGE_SUCCESS )
printf("成功");
```


EgTagRead

EgTag のデータを読み込みます:

シンタックス

```
Int32_t EgTagRead ( const char *Name,
                    void *byteArray,
                    uint16_t wSize );
```

引 数

- *Name** [IN] EgTag 名
- *byteArray** [OUT] 読み込みデータ
- wSize** [IN] byteArray バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

```
EgTagRead コール例

int32_t result;
result = EgTagCreate ( "MyApp.Value1",EgVtUInt32 ,4 ,"%D0","Sample tag 01" );
:
uint32_t dwValue1;
result = EgTagRead ( "MyApp.Value1",&dwValue1 ,sizeof(dwValue1) );
if ( result == EDGE_SUCCESS )
printf("成功");
```

EgTagGetProperty

EgTag の属性を読み込みます:

シンタックス

```
int32_t EgTagGetProperty ( const char *Name,
                          const char *propName,
                          void *Value,
                          int32_t dwSize);
```

引 数

*Name	[IN] EgTag 名
*propName	[IN] 要求する属性名
*Value	[OUT] 取得した属性値
dwSize	[IN] value バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagGetProperty コール例

```
int32_t result;
result = EgTagCreate ( "MyApp.Value1",EgVtUInt32,4,"%D0","Sample tag 01" );
:
char comment[49]; // コメントを読み出す
result = EgTagGetProperty ("MyApp.Value1","Comment",comment,sizeof(comment));
if ( result == EDGE_SUCCESS )
printf("成功");
```

注釈

- 指定可能な属性名は以下のとおりです。

PropName	型	内容
"Type"	UInt16	タグのデータ型番号を取得します
"Size"	UInt16	タグのデータサイズ(byte)を取得します
"Comment"	char[48+1]	コメント文字列を取得します
"Source"	char[48+1]	アドレスソース文字列を取得します
"Address"	char[48+1]	同上
"Value"		タグの値を取得します。データサイズはタグにより異なります
"RecordCount"	UInt16	(FIFO 型タグ) 登録可能なデータ数の上限を取得します
"Remain"	UInt16	(FIFO 型タグ) タグに登録されているデータ件数を取得します
"Entry"	void*	内部：エントリのメモリポインタを取得します
"Index"	void*	内部：インデックスのメモリポインタを取得します
"Hash"	UInt64	内部：タグのハッシュコードを取得します
"Name"	char[48+1]	内部：タグ名文字列を取得します

EgTagGetCount

EgTag の登録数を取得します:

シンタックス

```
int32_t EgTagGetCount ( uint32_t *pItemNum );
```

引 数

***pItemNum** [OUT] EgTag の登録数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagGetCount コール例

```
uint32_t itemNum;  
result = EgTagGetCount (&itemNum );  
if ( result == EDGE_SUCCESS )  
    printf("成功");
```

EgTagGetEnum

EgTag の一覧を取得します:

シンタックス

```
int32_t EgTagGetEnum ( EDGE_INDEX *pBuffer,
                      uint32_t    dwSize,
                      uint32_t    *pItemNum) ;
```

引 数

***pBuffer** [OUT] 取得したインデックス一覧

dwSize [IN] バッファサイズ

***pItemNum** [OUT] インデックス一覧の取得数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagGetEnum コール例

```
uint32_t itemNum;
EDGE_INDEX *pBuffer;
uint32_t size;
uint32_t getNum;
EgTagGetCount (&itemNum ); // 登録数取得
size = sizeof(EDGE_INDEX) * itemNum;
pBuffer = malloc(size); // バッファ確保
result = EgTagGetEnum (pBuffer, size, &getNum ); // 一覧取得
if ( result == EDGE_SUCCESS )
    printf("成功");
free(pBuffer);
```

EgTagWriteByIndex

EgTag にデータを書き込みます(インデックス指定版):

シンタックス

```
void EgTagWriteByIndex ( EDGE_INDEX *index,  
                        void          *pBuffer,  
                        uint16_t      wSize );
```

引 数

***index** [IN] EgTag のインデックス
***pBuffer** [IN] 書き込みデータ
wSize [IN] 書き込みサイズ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された値を Tag 構造体内の値に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagWriteByIndex コール例

```
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください  
uint32_t dwValue1 = 0x12345678;  
result = EgTagWriteByIndex ( &index ,&dwValue1 ,sizeof(dwValue1) );  
if ( result == EDGE_SUCCESS )  
    printf("成功");
```

EgTagReadByIndex

EgTag のデータを読み込みます(インデックス指定版):

シンタックス

```
Int32_t EgTagReadByIndex( EDGE_INDEX *index,
                          void         *byteArray,
                          uint16_t     wSize );
```

引 数

***index** [IN] EgTag のインデックス

***byteArray** [OUT] 読み込みデータ

wSize [IN] byteArray バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagReadByIndex コール例

```
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください
uint32_t dwValue1;
result = EgTagReadByIndex ( &index ,&dwValue1 ,sizeof(dwValue1) );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagGetPropertyByIndex

EgTag の属性を読み込みます(インデックス指定版):

シンタックス

```
int32_t EgTagGetPropertyByIndex ( EDGE_INDEX *index,
                                const char   *propName,
                                void         *Value,
                                int32_t      dwSize);
```

引 数

*index	[IN] EgTag のインデックス
*propName	[IN] 要求する属性名
*Value	[OUT] 取得した属性値
dwSize	[IN] value バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定したインデックスは登録されていない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagGetProperty コール例

```
char    comment[49]; // コメントを読み出す
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください
result = EgTagGetPropertyByIndex (&index, "Comment", comment, sizeof(comment));
if ( result == EDGE_SUCCESS )
    printf("成功");
```


注釈

- 指定可能な属性名は以下のとおりです。

PropName	型	内容
"Type"	UInt16	タグのデータ型番号を取得します
"Size"	UInt16	タグのデータサイズ(byte)を取得します
"Comment"	char[48+1]	コメント文字列を取得します
"Source"	char[48+1]	アドレスソース文字列を取得します
"Address"	char[48+1]	同上
"Value"		タグの値を取得します。データサイズはタグにより異なります
"RecordCount"	UInt16	(FIFO 型タグ) 登録可能なデータ数の上限を取得します
"Remain"	UInt16	(FIFO 型タグ) タグに登録されているデータ件数を取得します
"Entry"	void*	内部：エントリのメモリポインタを取得します
"Index"	void*	内部：インデックスのメモリポインタを取得します
"Hash"	UInt64	内部：タグのハッシュコードを取得します
"Name"	char[48+1]	内部：タグ名文字列を取得します

EgTagWriteByEntry

EgTag にデータを書き込みます(Entry 指定版):

シンタックス

```
void EgTagWriteByEntry ( void          *pEntry,
                        void          *pBuffer,
                        uint16_t      wSize );
```

引 数

***pEntry** [IN] EgTag のエントリ
***pBuffer** [IN] 書き込みデータ
wSize [IN] 書き込みサイズ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定された値を Tag 構造体内の値に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagWriteByEntry コール例

```
void *pEntry; // NOTE: エントリは EgTagGetProperty で取得してください
uint32_t dwValue1 = 0x12345678;
result = EgTagWriteByEntry ( pEntry ,&dwValue1 ,sizeof(dwValue1) );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagReadByEntry

EgTag のデータを読み込みます(Entry 指定版):

シンタックス

```
Int32_t EgTagReadByEntry ( void *pEntry, void *byteArray, uint16_t wSize );
```

引 数

***pEntry** [IN] EgTag のエントリ
***byteArray** [OUT] 読み込みデータ
wSize [IN] byteArray バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagReadByEntry コール例

```
void *pEntry; // NOTE: エントリは EgTagGetProperty で取得してください
uint32_t dwValue1;
result = EgTagReadByEntry ( pEntry ,&dwValue1 ,sizeof(dwValue1) );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagCreateSegment

文字列型/バイト配列型 EgTag を生成します。既に生成済みのタグ名を指定した場合は、別名でリンクタグを生成します。:

シンタックス

```
int32_t EgTagCreateSegment ( const char    *Name,
                             uint16_t     Type,
                             uint16_t     Size,
                             const char    *Source,
                             const char    *Comment
                             char          *regName
                             uint16_t     regNameSize
                             uint16_t     *isLink );
```

引 数

*Name	[IN] EgTag 名(半角英数時 48 バイトまで)
Type	[IN] 型(12:文字列、13:バイト配列)
Size	[IN] 文字列型/バイト配列型 EgTag のデータサイズ
*Source	[IN] データソース
*Comment	[IN] タグの説明文(全角半角英数 48 バイトまで)
*regName	[OUT]実際に登録されたタグ名
regNameSize	[IN]regName のバッファサイズ
*isLink	[OUT]リンクタグフラグ(0:通常タグ、1:リンクタグ)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_DUPLICATE_ERROR	0x80000418	Tag の重複登録失敗
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_RTCD_ARG_DATA_SIZE_ERROR	0x80000300	指定した引数の変数サイズが既定のサイズより大きい,または小さいためバッファを壊す
EDGE_TTABLE_MEMORRY_ERROR	0x80000400	メモリの作成ができない(INtime の場合はカーネルが起動していない等)
EDGE_TTABLE_NOEMPTY	0x8000040E	テーブルに空きがありません。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- リンクタグとして生成される場合、タグ名先頭に"#*"の2文字が自動的に付与されます。
(*は0~9の数字が順番に割り当てられます)
- リンクタグは最大10個まで生成可。

コールサンプル

EgTagCreateSegment コール例

```
int32_t result;  
char regName[EG_MAX_TAGNAME + 1]; // 実際に登録されたタグ名を取得する  
uint16_t isLink;  
result = EgTagCreateSegement ( "MyApp.Value1" ,EgVtByteArray ,64, "" ,"Sample tag  
01", regName, sizeof(regName), &isLink );  
if ( result == EDGE_SUCCESS )  
printf("成功");
```

EgTagWriteSegment

バイト配列型 EgTag にデータを書き込みます:

シンタックス

```
Int32_t EgTagWriteSegment ( const char    *Name,
                             uint16_t    wOffset,
                             void        *pBuffer,
                             uint16_t    wSize );
```

引 数

*Name	[IN] EgTag 名
wOffset	[IN] バイト配列オフセット
*pBuffer	[IN] 書き込みデータ
wSize	[IN] 書き込みサイズ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された値をバイト配列型用 Tag が確保した領域に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagWriteSegment コール例

```
int32_t result;
char regName[EG_MAX_TAGNAME + 1]; // 実際に登録されたタグ名を取得する
uint16_t isLink;
result = EgTagCreateSegment ( "MyApp.Value1", EgVtByteArray ,64 ,"" ,"Sample tag 01" ,
regName, sizeof(regName), &isLink );
:
uint8_t byArray[16];
memset( byArray ,0xFF ,sizeof(byArray) );
result = EgTagWriteSegment ( "MyApp.Value1" ,32 , byArray ,sizeof(byArray) );
if ( result == EDGE_SUCCESS )
printf("成功");
```

EgTagReadSegment

EgTag のデータを読み込みます:

シンタックス

```
Int32_t EgTagReadSegment ( const char *Name,
                           uint16_t wOffset,
                           void *byteArray,
                           uint16_t wSize );
```

引 数

<i>*Name</i>	[IN] EgTag 名
<i>wOffset</i>	[IN] バイト配列オフセット
<i>*byteArray</i>	[OUT] 読み込みデータ
<i>wSize</i>	[IN] byteArray バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagReadSegment コール例

```
int32_t result;
char regName[EG_MAX_TAGNAME + 1]; // 実際に登録されたタグ名を取得する
uint16_t isLink;
result = EgTagCreateSegment ( "MyApp.Value1", EgVtByteArray ,64 ,"" ,"Sample tag 01" ,
regName, sizeof(regName), &isLink );
:
uint8_t byArray[16];
result = EgTagReadSegment ( "MyApp.Value1" , 32, byArray ,sizeof(byArray) );
if ( result == EDGE_SUCCESS )
printf("成功");
```

EgTagWriteSegmentByIndex

バイト配列型 EgTag にデータを書き込みます(インデックス指定版):

シンタックス

```
void EgTagWriteSegmentByIndex( EDGE_INDEX    *index,
                               uint16_t      wOffset,
                               void          *pBuffer,
                               uint16_t      wSize );
```

引 数

<i>*index</i>	[IN] EgTag のインデックス
<i>wOffset</i>	[IN] バイト配列オフセット
<i>*pBuffer</i>	[IN] 書き込みデータ
<i>wSize</i>	[IN] 書き込みサイズ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された値をバイト配列型用 Tag が確保した領域に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagWriteSegmentByIndex コール例

```
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください
uint8_t    byArray[16];
memset( byArray , 0xFF ,sizeof(byArray) );
result = EgTagWriteSegmentByIndex ( &index ,32, byArray ,sizeof(byArray) );
if ( result == EDGE_SUCCESS )
    printf("成功");
```


EgTagReadSegmentByIndex

バイト配列型 EgTag のデータを読み込みます(インデックス指定版):

シンタックス

```
Int32_t EgTagReadSegmentByIndex ( EDGE_INDEX      *index,
                                   uint16_t         wOffset,
                                   void              *byteArray,
                                   uint16_t         wSize );
```

引 数

- *index** [IN] EgTag のインデックス
- wOffset** [IN] バイト配列オフセット
- *byteArray** [OUT] 読み込みデータ
- wSize** [IN] byteArray バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagReadSegmentByIndex コール例

```
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください
uint8_t    byteArray[16];
result = EgTagReadSegmentByIndex ( &index ,32 ,byteArray ,sizeof(byteArray) );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagWriteSegmentByEntry

バイト配列型 EgTag にデータを書き込みます(Entry 指定版):

シンタックス

```
void EgTagWriteSegmentByEntry( void *pEntry,
                               uint16_t wOffset,
                               void *pBuffer,
                               uint16_t wSize );
```

引 数

*pEntry	[IN] EgTag のエントリ
wOffset	[IN] バイト配列オフセット
*pBuffer	[IN] 書き込みデータ
wSize	[IN] 書き込みサイズ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された値をバイト配列型用 Tag が確保した領域に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagWriteSegmentByEntry コール例

```
void *pEntry; // NOTE: エントリは EgTagGetProperty で取得してください
uint8_t byArray[16];
memset( byArray , 0xFF ,sizeof(byArray) );
result = EgTagWriteSegmentByEntry ( pEntry ,32 ,byArray ,sizeof(byArray) );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagReadSegmentByEntry

バイト配列型 EgTag のデータを読み込みます(Entry 指定版):

シンタックス

```
Int32_t EgTagReadSegmentByEntry( void      *pEntry,
                                uint16_t  wOffset,
                                void      *byteArray,
                                uint16_t  wSize );
```

引 数

*pEntry	[IN] EgTag のエントリ
wOffset	[IN] バイト配列オフセット
*byteArray	[OUT] 読み込みデータ
wSize	[IN] byteArray バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagReadSegmentByEntry コール例

```
void *pEntry; // NOTE: エントリは EgTagGetProperty で取得してください
uint8_t  byteArray[16];
result = EgTagReadSegmentByEntry ( pEntry ,32 ,byteArray ,sizeof(byteArray) );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagCreateFIFO

FIFO 型 EgTag を生成します。既に生成済みのタグ名を指定した場合は、別名でリンクタグを生成します。:

シンタックス

```
int32_t EgTagCreateFIFO (  const char    *Name,
                          uint16_t      RecordSize,
                          uint16_t      RecordCount,
                          bool           Reserved1_LetFalse,
                          const char    *Comment,
                          char          *regName,
                          uint16_t      regNameSize,
                          uint16_t      *isLink );
```

引 数

*Name	[IN] EgTag 名(半角英数時 48 バイトまで)
RecordSize	[IN] タグに登録する 1 件のデータの長さ(バイト単位)
RecordCount	[IN] タグに登録可能な最大データ件数
Reserved1_LetFalse	[IN] 未使用(false)
*Comment	[IN] タグの説明文(全角半角英数 48 バイトまで)
*regName	[OUT] 実際に登録されたタグ名
regNameSize	[IN] regName のバッファサイズ
*isLink	[OUT] リンクタグフラグ(0:通常タグ、1:リンクタグ)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_DUPLICATE_ERROR	0x80000418	Tag の重複登録失敗
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_RTCD_ARG_DATA_SIZE_ERROR	0x80000300	指定した引数の変数サイズが既定のサイズより大きい,または小さいためバッファを壊す
EDGE_TTABLE_MEMORRY_ERROR	0x80000400	メモリの作成ができない(INtime の場合はカーネルが起動していない等)
EDGE_TTABLE_NOEMPTY	0x8000040E	テーブルに空きがありません。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- リンクタグとして生成される場合、タグ名先頭に"#*"の 2 文字が自動的に付与されます。
(*は 0~9 の数字が順番に割り当てられます)
- リンクタグは最大 10 個まで生成可。

コールサンプル

EgTagCreateFIFO コール例

```
int32_t result;
char regName[EG_MAX_TAGNAME + 1]; // 実際に登録されたタグ名を取得する
uint16_t isLink;
result = EgTagCreateFIFO ( "MyApp.FIFO1", 4, 1024, false, "FIFO(4 byte * 1024 record)",
regName, sizeof(regName), &isLink );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagEnqueueFIFO

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます:

シンタックス

```
int32_t EgTagEnqueueFIFO ( const char *Name,
                           void *byteArray,
                           uint16_t wSize,
                           uint16_t *wRemain );
```

引 数

*Name	[IN] EgTag 名
*byteArray	[IN] 登録するデータ
wSize	[IN] 登録するデータのサイズ(バイト単位)
*wRemain	[OUT] 登録が完了した時点での、タグに登録されているデータ件数 NULL を指定すると取得しない

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、古いデータから順に上書き(破壊)される動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- byteArray に NULL を指定するか、wSize に 0 を指定すると、現在書き込み位置にあるデータを登録することができます。書き込み位置のデータを更新するには、EgTagWrite API を使用してください。

コールサンプル

EgTagEnqueueFIFO コール例

```
int32_t result;
uint8_t byArray[4];
uint16_t remain;
memset( byArray, 0xFF, sizeof(byArray) );
result = EgTagEnqueueFIFO ( "MyApp.FIFO1", byArray, sizeof(byArray), &remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagEnqueueFIFOByIndex

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます(インデックス指定版):

シンタックス

```
int32_t EgTagEnqueueFIFOByIndex ( EDGE_INDEX *index,
                                   void *byteArray,
                                   uint16_t wSize,
                                   uint16_t *wRemain );
```

引 数

*index	[IN] EgTag のインデックス
*byteArray	[IN] 登録するデータ
wSize	[IN] 登録するデータのサイズ(バイト単位)
*wRemain	[OUT] 登録が完了した時点での、タグに登録されているデータ件数 NULL を指定すると取得しない

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、古いデータから順に上書き(破壊)される動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- byteArray に NULL を指定するか、wSize に 0 を指定すると、現在書き込み位置にあるデータを登録することができます。書き込み位置のデータを更新するには、EgTagWriteByIndex API を使用してください。

コールサンプル

EgTagEnqueueFIFOByIndex コール例

```
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください
uint8_t  byArray[4];
uint16_t remain;
memset( byArray, 0xFF, sizeof(byArray) );
result = EgTagEnqueueFIFOByIndex ( &index, byArray, sizeof(byArray), &remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagEnqueueFIFOByEntry

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます(Entry 指定版):

シンタックス

```
int32_t EgTagEnqueueFIFOByEntry ( void      *pEntry,
                                   void      *byteArray,
                                   uint16_t  wSize,
                                   uint16_t  *wRemain );
```

引 数

*pEntry	[IN] EgTag のエントリ
*byteArray	[IN] 登録するデータ
wSize	[IN] 登録するデータのサイズ(バイト単位)
*wRemain	[OUT] 登録が完了した時点での、タグに登録されているデータ件数 NULL を指定すると取得しない

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、古いデータから順に上書き(破壊)される動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- byteArray に NULL を指定するか、wSize に 0 を指定すると、現在書き込み位置にあるデータを登録することができます。書き込み位置のデータを更新するには、EgTagWriteByEntry API を使用してください。

コールサンプル

EgTagEnqueueFIFOByEntry コール例

```
void *pEntry; // NOTE: エントリは EgTagGetProperty で取得してください
uint8_t  byArray[4];
uint16_t  remain;
memset( byArray, 0xFF, sizeof(byArray) );
result = EgTagEnqueueFIFOByEntry ( pEntry, byArray, sizeof(byArray), &remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```


EgTagEnqueueRingFIFO

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます(Ring 動作版):

シンタックス

```
int32_t EgTagEnqueueRingFIFO ( const char    *Name,
                               void          *byteArray,
                               uint16_t      wSize,
                               uint16_t      *wRemain );
```

引 数

*Name	[IN] EgTag 名
*byteArray	[IN] 登録するデータ
wSize	[IN] 登録するデータのサイズ(バイト単位)
*wRemain	[OUT] 登録が完了した時点での、タグに登録されているデータ件数 NULL を指定すると取得しない

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、一番古いデータが読み捨てられる動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- byteArray に NULL を指定するか、wSize に 0 を指定すると、現在書き込み位置にあるデータを登録することができます。書き込み位置のデータを更新するには、EgTagWrite API を使用してください。

コールサンプル

EgTagEnqueueRingFIFO コール例

```
int32_t result;
uint8_t  byArray[4];
uint16_t remain;
memset( byArray, 0xFF, sizeof(byArray) );
result = EgTagEnqueueRingFIFO ( "MyApp.FIFO1", byArray, sizeof(byArray), &remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagEnqueueRingFIFOByIndex

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます(Ring 動作・インデックス指定版):

シンタックス

```
int32_t EgTagEnqueueRingFIFOByIndex ( EDGE_INDEX *index,
                                       void *byteArray,
                                       uint16_t wSize,
                                       uint16_t *wRemain );
```

引 数

*index	[IN] EgTag のインデックス
*byteArray	[IN] 登録するデータ
wSize	[IN] 登録するデータのサイズ(バイト単位)
*wRemain	[OUT] 登録が完了した時点での、タグに登録されているデータ件数 NULL を指定すると取得しない

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、一番古いデータが読み捨てられる動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- byteArray に NULL を指定するか、wSize に 0 を指定すると、現在書き込み位置にあるデータを登録することができます。書き込み位置のデータを更新するには、EgTagWriteByIndex API を使用してください。

コールサンプル

EgTagEnqueueRingFIFOByIndex コール例

```
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください
uint8_t  byteArray[4];
uint16_t remain;
memset( byteArray, 0xFF, sizeof(byteArray) );
result = EgTagEnqueueRingFIFOByIndex ( &index, byteArray, sizeof(byteArray), &remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagEnqueueRingFIFOByEntry

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます(Ring 動作・Entry 指定版):

シンタックス

```
int32_t EgTagEnqueueRingFIFOByEntry ( void      *pEntry,
                                       void      *byteArray,
                                       uint16_t  wSize,
                                       uint16_t  *wRemain );
```

引 数

*pEntry	[IN] EgTag のエントリ
*byteArray	[IN] 登録するデータ
wSize	[IN] 登録するデータのサイズ(バイト単位)
*wRemain	[OUT] 登録が完了した時点での、タグに登録されているデータ件数 NULL を指定すると取得しない

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、一番古いデータが読み捨てられる動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- byteArray に NULL を指定するか、wSize に 0 を指定すると、現在書き込み位置にあるデータを登録することができます。書き込み位置のデータを更新するには、EgTagWriteByEntry API を使用してください。

コールサンプル

EgTagEnqueueRingFIFOByEntry コール例

```
void *pEntry; // NOTE: エントリは EgTagGetProperty で取得してください
uint8_t  byteArray[4];
uint16_t  remain;
memset( byteArray, 0xFF, sizeof(byteArray) );
result = EgTagEnqueueRingFIFOByEntry ( pEntry, byteArray, sizeof(byteArray), &remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagDequeueFIFO

FIFO 型 EgTag からデータを 1 件取得し、読出し位置を進めます:

シンタックス

```
int32_t EgTagDequeueFIFO ( const char *Name,
                           void *byteArray,
                           uint16_t wSize,
                           uint16_t *wRemain );
```

引 数

*Name	[IN] EgTag 名
*byteArray	[OUT] データ取得先
wSize	[IN] 取得データのうち、取得先に読出すサイズ(バイト単位)
*wRemain	[OUT] 取得が完了した時点での、タグに登録されている残りデータ件数 NULL を指定すると取得しない

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_UNDERRUN	0x80006001	タグに登録されたデータがない

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録されたデータがない状態で取得を行うと、読出し位置にある不明なデータを読み出し、読出し位置を進めます。このとき、戻り値は EDGE_FIFO_UNDERRUN です。
- byteArray に NULL を指定するか、wSize に 0 を指定すると、現在読出し位置にあるデータを読み捨てることができます。
- 読出し位置を更新せずデータを取得するには、EgTagRead API をご利用下さい。

コールサンプル

EgTagDequeueFIFO コール例

```
int32_t result;
uint8_t  byArray[4];
uint16_t remain;
result = EgTagDequeueFIFO ( "MyApp.FIFO1", byArray, sizeof(byArray), &remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagDequeueFIFOByIndex

FIFO 型 EgTag からデータを 1 件取得し、読出し位置を進めます(インデックス指定版):

シンタックス

```
int32_t EgTagDequeueFIFOByIndex ( EDGE_INDEX *index,
                                   void         *byteArray,
                                   uint16_t      wSize,
                                   uint16_t      *wRemain );
```

引 数

*index	[IN] EgTag のインデックス
*byteArray	[OUT] データ取得先
wSize	[IN] 取得データのうち、取得先に読出すサイズ(バイト単位)
*wRemain	[OUT] 取得が完了した時点での、タグに登録されている残りデータ件数 NULL を指定すると取得しない

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_UNDERRUN	0x80006001	タグに登録されたデータがない

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録されたデータがない状態で取得を行うと、読出し位置にある不明なデータを読み出し、読出し位置を進めます。このとき、戻り値は EDGE_FIFO_UNDERRUN です。
- byteArray に NULL を指定するか、wSize に 0 を指定すると、現在読出し位置にあるデータを読み捨てることができます。
- 読出し位置を更新せずデータを取得するには、EgTagReadByIndex API をご利用下さい。

コールサンプル

EgTagDequeueFIFOByIndex コール例

```
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください
uint8_t    byteArray[4];
uint16_t    remain;
result = EgTagDequeueFIFOByIndex ( &index, byteArray, sizeof(byteArray), &remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagDequeueFIFOByEntry

FIFO 型 EgTag からデータを 1 件取得し、読出し位置を進めます(Entry 指定版):

シンタックス

```
int32_t EgTagDequeueFIFOByEntry ( void *pEntry, void *byteArray, uint16_t wSize, uint16_t *wRemain );
```

引 数

*pEntry	[IN] EgTag のエントリ
*byteArray	[OUT] データ取得先
wSize	[IN] 取得データのうち、取得先に読出すサイズ(バイト単位)
*wRemain	[OUT] 取得が完了した時点での、タグに登録されている残りデータ件数 NULL を指定すると取得しない

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_UNDERRUN	0x80006001	タグに登録されたデータがない

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録されたデータがない状態で取得を行うと、読出し位置にある不明なデータを読み出し、読出し位置を進めます。このとき、戻り値は EDGE_FIFO_UNDERRUN です。
- byteArray に NULL を指定するか、wSize に 0 を指定すると、現在読出し位置にあるデータを読み捨てることができます。
- 読出し位置を更新せずデータを取得するには、EgTagReadByEntry API をご利用下さい。

コールサンプル

EgTagDequeueFIFOByEntry コール例

```
void *pEntry; // NOTE: エントリは EgTagGetProperty で取得してください
uint8_t  byArray[4];
uint16_t  remain;
result = EgTagDequeueFIFOByEntry ( pEntry, byArray, sizeof(byArray), &remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagClearFIFO

FIFO 型 EgTag に登録されたデータを全て消去し、書き込み位置・読出し位置をリセットします:

シンタックス

```
int32_t EgTagClearFIFO ( const char *Name );
```

引 数

***Name** [IN] EgTag 名

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- データ領域のクリアは行われません。

コールサンプル

EgTagClearFIFO コール例

```
int32_t result;
result = EgTagClearFIFO ("MyApp.FIFO1");
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagClearFIFOByIndex

FIFO 型 EgTag に登録されたデータを全て消去し、書き込み位置・読出し位置をリセットします(インデックス指定版):

シンタックス

```
int32_t EgTagClearFIFOByIndex ( EDGE_INDEX *index );
```

引 数

***index** [IN] EgTag のインデックス

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- データ領域のクリアは行われません。

コールサンプル

EgTagClearFIFOByIndex コール例

```
int32_t result;
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください
result = EgTagClearFIFOByIndex (&index);
if ( result == EDGE_SUCCESS )
    printf("成功");
```


EgTagClearFIFOByEntry

FIFO 型 EgTag に登録されたデータを全て消去し、書き込み位置・読出し位置をリセットします(Entry 指定版):

シンタックス

```
int32_t EgTagClearFIFOByEntry ( void *pEntry );
```

引 数

***pEntry** [IN] EgTag のエントリ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- データ領域のクリアは行われません。

コールサンプル

EgTagClearFIFOByEntry コール例

```
int32_t result;
void *pEntry; // NOTE: エントリは EgTagGetProperty で取得してください
result = EgTagClearFIFOByEntry ( pEntry );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagGetRemainFIFO

FIFO 型 EgTag に登録されているデータ件数を取得します:

シンタックス

```
int32_t EgTagGetRemainFIFO ( const char *Name,
                             uint16_t *wRemain );
```

引 数

***Name** [IN] EgTag 名

***wRemain** [OUT] タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagGetRemainFIFO コール例

```
int32_t result;
char regName[EG_MAX_TAGNAME + 1]; // 実際に登録されたタグ名を取得する
uint16_t isLink;
result = EgTagCreateFIFO ( "MyApp.FIFO1", 4, 1024, false, "FIFO(4 byte * 1024 record)",
regName, sizeof(regName), &isLink );
:
uint8_t byArray[4] = {0};
uint16_t remain;
result = EgTagGetRemainFIFO ("MyApp.FIFO1", &remain );
if ( result == EDGE_SUCCESS && remain < 1024 ) {
    // データを登録する余裕がある。データを登録する
    result = EgTagEnqueueFIFO ( "MyApp.FIFO1", byArray, sizeof(byArray), NULL );
}
```

EgTagGetRemainFIFOByIndex

FIFO 型 EgTag に登録されているデータ件数を取得します(インデックス指定版):

シンタックス

```
int32_t EgTagGetRemainFIFOByIndex ( EDGE_INDEX *index,
                                     uint16_t   *wRemain );
```

引 数

***index** [IN] EgTag のインデックス
***wRemain** [OUT] タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagGetRemainFIFOByIndex コール例

```
int32_t result;
uint8_t  byArray[4];
EDGE_INDEX index; // NOTE: インデックスは一覧取得 API で取得してください
uint16_t remain;
result = EgTagGetRemainFIFOByIndex (&index, &remain );
if ( result == EDGE_SUCCESS && remain > 0 ) {
    // 取得可能なデータがある。データを取得する
    result = EgTagDequeueFIFOByIndex ( &index, byArray, sizeof(byArray), NULL );
}
```

EgTagGetRemainFIFOByEntry

FIFO 型 EgTag に登録されているデータ件数を取得します(Entry 指定版):

シンタックス

```
int32_t EgTagGetRemainFIFOByEntry ( void      *pEntry,
                                     uint16_t  *wRemain );
```

引 数

***pEntry** [IN] EgTag のエントリ

***wRemain** [OUT] タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagGetRemainFIFOByEntry コール例

```
int32_t result;
void *pEntry; // NOTE: エントリは EgTagGetProperty で取得してください
uint16_t remain;
result = EgTagGetRemainFIFOByEntry ( pEntry, &remain );
while ( result == EDGE_SUCCESS && remain > 0 ) {
    // 取得可能なデータがある。全て読み捨てる
    result = EgTagDequeueFIFOByEntry ( pEntry, NULL, 0, &remain );
}
```

EgTagSetDefaultValue

ECI(XML)で定義した Tag に対して、ECI の Value 値を書き込みます:

シンタックス

```
Int32_t EgTagSetDefaultValue (void) ;
```

引 数

なし

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_RESET_DEFAULT_VALUE_TIMEOUT	0x8000041A	フレームワークを利用していない コンテナから実行されている

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagSetDefaultValue コール例

```
result = EgTagSetDefaultValue ();
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagSetDefaultValueByGroup

ECI(XML)で定義した Tag に対して、ECI の Value 値を書き込みます。更新対象は ECI の "InitGroup" 設定を行っている Tag です。対象を API 引数で指定します:

シンタックス

```
Int32_t EgTagSetDefaultValueByGroup (uint16_t wGroup);
```

引 数

wGroup

[IN] 更新対象タグ InitGroup 番号

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_RESET_DEFAULT_VALUE_TIMEOUT	0x8000041A	フレームワークを利用していないコンテナから実行されている

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagSetDefaultValueByGroup コール例

```
Uint16_t groupNo = 1;
result = EgTagSetDefaultValueByGroup (groupNo);
if ( result == EDGE_SUCCESS )
    printf("成功");
```

ECI サンプル

```
<?xml version="1.0" encoding="utf-8"?>
<RTedge xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
:
<Tags>
  <Tag Name ="TagTest_0001" InitGroup="1" Type="6" Value="10"/>
  <Tag Name ="TagTest_0002" InitGroup="1" Type="3" Value="172"/>
  <Tag Name ="TagTest_0003" InitGroup="2" Type="3" Value="171"/>
  <Tag Name ="TagTest_0004" Type="3" Value="1"/>
  <Tag Name ="TagTest_0005" Type="1" Value="0"/>
</Tags>
</RTedge>
```

- 上記 ECI サンプル設定において、コードサンプルを実行した場合、更新されるタグは "TagTest_0001", "TagTest_0002" です。

EgCollectorGetCount

EgCollector の登録数を取得します:

シンタックス

```
int32_t EgCollectorGetCount ( uint32_t *pItemNum);
```

引 数

***pItemNum** [OUT] EgCollector の登録数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgCollectorGetCount コール例

```
uint32_t itemNum;
result = EgCollectorGetCount (&itemNum );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgCollectorGetEnum

EgCollector の一覧を取得します:

シンタックス

```
int32_t EgCollectorGetEnum ( EDGE_INDEX *pBuffer,
                             uint32_t   dwSize,
                             uint32_t   *pItemNum);
```

引 数

*pBuffer	[OUT] 取得したインデックス一覧
dwSize	[IN] バッファサイズ
*pItemNum	[OUT] インデックス一覧の取得数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgCollectorGetEnum コール例

```
uint32_t itemNum;
EDGE_INDEX *pBuffer;
uint32_t size;
uint32_t getNum;
EgCollectorGetCount (&itemNum ); // 登録数取得
size = sizeof(EDGE_INDEX) * itemNum;
pBuffer = malloc(size);          // バッファ確保
result = EgCollectorGetEnum (pBuffer, size, &getNum ); // 一覧取得
if ( result == EDGE_SUCCESS )
    printf("成功");
free(pBuffer);
```


EgCollectorGetProperty

EgCollector の情報を取得します:

シンタックス

```
Int32_t EgCollectorGetProperty ( const char *Name,
                                const char *propName,
                                void *Value,
                                int32_t dwSize );
```

引 数

*Name	[IN] EgCollector 名
*propName	[IN] Property 名
*Value	[OUT] Property 値
dwSize	[IN] Value バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定した名前は登録されていない
EDGE_PARAM_INVALID	0x80000004	指定した属性名は不正

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された Property 名に対応する値を返す。指定可能な Property 名は以下の文字列を指定。

Property 名	型	内容
"Interval"	uint32_t	収集周期(ms)
"Priority"	uint8_t	スレッドプライオリティ (INtime サービスコンテナ用)
"ColSize"	uint32_t	データ格納メールボックス 1 レコードサイズ
"RowSize"	uint32_t	データ格納メールボックスレコード数
"DatasetName"	char[]	データセット名

EgCollectorGetPropertyByIndex

EgCollector の情報を取得します(インデックス指定版):

シンタックス

```
Int32_t EgCollectorGetPropertyByIndex ( EDGE_INDEX *index,
                                         const char  *propName,
                                         void         *Value,
                                         int32_t      dwSize );
```

引 数

*index	[IN] EgCollector のインデックス
*propName	[IN] Property 名
*Value	[OUT] Property 値
dwSize	[IN] Value バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定したインデックスは登録されていない
EDGE_PARAM_INVALID	0x80000004	指定した属性名は不正

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された Property 名に対応する値を返す。指定可能な Property 名は以下の文字列を指定。

Property 名	型	内容
"Interval"	uint32_t	収集周期(ms)
"Priority"	uint8_t	スレッドプライオリティ (INtime サービスコンテナ用)
"ColSize"	uint32_t	データ格納メールボックス 1 レコードサイズ
"RowSize"	uint32_t	データ格納メールボックスレコード数
"DatasetName"	char[]	データセット名

EgDatasetCreate

EgDataset を生成します:

シンタックス

```
void EgDatasetCreate( const char *Name );
```

引 数

***Name** [IN] Dataset 名

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_ISEXIST_TAGNAME	0x80000409	同名登録済み
EDGE_TTABLE_NOEMPTY	0x8000040E	テーブルに空きがない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgDatasetAddTag

EgDataset に EgTag 情報を追加します:

シンタックス

```
int32_t EgDatasetAddTag ( const char    *Name,  
                          const char    *TagName );
```

引 数

***Name** [IN] Dataset 名

***TagName** [IN] Tag 名

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_DSETNAME	0x80000501	指定された Dataset 名はテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgDatasetGetSize

EgDataset 内全ての Tag の size を加算し、1 レコードのバイナリバイトサイズを返します:

シンタックス

```
WORD EgDatasetGetSize( const char *Name );
```

引 数

**Name* [IN] Dataset 名)

戻り値

バイナリバイトサイズ

EgDatasetGetBinary

EgDataset 内全ての Tag の Value を取得し、1 レコードのバイナリ値として連結作成し値を返します:

シンタックス

```
Int32_t EgDatasetGetBinary (  const char  *Name,
                             void          *Dest,
                             size_t        DestSize );
```

引 数

*Name	[IN] Dataset 名
*Dest	[OUT] 連結したバイナリ列の格納先
DestSize	[IN] Dest のバイトサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgDatasetGetFirst

EgDataset の最初のタグを取得します:

シンタックス

```
int32_t EgDatasetGetFirst ( const char *Name,
                           dslink *dslink);
```

引 数

***Name** [IN] Dataset 名

***dsLink** [OUT] Dataset サーチ用構造体

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_DSETNAME	0x80000501	指定された Dataset 名はテーブルに存在しない
EDGE_TTABLE_NOEXISTT_DSETITEM	0x80000502	指定された Dataset 名には Tag が登録されていない

dslink 構造体

```
typedef struct tdslink
{
    char*      TagName;    // タグ名
    void*      tag;        // TagDesc* Tag 情報へのポインタ
    void*      ds;         // DsetDesc* Dataset 情報へのポインタ
    uint16_t   inst;       // FindFirst/Next で使用
    uint16_t   offset;     // Tag オフセット位置
}dslink;
```

TDSLINK 構造体 フィールド

型	シンボル	内容
string	pTagName	取得したタグの名称
IntPtr	pTag	(内部で使用する為使用不可)
IntPtr	Pds	(内部で使用する為使用不可)
ushort	inst	(内部で使用する為使用不可)
Ushort	Offset	データセットに登録されている Tag のデータサイズ位置

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。

EgDatasetGetNext

EgDataset の次のタグを取得します:

シンタックス

```
Int32_t EgDatasetGetNext ( dslink    *dslink );
```

引 数

***dsLink** [IN] Dataset サーチ用構造体

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_TAG_IS_NULL	0x80000503	next の Tag は存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgDatasetGetCount

EgDataset の登録数を取得します:

シンタックス

```
int32_t EgDatasetGetCount ( uint32_t *pItemNum );
```

引 数

***pItemNum** [OUT] EgDataset の登録数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgDatasetGetCount コール例

```
uint32_t itemNum;
result = EgDatasetGetCount (&itemNum );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgDatasetGetEnum

EgDataset の一覧を取得します:

シンタックス

```
int32_t EgDatasetGetEnum ( EDGE_INDEX *pBuffer,
                           uint32_t   dwSize,
                           uint32_t   *pItemNum);
```

引 数

*pBuffer	[OUT] 取得したインデックス一覧
dwSize	[IN] バッファサイズ
*pItemNum	[OUT] インデックス一覧の取得数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgDatasetGetEnum コール例

```
uint32_t itemNum;
EDGE_INDEX *pBuffer;
uint32_t size;
uint32_t getNum;
EgDatasetGetCount (&itemNum ); // 登録数取得
size = sizeof(EDGE_INDEX) * itemNum;
pBuffer = malloc(size); // バッファ確保
result = EgDatasetGetEnum (pBuffer, size, &getNum ); // 一覧取得
if ( result == EDGE_SUCCESS )
    printf("成功");
free(pBuffer);
```

EgDatasetGetPropertyByIndex

EgDataset の情報を取得します(インデックス指定版):

シンタックス

```
Int32_t EgDatasetGetPropertyByIndex ( EDGE_INDEX *index,
                                     const char  *propName,
                                     void         *Value,
                                     int32_t      dwSize );
```

引 数

*index	[IN] EgDataset のインデックス
*propName	[IN] Property 名
*Value	[OUT] Property 値
dwSize	[IN] Value バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定したインデックスは登録されていない
EDGE_PARAM_INVALID	0x80000004	指定した属性名は不正

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された Property 名に対応する値を返す。指定可能な Property 名は以下の文字列を指定。

Property 名	型	内容
"Name"	string	データセット名
"ItemNum"	UInt16_t	登録タグ数

EgServiceGetCount

EgService の登録数を取得します:

シンタックス

```
int32_t EgServiceGetCount ( uint32_t *pItemNum );
```

引 数

***pItemNum** [OUT] EgService の登録数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgServiceGetCount コール例

```
uint32_t itemNum;  
result = EgServiceGetCount (&itemNum );  
if ( result == EDGE_SUCCESS )  
    printf("成功");
```

EgServiceGetEnum

EgService の一覧を取得します:

シンタックス

```
int32_t EgServiceGetEnum ( EDGE_INDEX *pBuffer,
                           uint32_t   dwSize,
                           uint32_t   *pItemNum);
```

引 数

***pBuffer** [OUT] 取得したインデックス一覧

dwSize [IN] バッファサイズ

***pItemNum** [OUT] インデックス一覧の取得数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgServiceGetEnum コール例

```
uint32_t itemNum;
EDGE_INDEX *pBuffer;
uint32_t size;
uint32_t getNum;
EgServiceGetCount (&itemNum ); // 登録数取得
size = sizeof(EDGE_INDEX) * itemNum;
pBuffer = malloc(size); // バッファ確保
result = EgServiceGetEnum (pBuffer, size, &getNum ); // 一覧取得
if ( result == EDGE_SUCCESS )
    printf("成功");
free(pBuffer);
```

EgServiceGetPropertyByIndex

EgService の情報を取得します(インデックス指定版):

シンタックス

```
Int32_t EgServiceGetPropertyByIndex ( EDGE_INDEX *index,
                                     const char  *propName,
                                     void        *Value,
                                     int32_t     dwSize );
```

引 数

*index	[IN] EgService のインデックス
*propName	[IN] Property 名
*Value	[OUT] Property 値
dwSize	[IN] Value バッファサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定したインデックスは登録されていない
EDGE_PARAM_INVALID	0x80000004	指定した属性名は不正

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された Property 名に対応する値を返す。指定可能な Property 名は以下の文字列を指定。

Property 名	型	内容
"Name"	char[]	EgService 名
"Path"	char[]	実行モジュールファイルパス
"Arg"	char[]	起動引数
"NodeName"	char[]	ノード名
"MailboxName"	char[]	コンテナ間通信用メールボックス名

EgFW_AddTagTrigger

サービスコンテナにタグトリガーを登録します:

シンタックス

```
Int32_t EgFW_AddTagTrigger ( const char *pTagName );
```

引 数

***pTagName** [IN] タグ名

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_FW_XML_FORMAT_ERROR	0x80001003	引数が NULL、または空文字、または文字数オーバー
EDGE_FW_OBJECT_SIZE_OVER	0x80001008	タグトリガー登録上限
EDGE_FW_DUPLICATE_OBJECT	0x80001007	タグトリガーとして既に登録済み
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	タグが存在しない

上記以外は 3.4. エラーコードリストを参照してください。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgDecode

バイナリデータ(バイト配列)から指定のフォーマットに基づいたバイナリデータを取得します:

シンタックス

```
Int32_t EgDecode ( void *dst,
                   size_t dstsize,
                   const void *src,
                   size_t srcsize,
                   const char *addrformat,
                   size_t *actualsize);
```

引 数

*dst	[OUT] 取得バイト配列
dstsize	[IN] 取得バイト配列サイズ
*src	[IN] 元バイト配列
srcsize	[IN] 元バイト配列サイズ
*addrformat	[IN] アドレスフォーマット

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータ不正
EGDE_COMMON_DECODE_FORMAT_CMPIDENT_ERR	0x80002001	アドレス書式不正
EGDE_COMMON_DECODE_FORMAT_PRIOD1_ERR	0x80002002	Decode 時のアドレス書式 (format)が不正(ピリオド指定があるべきところがない)
EGDE_COMMON_DECODE_FORMAT_PRIOD2_ERR	0x80002003	Decode 時のアドレス書式 (format)が不正(ピリオド後の指定がない)

上記以外は 3.4. エラーコードリストを参照してください。

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- アドレス書式は「[表 5 アドレス表記](#)」を参照してください。
- 引数で当たられたバイナリ(バイト配列)・フォーマット文字列が空の場合は null を返します。

EgEncode

Address format に則ってバイナリデータ列の所定位置にデータを書き出します:

シンタックス

```
void EgEncode ( void          *dst,
                size_t        dstsize,
                const void    *src,
                size_t        srcsize,
                const char     *addrformat,
                size_t         *actualsize);
```

引 数

*dst	[OUT] 書き込み先配列
dstsize	[IN] 書き込み先配列のバイトサイズ
*src	[IN] 書き込みたい変数
srcsize	[IN] 変数のバイトサイズ
*addrformat	[IN] フォーマット(EgDecode の説明参照)
*actualsize	[OUT] 書き込まれたバイトサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータ不正
EGDE_COMMON_DECODE_FORMAT_CMPIDENT_ERR	0x80002001	アドレス書式不正
EGDE_COMMON_DECODE_FORMAT_PRIOD1_ERR	0x80002002	Encode 時のアドレス書式 (format)が不正(ピリオド指定があるべきところがない)
EGDE_COMMON_DECODE_FORMAT_PRIOD2_ERR	0x80002003	Encode 時のアドレス書式 (format)が不正(ピリオド後の指定がない)

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- アドレス書式は「[表 5 アドレス表記](#)」を参照してください。

EgMakeAddress

オフセットとサイズの情報から address 書式を作成します:

シンタックス

```
char* EgMakeAddress (  int32_t    offset,  
                       size_t     size,  
                       char*      *buff,  
                       size_t     buffsize);
```

引 数

<i>offset</i>	[IN] オフセットバイト値
<i>size</i>	[IN] データバイトサイズ値
<i>*buff</i>	[OUT] アドレス書式文字列の格納先
<i>buffsize</i>	[IN] buff のバイトサイズ

戻り値

アドレス文字列（生成できない場合は NULL）

注釈

- アドレス書式は「[表 5 アドレス表記](#)」を参照してください。

EgParseAddress

address 書式からオフセットとサイズに分解します:

シンタックス

```
int32_t EgParseAddress ( const char    *addrformat,
                        int32_t        *offset,
                        size_t         *size,
                        int16_t        *bit,
                        uint8_t        *typemode);
```

引 数

*addressformat	[IN] 元になるアドレス書式文字列
*offset	[OUT] 取り出せたオフセットバイト値
*size	[OUT] 取り出せたバイトサイズ値
*bit	[OUT] 取り出せたビット番号値
*typemode	[OUT] 取り出せたデータ型区分番号

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータ不正
EGDE_COMMON_DECODE_FORMAT_CMPIDENT_ERR	0x80002001	アドレス書式不正
EGDE_COMMON_DECODE_FORMAT_PRIOD1_ERR	0x80002002	アドレス書式(format)が不正(ピリオド指定があるべきところがない)
EGDE_COMMON_DECODE_FORMAT_PRIOD2_ERR	0x80002003	アドレス書式(format)が不正(ピリオド後の指定がない)
EGDE_COMMON_DECODE_FORMAT_PRIOD3_ERR	0x80002004	アドレス書式(format)が不正(ピリオド値下限オーバー)
EGDE_COMMON_DECODE_FORMAT_PRIOD4_ERR	0x80002005	アドレス書式(format)が不正(ピリオド値上限オーバー)

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- アドレス書式は「[表 5 アドレス表記](#)」を参照してください。

3.3. API 関数(.Net)

C#を例に記載します：

edge_API Class

C#でのAPI ライブラリは、edge_API クラスのメンバ関数と呼び出します：

コールサンプル

edge_API インスタンスコール例

```
edge_API EGAPI = new edge_API();  
EGAPI.EgTagWrite("TestTagFlag", (bool>true);
```

EgInit

API ライブラリの初期化をします:

シンタックス

```
int EgInit(edge_API.EDGECONFIG Config );
```

引 数

Config

[IN] edge フレームワークの動作設定構造体
デフォルト設定で良い場合は指定なしで指定します。

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

EDGECONFIG 構造体

```
public struct EDGECONFIG
{
    public byte bDebug;           // デバッグフラグ
    public byte bReadXmlf;       // サービスコンフィグファイルを読むか読まないかのフラグ 0:読まない, 1:読む
    public ushort shPriority;     // メールボックススレッド プライオリティ
    public InstanceMessageHandler mInstMessFunc;
    public string Path;          // サービスプロセス/DLL ファイルパス
};
```

EDGECONFIG 構造体 フィールド

型	シンボル	内容
byte	bDebug	デバッグフラグ(デフォルト false)
byte	bReadXmlf	(内部で使用の為使用不可)
ushort	shPriority	サービス間通信用メールボックスのプライオリティ指定 (1~254) デフォルト 150)
InstanceMessageHandler	mInstMessFunc	ユーザメッセージハンドラコールバック関数
string	Path	サービスプロセスファイルパス

ユーザメッセージハンドラ(コールバック関数)指定

コールバック関数とは、他の関数に引数として渡される関数で、外側の関数で何らかの処理やアクションを実行します。このコールバック関数の原理を利用して、ユーザメッセージハンドラ関数(ユーザ定義関数)をEDGECONFIG 構造体変数に指定します。ユーザメッセージハンドラの作成方法については、「RT-edge サービス作成マニュアル」を参照してください。

コールサンプル

EgInit コール例

```
edge_API EGAPI = new edge_API();           //インスタンス作成
edge_API.EDGECONFIG config = new edge_API.EDGECONFIG();
config = EGAPI.EDGE_CONFIG_DEFAULT;        //初期値設定
config.mInstMessFunc = UserMessageHandler;  //ユーザメッセージハンドラ追加

int ret = EGAPI.EgInit(config);             //イニシャライズ実行
if (ret == 0)
    printf("成功");
```

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- EgInit を使用したサービスプロセスは終了時に EgFinalize をコールする必要があります。

EgFinalize

edge フレームワークを終了します:

シンタックス

```
int EgFinalize ( void );
```

引 数

なし

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_FW_INIT_ERROR	0x80001001	フレームワークとして動作不可

コールサンプル

EgFinalize コール例

```
edge_API EGAPI = new edge_API();           //インスタンス作成
edge_API.EDGECONFIG config = new edge_API.EDGECONFIG();
config = EGAPI.EDGE_CONFIG_DEFAULT;        //初期値設定
config.mInstMessFunc = UserMessageHandler;  //ユーザメッセージハンドラ追加

int result = EGAPI.EgInit(config);           //イニシャライズ実行
:
result = EgFinalize( );                      //ファイナライズ実行
```

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- EgFinalize は edge フレームワーク処理スレッドを削除します。このため edge フレームワークが処理を行っている edge メッセージハンドラ関数内で EgFinalize をコールすることはできませんのでご注意ください。edge メッセージハンドラ処理が停止してしまいます。この関数はアプリケーションの終了時にコールされる必要があります。

EgMailboxCreateEx

EgMailBox を生成します:

シンタックス

```
int EgMailboxCreateEx ( string      Name,
                        uint        uiLength,
                        uint        uiNum,
                        ref RTCDMEMINFO stMemInfo);
```

引 数

Name	[IN] EgMailBox の名前(半角英数字 8 文字まで)
uiLength	[IN] レコード長(バイト) 設定範囲:1 以上
uiNum	[IN] レコード数 設定範囲:2 以上
stMemInfo	[OUT](ref) メールボックスメモリ情報

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

RTCDMEMINFO 構造体

```
public struct RTCDMEMINFO
{
    public IntPtr pRtcd;           // メモリポインタ(本体)
    public ulong mem_handle;       // メモリハンドル
    public ulong sem1_handle;      // 排他用セマフォハンドル
    public ulong sem2_handle;      // 書込通知用セマフォハンドル
    public string name;            // 名称(最大 10 文字まで)
}
```

RTCDMEMINFO 構造体 フィールド

型	シンボル	内容
IntPtr	pRtcd	メモリポインタ
ulong	mem_handle	(内部で使用 メモリハンドル)
ulong	sem1_handle	(内部で使用 排他用セマフォハンドル)
ulong	sem2_handle	(内部で使用 書込通知用セマフォハンドル)
string	Name	名称(最大 10 文字まで)

コールサンプル

EgMailboxCreateEx コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
RTCDMEMINFO stMemInfo = new RTCDMEMINFO();
//メールボックス作成
int result = EGAPI.EgMailboxCreateEx ("mbox1",2048,1024,ref stMemInfo);
if ( result == EDGE_SUCCESS )
    printf("成功");
```

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgMailboxDelete

EgMailBox を削除します:

シンタックス

```
int EgMailboxDelete ( RTCDMEMINFO stMemInfo );
```

引 数

stMemInfo

[IN] メールボックスメモリ情報
(EgMailboxCreateEx にて取得したメモリ情報構造体)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_FW_INIT_ERROR	0x80001001	フレームワークとして動作不可

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxDelete コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
RTCDMEMINFO stMemInfo = new RTCDMEMINFO();
//メールボックス作成
int result = EGAPI.EgMailboxCreateEx ("mbox1",2048,1024,ref stMemInfo);
:
result = EGAPI.EgMailboxDelete( &rtcdi );
```

EgMailboxOpen

C#では Open を呼び出しません。EgMailboxSend/EgMailboxReceive の際に自動で Open します:

EgMailboxClose

C#では Close を呼び出しません:

EgMailboxSend

EgMailBox にメッセージを送信します:

シンタックス

```
int EgMailboxSend( string ReceiverName,
                  string SenderName,
                  int MessageNo,
                  byte[] arg );
```

引 数

ReceiverName	[IN] 送信先 EgMailBox 名
SenderName	[IN] 送信元 EgMailBox 名
MessageNo	[IN] 送信するメッセージ番号
arg	[IN] 送信するメッセージデータ(byte 配列)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxSend コール例

```
edge_API EGAPI = new edge_API(); //インスタンス作成
:
byte[] byteArr = new byte[128];
byteArr[0] = 1; //データ作成
result = EGAPI.EgMailboxSend( "tgtMbox", "myServ", 100, byteArr ); //メール送信
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgMailboxSendSC

サービスコンテナ間通信用 EgMailBox にメッセージを送信します。他サービスにメッセージを送る場合に使用します:

シンタックス

```
int EgMailboxSend( string ReceiverName,
                  string SenderName,
                  int MessageNo,
                  byte[] arg = null);
```

引 数

ReceiverName	[IN] 送信先 EgMailBox 名
SenderName	[IN] 送信元 EgMailBox 名
MessageNo	[IN] 送信するメッセージ番号
arg	[IN] 送信するメッセージデータ(byte 配列) 指定なし : null

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxSend コール例

```
edge_API EGAPI = new edge_API(); //インスタンス作成
:
byte[] byteArr = new byte[128];
byteArr[0] = 1; //データ作成
result = EGAPI.EgMailboxSend( "tgtServ", "myServ", 100, byteArr ); //メール送信
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgMailboxReceive

EgMailBox からメッセージを受信します:

シンタックス

```
int EgMailboxReceive( string      ReceiverName,
                      ref string  SenderName,
                      ref int     MessageNo,
                      ref byte[]  Params,
                      uint        uitimeout);
```

引 数

ReceiverName	[IN] 受信 EgmailBox 名
SenderName	[OUT](ref) 送信元 EgMailBox 名が格納箱(string)
MessageNo	[OUT](ref) 受信メッセージ番号が格納箱(int)
Params	[OUT](ref) 受信メッセージデータが格納箱(byte 配列)
uitimeOut	[IN] タイムアウト指定(ミリ秒)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgMailboxReceive コール例

```
edge_API EGAPI = new edge_API(); //インスタンス作成
:
int  MsgNo = 0; //メッセージ番号 (不要の場合はテンポラリー)
string from = ""; //送信メールアドレス名(不要の場合はテンポラリー)
byte[] data = new byte[1024];
uint  Timeout = 5000; //タイムアウト時間(ミリ秒)
//メールボックス受信
result = EGAPI.EgMailboxReceive( "myServ" ,ref from ,ref MsgNo ,ref data ,Timeout );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgMailboxGetInfo

C#では使用できません:

EgMailboxGetProperty

EgMailbox の属性を読み込みます:

シンタックス

```
int EgMailboxGetProperty ( string      stName,
                           string      stPropName,
                           ref object  value );
```

引 数

stName [IN] EgMailbox 名

stPropName [IN] 要求する属性名

value [OUT](ref) 取得した属性値

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgMailboxGetProperty コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int      colSize; // ColSize を読み出す
result = EGAPI.EgMailboxGetProperty ("tgtServ","ColSize",ref colSize);
if ( result == EDGE_SUCCESS )
    printf("成功");
```

注釈

- 指定可能な属性名は以下のとおりです。

PropName	型	内容
"ColSize"	uint32_t	データ格納メールボックス 1 レコードサイズ
"RowSize"	uint32_t	データ格納メールボックスレコード数
"Name"	String	メールボックス名文字列を取得します

EgMailboxGetPropertyByIndex

EgMailbox の属性を読み込みます(インデックス指定版):

シンタックス

```
int EgMailboxGetPropertyByIndex ( EGOBJINDEX index,
                                string      stPropName,
                                ref object  value);
```

引 数

index [IN] EgMailbox のインデックス

stPropName [IN] 要求する属性名

value [OUT](ref) 取得した属性値

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定したインデックスは登録されていない
EDGE_PARAM_INVALID	0x80000004	指定した属性名は不正

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgMailboxGetProperty コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int      colSize; // ColSize を読み出す
edge_API.EGOBJINDEX index; // NOTE: インデックスは一覧取得 API で取得してください
result = EGAPI.EgMailboxGetPropertyByIndex (index, "ColSize", ref colSize);
if ( result == EDGE_SUCCESS )
    printf("成功");
```

注釈

- 指定可能な属性名は以下のとおりです。

PropName	型	内容
"ColSize"	uint32_t	データ格納メールボックス 1 レコードサイズ
"RowSize"	uint32_t	データ格納メールボックスレコード数
"Name"	String	メールボックス名文字列を取得します

EgMailboxGetCount

EgMailbox の登録数を取得します:

シンタックス

```
int EgMailboxGetCount ( ref uint itemNum );
```

引 数

itemNum [OUT](ref) EgMailbox の登録数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgMailboxGetCount コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
uint    itemNum = 0;
result = EGAPI.EgMailboxGetCount (ref itemNum);
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgMailBoxGetEnum

EgMailbox の一覧を取得します:

シンタックス

```
int EgMailBoxGetEnum ( ref EGOBJINDEX[] indexArray,  
                      ref uint      itemNum);
```

引 数

indexArray [OUT](ref) 取得したインデックス一覧

itemNum [OUT](ref) インデックス一覧の取得数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgMailBoxGetEnum コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成  
:  
uint      itemNum = 0;  
EGAPI.EgMailboxGetCount (ref itemNum); // 登録数取得  
edge_API.EGOBJINDEX[] idxArray = new edge_API.EGOBJINDEX[itemNum];  
uint      getNum = 0;  
result = EGAPI.EgMailboxGetEnum (ref idxArray, ref getNum); // 一覧取得  
if ( result == EDGE_SUCCESS )  
    printf("成功");
```

EgTagCreateEx

EgTag を生成します:

シンタックス

```
int EgTagCreateEx ( string      stName,
                   ushort     Type,
                   string      stSource,
                   string      stComment );
```

引 数

<i>stName</i>	[IN] EgTag 名(半角英数時 48 バイトまで)
<i>Type</i>	[IN] 型(後述)
<i>stSource</i>	[IN] データソース
<i>stComment</i>	[IN] タグの説明文(全角半角英数 48 バイトまで)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_RTCD_ARG_DATA_SIZE_ERROR	0x80000300	指定した引数の変数サイズが既定のサイズより大きい,または小さいためバッファを壊す
EDGE_TTABLE_MEMORRY_ERROR	0x80000400	メモリの作成ができない(INtime の場合はカーネルが起動していない等)
EDGE_TTABLE_ISEXIST_TAGNAME	0x80000409	同名登録済み
EDGE_TTABLE_NOEMPTY	0x8000040E	テーブルに空きがありません。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 既に登録されているか内部的にチェックを行い、登録されている場合は作成しません。
- この API を使用して、文字列型/バイト配列型 EgTag を作成することはできません。これらのタグを作成する場合は、EgTagCreateSegment API を使用してください。
- この API を使用して、FIFO 型 EgTag を作成することはできません。この型のタグを作成する場合は、EgTagCreateFIFO API を使用してください。

コールサンプル

EgTagCreateEx コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
result = EGAPI.EgTagCreateEx ( "MyTag01" ,
                              (ushort)EGDEFINE.egTagDataType.UInt16,
                              "%D0" , "com1" );

if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagCreateSp

EgTag を生成します。既に生成済みのタグ名を指定した場合は、別名でリンクタグを生成します。:

シンタックス

```
int EgTagCreateSp ( string      stName,
                   ushort      Type,
                   string      stSource,
                   string      stComment
                   ref object   regName
                   ref ushort   isLink );
```

引 数

<i>stName</i>	[IN] EgTag 名(半角英数時 48 バイトまで)
<i>Type</i>	[IN] 型(後述)
<i>stSource</i>	[IN] データソース
<i>stComment</i>	[IN] タグの説明文(全角半角英数 48 バイトまで)
<i>regName</i>	[OUT]実際に登録されたタグ名
<i>isLink</i>	[OUT]リンクタグフラグ(0:通常タグ、1:リンクタグ)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_DUPLICATE_ERROR	0x80000418	Tag の重複登録失敗
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_RTCD_ARG_DATA_SIZE_ERROR	0x80000300	指定した引数の変数サイズが既定のサイズより大きい,または小さいためバッファを壊す
EDGE_TTABLE_MEMORRY_ERROR	0x80000400	メモリの作成ができない(INtime の場合はカーネルが起動していない等)
EDGE_TTABLE_ISEXIST_TAGNAME	0x80000409	同名登録済み
EDGE_TTABLE_NOEMPTY	0x8000040E	テーブルに空きがありません。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- リンクタグとして生成される場合、タグ名先頭に"#*"の 2 文字が自動的に付与されます。
(*は 0~9 の数字が順番に割り当てられます)
- リンクタグは最大 10 個まで生成可。
- この API を使用して、文字列型/バイト配列型 EgTag を作成することはできません。これらのタグを作成する場合は、EgTagCreateSegment API を使用してください。
- この API を使用して、FIFO 型 EgTag を作成することはできません。この型のタグを作成する場合は、EgTagCreateFIFO API を使用してください。

コールサンプル

EgTagCreateSp コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
object regName = string.Empty;
ushort isLink = 0;
result = EGAPI.EgTagCreateSp ( "MyTag01" ,
                                (ushort)EGDEFINE.egTagDataType.UInt16,
                                "%D0" , "com1", ref regName, ref isLink );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagWrite

EgTag にデータを書き込みます:

シンタックス

```
int EgTagWrite ( string      Name,
                  object      value );
```

引 数

- Name** [IN] EgTag 名
- Value** [IN] 書き込みデータ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された値を Tag 構造体内の値に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

```
EgTagWrite コール例

edge_API EGAPI = new edge_API();    //インスタンス作成
:
result = EGAPI.EgTagWrite ( "MyApp.Value1" ,(bool>true );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagRead

EgTag のデータを読み込みます:

シンタックス

```
int EgTagRead ( string      stName,
                ref object  value,
                bool        byteF = false );
```

引 数

stName	[IN] EgTag 名
Value	[OUT] 読み込みデータ
byteF	[IN] byte 配列かどうか、byte 配列で取得の場合は true を指定

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagRead コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
object dwValue1;
result = EgTagRead ( "MyApp.Value1",ref dwValue1 );
if ( result == EDGE_SUCCESS )
    printf("成功");
```


EgTagGetProperty

EgTag の属性を読み込みます:

シンタックス

```
int EgTagGetProperty ( string      stTagName,
                      string      stPropName,
                      ref object  value );
```

引 数

stTagName [IN] EgTag 名

stPropName [IN] 要求する属性名

value [OUT](ref) 取得した属性値

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgTagGetProperty コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
object  comment;                    // コメントを読み出す
result = EgTagGetProperty ("MyApp.Value1", "Comment", ref comment);
if ( result == EDGE_SUCCESS )
    printf("成功");
string stComment = (string)comment;
```

注釈

- 指定可能な属性名は以下のとおりです。

PropName	型	内容
"Type"	UInt16	タグのデータ型番号を取得します
"Size"	UInt16	タグのデータサイズ(byte)を取得します
"Comment"	String	コメント文字列を取得します
"Source"	String	アドレスソース文字列を取得します
"Address"	String	同上
"Value"		タグの値を取得します。データサイズはタグにより異なります
"RecordCount"	UInt16	(FIFO 型タグ) 登録可能なデータ数の上限を取得します
"Remain"	UInt16	(FIFO 型タグ) タグに登録されているデータ件数を取得します
"Hash"	UInt64	内部：タグのハッシュコードを取得します
"Name"	String	内部：タグ名文字列を取得します

EgTagGetCount

EgTag の登録数を取得します:

シンタックス

```
int EgTagGetCount ( ref uint itemNum );
```

引 数

itemNum [OUT](ref) EgTag の登録数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgTagGetCount コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
uint    itemNum = 0;
result = EGAPI.EgTagGetCount (ref itemNum);
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagGetEnum

EgTag の一覧を取得します:

シンタックス

```
int EgTagGetEnum ( ref EGOBJINDEX[] indexArray,
                  ref uint          itemNum);
```

引 数

indexArray [OUT](ref) 取得したインデックス一覧

itemNum [OUT](ref) インデックス一覧の取得数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgTagGetEnum コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
uint    itemNum = 0;
EGAPI.EgTagGetCount (ref itemNum); // 登録数取得
edge_API.EGOBJINDEX[] idxArray = new edge_API.EGOBJINDEX[itemNum];
uint    getNum = 0;
result = EGAPI.EgTagGetEnum (ref idxArray, ref getNum); // 一覧取得
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagWriteByIndex

EgTag にデータを書き込みます(インデックス指定版):

シンタックス

```
int EgTagWriteByIndex ( EGOBJINDEX index,
                        object value );
```

引 数

index [IN] EgTag のインデックス

Value [IN] 書き込みデータ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された値を Tag 構造体内の値に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagWriteByIndex コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
edge_API.EGOBJINDEX index; // NOTE: インデックスは一覧取得 API で取得してください
result = EGAPI.EgTagWriteByIndex ( index ,(bool>true );
if ( result == EDGE_SUCCESS )
printf("成功");
```

EgTagReadByIndex

EgTag のデータを読み込みます(インデックス指定版):

シンタックス

```
int EgTagReadByIndex (EGOBJINDEX index,
                      ref object value,
                      bool byteF = false );
```

引 数

<i>index</i>	[IN] EgTag のインデックス
<i>Value</i>	[OUT] 読み込みデータ
<i>byteF</i>	[IN] byte 配列かどうか、byte 配列で取得の場合は true を指定

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagReadByIndex コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
object dwValue1;
edge_API.EGOBJINDEX index; // NOTE: インデックスは一覧取得 API で取得してください
result = EgTagReadByIndex ( index ,ref dwValue1 );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagGetPropertyByIndex

EgTag の属性を読み込みます(インデックス指定版):

シンタックス

```
int EgTagGetPropertyByIndex (EGOBJINDEX index,
                             string stPropName,
                             ref object value);
```

引 数

index [IN] EgTag のインデックス

stPropName [IN] 要求する属性名

value [OUT](ref) 取得した属性値

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定したインデックスは登録されていない

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgTagGetPropertyByIndex コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
object comment;                     // コメントを読み出す
result = EgTagGetPropertyByIndex ("MyApp.Value1", "Comment", ref comment);
if ( result == EDGE_SUCCESS )
    printf("成功");
string stComment = (string)comment;
```

注釈

- 指定可能な属性名は以下のとおりです。

PropName	型	内容
"Type"	UInt16	タグのデータ型番号を取得します
"Size"	UInt16	タグのデータサイズ(byte)を取得します
"Comment"	String	コメント文字列を取得します
"Source"	String	アドレスソース文字列を取得します
"Address"	String	同上
"Value"		タグの値を取得します。データサイズはタグにより異なります
"RecordCount"	UInt16	(FIFO 型タグ) 登録可能なデータ数の上限を取得します
"Remain"	UInt16	(FIFO 型タグ) タグに登録されているデータ件数を取得します
"Hash"	UInt64	内部：タグのハッシュコードを取得します
"Name"	String	内部：タグ名文字列を取得します

EgTagWriteByEntry

EgTag にデータを書き込みます(インデックス指定版):

シンタックス

```
int EgTagWriteByEntry ( IntPtr entry,
                      object value );
```

引 数

entry [IN] EgTag のエントリ

Value [IN] 書き込みデータ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された値を Tag 構造体内の値に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagWriteByEntry コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
object entry; // NOTE: エントリは EgTagGetProperty で取得してください
uint dwValue1 = 0x12345678;
result = EGAPI.EgTagWriteByEntry ( (IntPtr)entry ,dwValue1);
if ( result == EDGE_SUCCESS )
    printf("成功");
```


EgTagReadByEntry

EgTag のデータを読み込みます(インデックス指定版):

シンタックス

```
int EgTagReadByEntry ( IntPtr      entry,
                      ref object  value,
                      bool        byteF = false );
```

引 数

<i>entry</i>	[IN] EgTag のエントリ
<i>Value</i>	[OUT] 読み込みデータ
<i>byteF</i>	[IN] byte 配列かどうか、byte 配列で取得の場合は true を指定

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagReadByEntry コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
object dwValue1;
object entry; // NOTE: エントリは EgTagGetProperty で取得してください
result = EgTagReadByEntry ( (IntPtr)entry ,ref dwValue1 );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagCreateSegment

文字列型/バイト配列型 EgTag を生成します。既に生成済みのタグ名を指定した場合は、別名でリンクタグを生成します。:

シンタックス

```
int EgTagCreateSegment ( string      stName,
                        ushort      Type,
                        ushort      Size,
                        string      stSource,
                        string      stComment
                        ref object   regName
                        ref ushort   isLink );
```

引 数

stName	[IN] EgTag 名(半角英数時 48 バイトまで)
Type	[IN] 型(12:文字列、13:バイト配列)
Size	[IN] 文字列型/バイト配列型 EgTag のデータサイズ
stSource	[IN] データソース
stComment	[IN] タグの説明文(全角半角英数 48 バイトまで)
regName	[OUT]実際に登録されたタグ名
isLink	[OUT]リンクタグフラグ(0:通常タグ、1:リンクタグ)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_DUPLICATE_ERROR	0x80000418	Tag の重複登録失敗
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_RTCD_ARG_DATA_SIZE_ERROR	0x80000300	指定した引数の変数サイズが既定のサイズより大きい,または小さいためバッファを壊す
EDGE_TTABLE_MEMORRY_ERROR	0x80000400	メモリの作成ができない(INtime の場合はカーネルが起動していない等)
EDGE_TTABLE_ISEXIST_TAGNAME	0x80000409	同名登録済み
EDGE_TTABLE_NOEMPTY	0x8000040E	テーブルに空きがありません。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- リンクタグとして生成される場合、タグ名先頭に"#*"の2文字が自動的に付与されます。
(*は0~9の数字が順番に割り当てられます)
- リンクタグは最大10個まで生成可。

コールサンプル

EgTagCreateSegment コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
object regName = string.Empty;
ushort isLink = 0;
result = EGAPI.EgTagCreateSegment ( "MyTag01" ,
                                   (ushort)EGDEFINE.egTagDataType.ByteArray, 64,
                                   "" , "com1", ref regName, ref isLink );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagWriteSegment

バイト配列型 EgTag にデータを書き込みます:

シンタックス

```
int EgTagWriteSegment ( string      Name ,
                        ushort      wOffset ,
                        byte[]      pBuffer );
```

引 数

Name [IN] EgTag 名

wOffset [IN] バイト配列オフセット

pBuffer [IN] 書き込みデータ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定された値をバイト配列型用 Tag が確保した領域に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagWriteSegment コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
byte[] byArray = new byte[16];
:
result = EGAPI.EgTagWriteSegment ( "MyApp.Value1" ,32 ,byArray );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagReadSegment

バイト配列型 EgTag のデータを読み込みます:

シンタックス

```
int EgTagReadSegment ( string      stName ,
                        ushort     wOffset ,
                        ref byte[]  pBuffer );
```

引 数

<i>stName</i>	[IN] EgTag 名
<i>wOffset</i>	[IN] バイト配列オフセット
<i>pBuffer</i>	[OUT] 読み込みデータ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagReadSegment コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
byte[] byArray = new byte[16];
result = EgTagReadSegment ( "MyApp.Value1" ,32 ,ref byArray );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagWriteSegmentByIndex

バイト配列型 EgTag にデータを書き込みます(インデックス指定版):

シンタックス

```
int EgTagWriteSegmentByIndex( EGOBJINDEX index,
                              ushort wOffset ,
                              byte[] pBuffer );
```

引 数

index [IN] EgTag のインデックス
wOffset [IN] バイト配列オフセット
pBuffer [IN] 書き込みデータ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定された値をバイト配列型用 Tag が確保した領域に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagWriteSegmentByIndex コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
edge_API.EGOBJINDEX index; // NOTE: インデックスは一覧取得 API で取得してください
byte[] byArray = new byte[16];
:
result = EGAPI.EgTagWriteSegmentByIndex ( index ,32 ,byArray );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagReadSegmentByIndex

バイト配列型 EgTag のデータを読み込みます(インデックス指定版):

シンタックス

```
int EgTagReadSegmentByIndex (  EGOBJINDEX    index,
                                ushort          wOffset ,
                                ref byte[]      pBuffer );
```

引 数

index [IN] EgTag のインデックス

wOffset [IN] バイト配列オフセット

pBuffer [OUT] 読み込みデータ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagReadSegmentByIndex コール例

```
edge_API EGAPI = new edge_API(); //インスタンス作成
:
int result;
byte[] byArray = new byte[16];
edge_API.EGOBJINDEX index; // NOTE: インデックスは一覧取得 API で取得してください
result = EgTagReadSegmentByIndex ( index ,32 ,ref byArray );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagWriteSegmentByEntry

バイト配列型 EgTag にデータを書き込みます(インデックス指定版):

シンタックス

```
int EgTagWriteSegmentByEntry( IntPtr entry,
                             ushort wOffset ,
                             byte[] pBuffer );
```

引 数

index [IN] EgTag のエントリ

wOffset [IN] バイト配列オフセット

pBuffer [IN] 書き込みデータ

戻り値

ステータス値

	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定された値をバイト配列型用 Tag が確保した領域に書き込みます。
- 指定されたデータサイズが Tag のデータサイズより大きい場合はエラー終了になります。

コールサンプル

EgTagReadSegmentByIndex コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
object entry; // NOTE: エントリは EgTagGetProperty で取得してください
byte[] byArray = new byte[16];
:
result = EGAPI.EgTagWriteSegmentByEntry ( (IntPtr)entry ,32 ,byArray );
if ( result == EDGE_SUCCESS )
    printf("成功");
```


EgTagReadSegmentByEntry

バイト配列型 EgTag のデータを読み込みます(インデックス指定版):

シンタックス

```
int EgTagReadSegmentByEntry ( IntPtr      entry,
                             ushort      wOffset ,
                             ref byte[]  pBuffer );
```

引 数

- index* [IN] EgTag のエントリ
- wOffset* [IN] バイト配列オフセット
- pBuffer* [OUT] 読み込みデータ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagReadSegmentByEntry コール例

```
edge_API EGAPI = new edge_API(); //インスタンス作成
:
int result;
byte[] byteArray = new byte[16];
object entry; // NOTE: エントリは EgTagGetProperty で取得してください
result = EgTagReadSegmentByEntry ( (IntPtr)entry ,32 ,ref byteArray );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagCreateFIFO

FIFO 型 EgTag を生成します。既に生成済みのタグ名を指定した場合は、別名でリンクタグを生成します。:

シンタックス

```
int EgTagCreateFIFO ( string      stName,
                     ushort      RecordSize,
                     ushort      RecordCount,
                     bool        Reserved1_LetFalse,
                     string      stComment,
                     ref object   regName,
                     ref ushort   isLink );
```

引 数

stName	[IN] EgTag 名(半角英数時 48 バイトまで)
RecordSize	[IN] タグに登録する 1 件のデータの長さ(バイト単位)
RecordCount	[IN] タグに登録可能な最大データ件数
Reserved1_LetFalse	[IN] 未使用(false)
stComment	[IN] タグの説明文(全角半角英数 48 バイトまで)
regName	[OUT] 実際に登録されたタグ名
isLink	[OUT] リンクタグフラグ(0:通常タグ、1:リンクタグ)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_DUPLICATE_ERROR	0x80000418	Tag の重複登録失敗
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_RTCD_ARG_DATA_SIZE_ERROR	0x80000300	指定した引数の変数サイズが既定のサイズより大きい,または小さいためバッファを壊す
EDGE_TTABLE_MEMORRY_ERROR	0x80000400	メモリの作成ができない(INtime の場合はカーネルが起動していない等)
EDGE_TTABLE_NOEMPTY	0x8000040E	テーブルに空きがありません。

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- リンクタグとして生成される場合、タグ名先頭に"#*"の 2 文字が自動的に付与されます。
(*は 0~9 の数字が順番に割り当てられます)
- リンクタグは最大 10 個まで生成可。

コールサンプル

EgTagCreateFIFO コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
object regName = string.Empty;
ushort isLink = 0;
result = EGAPI.EgTagCreateFIFO ("MyApp.FIFO1", 4, 1024, false,
    "FIFO(4 byte * 1024 record)", ref regName, ref isLink );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagEnqueueFIFO

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます:

シンタックス

```
int EgTagEnqueueFIFO <T> ( string      stName,
                           ref T       pBuffer,
                           ref ushort  wRemain );
```

引 数

stName [IN] EgTag 名

pBuffer [IN] 登録するデータ
データ型 T は、.NET の blittable 型のみ使用可能です。

wRemain [OUT] 登録が完了した時点での、タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した
-	0xFFFFFFFF	T に未対応のデータ型が指定された

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータのサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、古いデータから順に上書き(破壊)される動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- pBuffer に長さ 0 の配列を指定すると、現在の書き込み位置にあるデータを登録できます。書き込み位置のデータを更新するには、EgTagWrite API を使用してください。

コールサンプル

EgTagEnqueueFIFO コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
byte[] writeData = new byte[] {1, 2, 3, 4};
ushort remain = 0;
int result = EGAPI.EgTagEnqueueFIFO ( "MyApp.FIFO1", ref writeData, ref remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagEnqueueFIFOByIndex

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます(インデックス指定版):

シンタックス

```
int EgTagEnqueueFIFOByIndex <T> ( EGOBJINDEX index,
                                   ref T      pBuffer,
                                   ref ushort wRemain );
```

引 数

<i>index</i>	[IN] EgTag のインデックス
<i>pBuffer</i>	[IN] 登録するデータ データ型 T は、.NET の blittable 型のみ使用可能です。
<i>wRemain</i>	[OUT] 登録が完了した時点での、タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した
-	0xFFFFFFFF	T に未対応のデータ型が指定された

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータのサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、古いデータから順に上書き(破壊)される動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- pBuffer に長さ 0 の配列を指定すると、現在の書き込み位置にあるデータを登録できます。書き込み位置のデータを更新するには、EgTagWriteByIndex API を使用してください。

コールサンプル

EgTagEnqueueFIFOByIndex コール例

```
edge_API EGAPI = new edge_API(); //インスタンス作成
:
edge_API.EGOBJINDEX index; // NOTE: インデックスは一覧取得 API で取得してください
int writeData = 0x04030201;
ushort remain = 0;
int result = EGAPI.EgTagEnqueueFIFOByIndex ( index, ref writeData, ref remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagEnqueueFIFOByEntry

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます(Entry 指定版):

シンタックス

```
int EgTagEnqueueFIFOByEntry <T> ( IntPtr    entry,
                                   ref T      pBuffer,
                                   ref ushort wRemain );
```

引 数

entry [IN] EgTag のエントリ

pBuffer [IN] 登録するデータ
データ型 T は、.NET の blittable 型のみ使用可能です。

wRemain [OUT] 登録が完了した時点での、タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した
-	0xFFFFFFFF	T に未対応のデータ型が指定された

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータのサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、古いデータから順に上書き(破壊)される動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- pBuffer に長さ 0 の配列を指定すると、現在の書き込み位置にあるデータを登録できます。書き込み位置のデータを更新するには、EgTagWriteByEntry API を使用してください。

コールサンプル

EgTagEnqueueFIFOByEntry コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
object entry; // NOTE: エントリは EgTagGetProperty で取得してください
float writeData = 1.2345F;
ushort remain = 0;
int result = EGAPI.EgTagEnqueueFIFOByEntry ( (IntPtr)entry, ref writeData, ref
remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagEnqueueRingFIFO

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます(Ring 動作版):

シンタックス

```
int EgTagEnqueueRingFIFO <T> ( string      stName,
                                ref T      pBuffer,
                                ref ushort  wRemain );
```

引 数

stName	[IN] EgTag 名
pBuffer	[IN] 登録するデータ データ型 T は、.NET の blittable 型のみ使用可能です。
wRemain	[OUT] 登録が完了した時点での、タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した
-	0xFFFFFFFF	T に未対応のデータ型が指定された

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータのサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、一番古いデータが読み捨てられる動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- pBuffer に長さ 0 の配列を指定すると、現在の書き込み位置にあるデータを登録できます。書き込み位置のデータを更新するには、EgTagWrite API を使用してください。

コールサンプル

EgTagEnqueueRingFIFO コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
byte[] writeData = new byte[] {1, 2, 3, 4};
ushort remain = 0;
int result = EGAPI.EgTagEnqueueRingFIFO ( "MyApp.FIFO1", ref writeData, ref remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagEnqueueRingFIFOByIndex

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます(Ring 動作・インデックス指定版):

シンタックス

```
int EgTagEnqueueRingFIFOByIndex <T> ( EGOBJINDEX index,
                                       ref T      pBuffer,
                                       ref ushort wRemain );
```

引 数

index [IN] EgTag のインデックス

pBuffer [IN] 登録するデータ
データ型 T は、.NET の blittable 型のみ使用可能です。

wRemain [OUT] 登録が完了した時点での、タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した
-	0xFFFFFFFF	T に未対応のデータ型が指定された

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータのサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、一番古いデータが読み捨てられる動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- pBuffer に長さ 0 の配列を指定すると、現在の書き込み位置にあるデータを登録できます。書き込み位置のデータを更新するには、EgTagWriteByIndex API を使用してください。

コールサンプル

EgTagEnqueueRingFIFOByIndex コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
edge_API.EGOBJINDEX index; // NOTE: インデックスは一覧取得 API で取得してください
int writeData = 0x04030201;
ushort remain = 0;
int result = EGAPI.EgTagEnqueueRingFIFOByIndex ( index, ref writeData, ref remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```


EgTagEnqueueRingFIFOByEntry

FIFO 型 EgTag に 1 件のデータを登録し、書き込み位置を進めます(Ring 動作・Entry 指定版):

シンタックス

```
int EgTagEnqueueRingFIFOByEntry <T> ( IntPtr      entry,
                                       ref T        pBuffer,
                                       ref ushort    wRemain );
```

引 数

entry [IN] EgTag のエントリ

pBuffer [IN] 登録するデータ
データ型 T は、.NET の blittable 型のみ使用可能です。

wRemain [OUT] 登録が完了した時点での、タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_OVERRUN	0x80006002	タグに登録できるデータ件数を超過した
-	0xFFFFFFFF	T に未対応のデータ型が指定された

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータのサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録できるデータ件数を超過して登録を行うと、一番古いデータが読み捨てられる動作となります。このとき、戻り値は EDGE_FIFO_OVERRUN となります。
- pBuffer に長さ 0 の配列を指定すると、現在の書き込み位置にあるデータを登録できます。書き込み位置のデータを更新するには、EgTagWriteByEntry API を使用してください。

コールサンプル

EgTagEnqueueRingFIFOByEntry コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
object entry; // NOTE: エントリは EgTagGetProperty で取得してください
float writeData = 1.2345F;
ushort remain = 0;
int result = EGAPI.EgTagEnqueueRingFIFOByEntry ( (IntPtr)entry, ref writeData, ref
remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagDequeueFIFO

FIFO 型 EgTag からデータを 1 件取得し、読出し位置を進めます:

シンタックス

```
int EgTagDequeueFIFO <T> ( string      stName,
                           ref T      pBuffer,
                           ref ushort wRemain );
```

引 数

stName	[IN] EgTag 名
pBuffer	[OUT] データ取得先 データ型 T は、.NET の blittable 型のみ使用可能です。
wRemain	[OUT] 取得が完了した時点での、タグに登録されている残りデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_UNDERRUN	0x80006001	タグに登録されたデータがない
-	0xFFFFFFFF	T に未対応のデータ型が指定された

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータのサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録されたデータがない状態で取得を行うと、読出し位置にある不明なデータを読み出し、読出し位置を進めます。このとき、戻り値は EDGE_FIFO_UNDERRUN です。
- pBuffer に長さ 0 の配列を指定すると、現在読出し位置にあるデータを読み捨てます。
- 読出し位置を更新せずデータを取得するには、EgTagRead API をご利用下さい。

コールサンプル

EgTagDequeueFIFO コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
byte[] readBuf = new byte[4];
ushort remain = 0;
int result = EGAPI.EgTagDequeueFIFO ( "MyApp.FIFO1", ref readBuf, ref remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagDequeueFIFOByIndex

FIFO 型 EgTag からデータを 1 件取得し、読出し位置を進めます(インデックス指定版):

シンタックス

```
int EgTagDequeueFIFOByIndex <T> ( EGOBJINDEX index,
                                   ref T      pBuffer,
                                   ref ushort wRemain );
```

引 数

<i>index</i>	[IN] EgTag のインデックス
<i>pBuffer</i>	[OUT] データ取得先 データ型 T は、.NET の blittable 型のみ使用可能です。
<i>wRemain</i>	[OUT] 取得が完了した時点での、タグに登録されている残りデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_UNDERRUN	0x80006001	タグに登録されたデータがない
-	0xFFFFFFFF	T に未対応のデータ型が指定された

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータのサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録されたデータがない状態で取得を行うと、読出し位置にある不明なデータを読み出し、読出し位置を進めます。このとき、戻り値は EDGE_FIFO_UNDERRUN です。
- pBuffer に長さ 0 の配列を指定すると、現在読出し位置にあるデータを読み捨てます。
- 読出し位置を更新せずデータを取得するには、EgTagReadByIndex API をご利用下さい。

コールサンプル

EgTagDequeueFIFOByIndex コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
edge_API.EGOBJINDEX index; // NOTE: インデックスは一覧取得 API で取得してください
int readBuf = 0;
ushort remain = 0;
int result = EGAPI.EgTagDequeueFIFOByIndex ( index, ref readBuf, ref remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagDequeueFIFOByEntry

FIFO 型 EgTag からデータを 1 件取得し、読出し位置を進めます(Entry 指定版):

シンタックス

```
int EgTagDequeueFIFOByEntry <T> ( IntPtr      entry,
                                   ref T        pBuffer,
                                   ref ushort   wRemain );
```

引 数

entry	[IN] EgTag のエントリ
pBuffer	[OUT] データ取得先 データ型 T は、.NET の blittable 型のみ使用可能です。
wRemain	[OUT] 取得が完了した時点での、タグに登録されている残りデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_FIFO_UNDERRUN	0x80006001	タグに登録されたデータがない
-	0xFFFFFFFF	T に未対応のデータ型が指定された

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- 指定されたデータのサイズが Tag のレコードサイズより大きい場合はエラー終了になります。
- タグに登録されたデータがない状態で取得を行うと、読出し位置にある不明なデータを読み出し、読出し位置を進めます。このとき、戻り値は EDGE_FIFO_UNDERRUN です。
- pBuffer に長さ 0 の配列を指定すると、現在読出し位置にあるデータを読み捨てます。
- 読出し位置を更新せずデータを取得するには、EgTagReadByEntry API をご利用下さい。

コールサンプル

EgTagDequeueFIFOByEntry コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
object entry; // NOTE: エントリは EgTagGetProperty で取得してください
float readBuf = 0;
ushort remain = 0;
int result = EGAPI.EgTagDequeueFIFOByEntry ( (IntPtr)entry, ref readBuf, ref remain );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagClearFIFO

FIFO 型 EgTag に登録されたデータを全て消去し、書き込み位置・読出し位置をリセットします:

シンタックス

```
int EgTagClearFIFO ( string stName );
```

引 数

stName [IN] EgTag 名

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- データ領域のクリアは行われません。

コールサンプル

EgTagClearFIFO コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result = EGAPI.EgTagClearFIFO ( "MyApp.FIFO1" );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagClearFIFOByIndex

FIFO 型 EgTag に登録されたデータを全て消去し、書き込み位置・読出し位置をリセットします(インデックス指定版):

シンタックス

```
int EgTagClearFIFOByIndex ( EGOBJINDEX index );
```

引 数

index

[IN] EgTag のインデックス

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- データ領域のクリアは行われません。

コールサンプル

EgTagClearFIFOByIndex コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
edge_API.EGOBJINDEX index; // NOTE: インデックスは一覧取得 API で取得してください
:
int result = EGAPI.EgTagClearFIFOByIndex ( index );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagClearFIFOByEntry

FIFO 型 EgTag に登録されたデータを全て消去し、書き込み位置・読出し位置をリセットします(Entry 指定版):

シンタックス

```
int EgTagClearFIFOByEntry ( IntPtr entry );
```

引 数

entry [IN] EgTag のエントリ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- データ領域のクリアは行われません。

コールサンプル

EgTagClearFIFOByEntry コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
object entry; // NOTE: エントリは EgTagGetProperty で取得してください
:
int result = EGAPI.EgTagClearFIFOByEntry ( (IntPtr)entry );
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagGetRemainFIFO

FIFO 型 EgTag に登録されているデータ件数を取得します:

シンタックス

```
int EgTagGetRemainFIFO ( string stName,
                        ref ushort wRemain );
```

引 数

stName [IN] EgTag 名

wRemain [OUT] タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagGetRemainFIFO コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
object regName = string.Empty;
ushort isLink = 0;
result = EGAPI.EgTagCreateFIFO ("MyApp.FIFO1", 4, 1024, false,
                                "FIFO(4 byte * 1024 record)", ref regName, ref isLink );
:
ushort remain = 0;
result = EGAPI.EgTagGetRemainFIFO ( "MyApp.FIFO1", &remain );
if ( result == EDGE_SUCCESS && remain < 1024 ) {
    // データを登録する余裕がある。データを登録する
    byte[] writeData = new byte[] {1, 2, 3, 4};
    result = EGAPI.EgTagEnqueueFIFO ( "MyApp.FIFO1", ref writeData, ref remain );
}
```


EgTagGetRemainFIFOByIndex

FIFO 型 EgTag に登録されているデータ件数を取得します(インデックス指定版):

シンタックス

```
int EgTagGetRemainFIFOByIndex ( EGOBJINDEX index,
                                ref ushort wRemain );
```

引 数

index [IN] EgTag のインデックス

wRemain [OUT] タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagGetRemainFIFOByIndex コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
edge_API.EGOBJINDEX index; // NOTE: インデックスは一覧取得 API で取得してください
ushort remain = 0;
:
result = EGAPI.EgTagGetRemainFIFOByIndex ( index, ref remain);
if ( result == EDGE_SUCCESS && remain > 0 ) {
    // 取得可能なデータがある。データを取得する
    byte[] readBuf = new byte[4];
    result = EGAPI.EgTagDequeueFIFOByIndex ( index, ref readBuf, ref remain );
    :
}
```

EgTagGetRemainFIFOByEntry

FIFO 型 EgTag に登録されているデータ件数を取得します(Entry 指定版):

シンタックス

```
int EgTagGetRemainFIFOByEntry ( IntPtr    entry,
                                ref ushort wRemain );
```

引 数

entry [IN] EgTag のエントリ

wRemain [OUT] タグに登録されているデータ件数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータが不正

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagGetRemainFIFOByEntry コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
int result;
object entry; // NOTE: エントリは EgTagGetProperty で取得してください
byte[] readBuf = new byte[0];
ushort remain = 0;
:
result = EGAPI.EgTagGetRemainFIFOByEntry ( (IntPtr)entry, ref remain );
while ( result == EDGE_SUCCESS && remain > 0 ) {
    // 取得可能なデータがある。全て読み捨てる
    result = EGAPI.EgTagDequeueFIFOByEntry ( (IntPtr)entry, ref readBuf, ref remain );
}
```

EgTagSetDefaultValue

ECI(XML)で定義した Tag に対して、ECI の Value 値を書き込みます:

シンタックス

```
int EgTagSetDefaultValue ();
```

引 数

なし

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_RESET_DEFAULT_VALUE_TIMEOUT	0x8000041A	フレームワークを利用していない コンテナから実行されている

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagSetDefaultValue コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
result = EGAPI.EgTagSetDefaultValue ();
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgTagSetDefaultValueByGroup

ECI(XML)で定義した Tag に対して、ECI の Value 値を書き込みます。更新対象は ECI の "InitGroup" 設定を行っている Tag です。対象を API 引数で指定します:

シンタックス

```
int EgTagSetDefaultValueByGroup (UInt16 wGroup) ;
```

引 数

wGroup

[IN] 更新対象タグ InitGroup 番号

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_RESET_DEFAULT_VALUE_TIMEOUT	0x8000041A	フレームワークを利用していない コンテナから実行されている

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

コールサンプル

EgTagSetDefaultValueByGroup コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
UInt16_t groupNo = 1;
result = EGAPI.EgTagSetDefaultValueByGroup (groupNo);
if ( result == EDGE_SUCCESS )
    printf("成功");
```

ECI サンプル

```
<?xml version="1.0" encoding="utf-8"?>
<RTedge xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
:
<Tags>
  <Tag Name ="TagTest_0001" InitGroup="1" Type="6" Value="10"/>
  <Tag Name ="TagTest_0002" InitGroup="1" Type="3" Value="172"/>
  <Tag Name ="TagTest_0003" InitGroup="2" Type="3" Value="171"/>
  <Tag Name ="TagTest_0004" Type="3" Value="1"/>
  <Tag Name ="TagTest_0005" Type="1" Value="0"/>
</Tags>
</RTedge>
```

- 上記 ECI サンプル設定において、コードサンプルを実行した場合、更新されるタグは "TagTest_0001", "TagTest_0002" です。

EgCollectorGetCount

EgCollector の登録数を取得します:

シンタックス

```
int EgCollectorGetCount ( ref uint itemNum );
```

引 数

itemNum [OUT](ref) EgCollector の登録数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgCollectorGetCount コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
uint    itemNum = 0;
result = EGAPI.EgCollectorGetCount (ref itemNum);
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgCollectorGetEnum

EgCollector の一覧を取得します:

シンタックス

```
int EgCollectorGetEnum ( ref EGOBJINDEX[] indexArray,  
                        ref uint          itemNum);
```

引 数

indexArray [OUT](ref) 取得したインデックス一覧

itemNum [OUT](ref) インデックス一覧の取得数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgCollectorGetEnum コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成  
:  
uint          itemNum = 0;  
EGAPI.EgCollectorGetCount (ref itemNum); // 登録数取得  
edge_API.EGOBJINDEX[] idxArray = new edge_API.EGOBJINDEX[itemNum];  
uint          getNum = 0;  
result = EGAPI.EgCollectorGetEnum (ref idxArray, ref getNum); // 一覧取得  
if ( result == EDGE_SUCCESS )  
    printf("成功");
```

EgCollectorGetProperty

EgCollector の情報を取得します:

シンタックス

```
int EgCollectorGetProperty ( string      stName,
                             string      stPropName,
                             ref object  value );
```

引 数

- stName** [IN] EgCollector 名
- stPropName** [IN] Property 名
- value** [OUT] Property 値

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定した名前は登録されていない
EDGE_PARAM_INVALID	0x80000004	指定した属性名は不正

上記以外は 3.4. エラーコードリストを参照してください。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された Property 名に対応する値を返す。指定可能な Property 名は以下の文字列を指定。

Property 名	型	内容
"Interval"	uint32_t	収集周期(ms)
"Priority"	uint8_t	スレッドプライオリティ(INtime サービス用)
"ColSize"	uint32_t	データ格納メールボックス 1 レコードサイズ
"RowSize"	uint32_t	データ格納メールボックスレコード数
"DatasetName"	char[]	データセット名

EgCollectorGetPropertyByIndex

EgCollector の情報を取得します(インデックス指定版):

シンタックス

```
int EgCollectorGetPropertyByIndex (EGOBJINDEX index,
string stPropName,
ref object value );
```

引 数

index [IN] EgCollector のインデックス
stPropName [IN] Property 名
value [OUT] Property 値

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定したインデックスは登録されていない
EDGE_PARAM_INVALID	0x80000004	指定した属性名は不正

上記以外は 3.4. エラーコードリストを参照してください。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された Property 名に対応する値を返す。指定可能な Property 名は以下の文字列を指定。

Property 名	型	内容
"Interval"	uint32_t	収集周期(ms)
"Priority"	uint8_t	スレッドプライオリティ(INtime サービス用)
"ColSize"	uint32_t	データ格納メールボックス 1 レコードサイズ
"RowSize"	uint32_t	データ格納メールボックスレコード数
"DatasetName"	char[]	データセット名

EgDatasetCreate

EgDataset を生成します:

シンタックス

```
int EgDatasetCreate( string stName );
```

引 数

stName [IN] Dataset 名

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_ISEXIST_TAGNAME	0x80000409	同名登録済み
EDGE_TTABLE_NOEMPTY	0x8000040E	テーブルに空きがない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgDatasetAddTag

EgDataset に EgTag 情報を追加します:

シンタックス

```
int EgDatasetAddTag ( string      stName,  
                     string      stTagName );
```

引 数

stName [IN] Dataset 名

stTagName [IN] Tag 名

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_DSETNAME	0x80000501	指定された Dataset 名はテーブルに存在しない
EDGE_TTABLE_VALUE_OVERFLOW	0x8000040F	書き込みデータサイズが格納サイズより大きい
EDGE_TTABLE_NOEXISTT_TAGNAME	0x8000040A	指定したタグ名がタグテーブルに存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgDatasetGetSize

C#では使用できません:

EgDatasetGetBinary

C#では使用できません:

EgDatasetGetFirst

EgDataset の最初のタグを取得します:

シンタックス

```
int EgDatasetGetFirst ( string      stName,
                       TDSLINK     stDslink);
```

引 数

<i>stName</i>	[IN] Dataset 名
<i>stDslink</i>	[OUT] Dataset サーチ用構造体

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOEXISTT_DSETNAME	0x80000501	指定された Dataset 名はテーブルに存在しない
EDGE_TTABLE_NOEXISTT_DSETITEM	0x80000502	指定された Dataset 名には Tag が登録されていない

TDSLINK 構造体

```
public struct TDSLINK
{
    public string      pTagName;    // タグ名
    public IntPtr      pTag;        // Tag 情報へのポインタ
    public IntPtr      pds;        // Dataset 情報へのポインタ
    public ushort      inst;       // FindFirst/Next で使用
    public ushort      offset;     // T データセットに登録されている Tag のデータサイズ位置
}
```

TDSLINK 構造体 フィールド

型	シンボル	内容
string	pTagName	取得したタグの名称
IntPtr	pTag	(内部で使用の為使用不可)
IntPtr	Pds	(内部で使用の為使用不可)
ushort	inst	(内部で使用の為使用不可)
Ushort	Offset	データセットに登録されている Tag のデータサイズ位置

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgDatasetGetNext

EgDataset の次のタグを取得します:

シンタックス

```
int EgDatasetGetNext ( TDSLINK    stDslink );
```

引 数

stDsLink [OUT] Dataset サーチ用構造体

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_TAG_IS_NULL	0x80000503	next の Tag は存在しない

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgDatasetGetCount

EgDataset の登録数を取得します:

シンタックス

```
int EgDatasetGetCount ( ref uint itemNum);
```

引 数

itemNum [OUT](ref) EgDataset の登録数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgDatasetGetCount コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
uint    itemNum = 0;
result = EGAPI.EgDatasetGetCount (ref itemNum);
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgDatasetGetEnum

EgDataset の一覧を取得します:

シンタックス

```
int EgDatasetGetEnum ( ref EGOBJINDEX[] indexArray,  
                      ref uint      itemNum);
```

引 数

indexArray [OUT](ref) 取得したインデックス一覧

itemNum [OUT](ref) 取得数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgDatasetGetEnum コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成  
:  
uint      itemNum = 0;  
EGAPI.EgDatasetGetCount (ref itemNum); // 登録数取得  
edge_API.EGOBJINDEX[] idxArray = new edge_API.EGOBJINDEX[itemNum];  
uint      getNum = 0;  
result = EGAPI.EgDatasetGetEnum (ref idxArray, ref getNum); // 一覧取得  
if ( result == EDGE_SUCCESS )  
    printf("成功");
```

EgDatasetGetPropertyByIndex

EgDataset の情報を取得します(インデックス指定版):

シンタックス

```
int EgDatasetGetPropertyByIndex ( EGOBJINDEX index,
                                  string stPropName,
                                  ref object value );
```

引 数

- index* [IN] EgDataset のインデックス
- stPropName* [IN] Property 名
- value* [OUT] Property 値

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定したインデックスは登録されていない
EDGE_PARAM_INVALID	0x80000004	指定した属性名は不正

上記以外は 3.4. エラーコードリストを参照してください。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された Property 名に対応する値を返す。指定可能な Property 名は以下の文字列を指定。

Property 名	型	内容
"Name"	string	データセット名
"ItemNum"	UInt16_t	登録タグ数

EgServiceGetCount

EgService の登録数を取得します:

シンタックス

```
int EgServiceGetCount ( ref uint itemNum);
```

引 数

itemNum [OUT](ref) EgService の登録数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgServiceGetCount コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
uint    itemNum = 0;
result = EGAPI.EgServiceGetCount (ref itemNum);
if ( result == EDGE_SUCCESS )
    printf("成功");
```


EgServiceGetEnum

EgService の一覧を取得します:

シンタックス

```
int EgServiceGetEnum ( ref EGOBJINDEX[] indexArray,
                      ref uint      itemNum);
```

引 数

indexArray [OUT](ref) 取得したインデックス一覧

itemNum [OUT](ref) 取得数

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時

上記以外は [3.4. エラーコードリスト](#) を参照してください。

コールサンプル

EgServiceGetEnum コール例

```
edge_API EGAPI = new edge_API();    //インスタンス作成
:
uint      itemNum = 0;
EGAPI.EgServiceGetCount (ref itemNum); // 登録数取得
edge_API.EGOBJINDEX[] idxArray = new edge_API.EGOBJINDEX[itemNum];
uint      getNum = 0;
result = EGAPI.EgServiceGetEnum (ref idxArray, ref getNum); // 一覧取得
if ( result == EDGE_SUCCESS )
    printf("成功");
```

EgServiceGetPropertyByIndex

EgService の情報を取得します(インデックス指定版):

シンタックス

```
int EgServiceGetPropertyByIndex ( EGOBJINDEX    index,
                                string          stPropName,
                                ref object      value );
```

引 数

index [IN] EgService のインデックス

stPropName [IN] Property 名

value [OUT] Property 値

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	指定したインデックスは登録されていない
EDGE_PARAM_INVALID	0x80000004	指定した属性名は不正

上記以外は 3.4. エラーコードリストを参照してください。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。
- 指定された Property 名に対応する値を返す。指定可能な Property 名は以下の文字列を指定。

Property 名	型	内容
"Name"	string	EgService 名
"Path"	string	実行モジュールファイルパス
"Arg"	string	起動引数
"NodeName"	string	ノード名

EgFW_AddTagTrigger

サービスコンテナにタグトリガーを登録します:

シンタックス

```
int EgFW_AddTagTrigger ( string      tagName );
```

引 数

**tagName* [IN] タグ名

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_FW_XML_FORMAT_ERROR	0x80001003	引数が NULL、または空文字、または文字数オーバー
EDGE_FW_OBJECT_SIZE_OVER	0x80001008	タグトリガー登録上限
EDGE_FW_DUPLICATE_OBJECT	0x80001007	タグトリガーとして既に登録済み
EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	タグが存在しない

上記以外は 3.4. エラーコードリストを参照してください。

注釈

- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgGatherExecute

リストに基づいたログなどのファイルを指定のフォルダへコピー後、圧縮(ZIP)します:

シンタックス

```
int EgGatherExecute ( string outputDirPath,
                     string listPath );
```

引 数

outputDirPath [IN] 出力フォルダパス

listPath [IN] 収集対象のファイル/フォルダパスが記載された収集ファイルリストのファイルパス

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータ不正
EDGE_FAILED_TO_START_PROCESS	0x80000601	プロセスを開始できませんでした。ログ収集コンテナがインストールされているか、指定したファイルのパスが存在するか確認してください。
	0xFFFFFFFF	

注釈

- 本 API を使用する場合は、ログ収集コンテナをインストールする必要があります。また本 API は C# でのみ使用可能です。
- 収集ファイルリストのフォーマットや出力するファイル名、その他詳細な仕様に関しては、ログ収集コンテナのマニュアルを参照ください。
- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgMDiagExecute_PfmCntAdd

Windows パフォーマンスカウンターの値を取得するためのキーを登録します:

シンタックス

```
int EgMDiagExecute_PfmCntAdd ( string      counterName_All );
```

引 数

counterName_All [IN] カウンター名(全体)
(例: " ¥Process(_Total)¥Working Set" 等)

"¥カテゴリ名(インスタンス名)¥ カウンター名"

カテゴリ名 : (例: "Processor" や "Process" 等)
インスタンス名: (例: "% Processor Utility" や "Working Set" 等)
カウンター名 : (例: "_Total" 等)

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータ不正
-	0xFFFFFFFF	指定したパラメータが正しいか確認してください。

注釈

- 本 API を使用する場合は、診断・通知コンテナをインストールする必要があります。また本 API は C# でのみ使用可能です。
- 指定可能なパラメータは、Windows パフォーマンスモニターで使用可能なカウンターの各情報(カテゴリ名・カウンター名・インスタンス名)がベースとなっております。
- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgMDiagExecute_PfmCntGet

Windows パフォーマンスカウンターの値を取得します。パフォーマンスモニターで見えるような情報を取得できます:

シンタックス

```
int EgMDiagExecute_PfmCntGet (string    counterName_All,
                              ref float refValue);
```

引 数

counterName_All [IN] カウンター名(全体)
(例: " ¥Process(_Total)¥Working Set" 等)

"¥カテゴリ名(インスタンス名)¥ カウンター名"

カテゴリ名 : (例: "Processor" や "Process" 等)
 インスタンス名: (例: "% Processor Utility" や "Working Set" 等)
 カウンター名 : (例: "_Total" 等)

refValue [OUT](ref) 格納したデータ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータ不正
-	0xFFFFFFFF	指定したパラメータが正しいか確認してください。

注釈

- 本 API を使用する場合は、診断・通知コンテナをインストールする必要があります。また本 API は C# でのみ使用可能です。
- 指定可能なパラメータは、Windows パフォーマンスモニターで使用可能なカウンターの各情報(カテゴリ名・カウンター名・インスタンス名)がベースとなっております。
- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgMDiagExecute_DiskInfo

S.M.A.R.T.情報を取得します:

シンタックス

```
int EgMDiagExecute_DiskInfo(ref DiskExecInfo diskExecInfo);
```

引 数

diskExecInfo [OUT](ref) 格納した S.M.A.R.T.情報全般を含む構造体
【S.M.A.R.T.情報全般の構造体】参照

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_FAILED_TO_START_PROCESS	0x80000601	プロセスを開始できませんでした。診断・通知コンテナがインストールされているか確認してください。
-	0xFFFFFFFF	

注釈

- 本 API を使用する場合は、診断・通知コンテナをインストールする必要があります。また本 API は C# でのみ使用可能です。
- 本 API はソフトウェア「CrystalDiskInfo」を利用しております。
- その他詳細な仕様に関しては、診断・通知コンテナのマニュアルを参照ください。
- 上記以外のステータス値は [「3.4. エラーコードリスト」](#) を参照してください。

EgMDiagGetSMARTDiskName

「EgMDiagExecute_DiskInfo」API で取得した S.M.A.R.T 情報から指定のディスクのモデル名を取得します:

シンタックス

```
bool EgMDiagGetSMARTDiskName ( DiskExecInfo diskExecInfo,  
                                int diskno,  
                                ref string value);
```

引 数

diskExecInfo	[IN] 「EgMDiagExecute_DiskInfo」API で取得した S.M.A.R.T 情報
diskno	[IN] 名前を取得したいディスク番号(見つかった順の数値(1～の数値))
value	[OUT](ref) 格納したディスクのモデル名

戻り値

True	モデル名の取得成功
False	モデル名の取得失敗

注釈

- 本 API を使用する場合は、診断・通知コンテナをインストールする必要があります。また本 API は C# でのみ使用可能です。
- 本 API はソフトウェア「CrystalDiskInfo」を利用しております。
- その他詳細な仕様に関しては、診断・通知コンテナのマニュアルを参照ください。
- ディスク番号は見つかった順の番号になります。一度インストール済みのソフトウェア「CrystalDiskInfo」を使用して表示される順番を確認してください。

EgMDiagGetSMARTValue

「EgMDiagExecute_DiskInfo」API で取得した S.M.A.R.T 情報から指定のディスク・検査項目 ID に対応する現在値、ステータス値を取得します:

シンタックス

```
bool EgMDiagGetSMARTValue ( DiskExecInfo  diskExecInfo,
                             int            diskNo,
                             int            id,
                             ref byte       value,
                             ref byte       status);
```

引 数

diskExecInfo	[IN] 「EgMDiagExecute_DiskInfo」API で取得した S.M.A.R.T 情報
diskNo	[IN] 現在値を取得したいディスク番号(見つかった順の数値(1~の数値))
id	[IN] 現在値を取得したい検査項目 ID
value	[OUT](ref) 格納した現在値
status	[OUT](ref) 格納したステータス値

戻り値

True	値・ステータスの取得成功
False	値・ステータスの取得失敗

注釈

- 本 API を使用する場合は、診断・通知コンテナをインストールする必要があります。また本 API は C# でのみ使用可能です。
- 本 API はソフトウェア「CrystalDiskInfo」を利用しております。
- ディスク番号は見つかった順の番号になります。一度インストール済みのソフトウェア「CrystalDiskInfo」を使用して表示される順番を確認してください。
- 検査項目 ID の詳細、その他詳細な仕様に関しては、診断・通知コンテナのマニュアルを参照ください。

EgDecode

バイナリデータ(バイト配列)から指定のフォーマットに基づいたバイナリデータを取得します:

シンタックス

```
int EgDecode (  ref byte[]      byDst,
                byte[]         bySrc,
                string          addrformat,
                ref uint        actualsize);
```

引 数

byDst	[OUT] 取り出したバイト配列
bySrc	[IN] 取り出し元バイト配列
addrformat	[IN] フォーマット
Actualsize	[OUT](ref) 格納したデータの有効サイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータ不正
EGDE_COMMON_DECODE_FORMAT_CMPIDENT_ERR	0x80002001	アドレス書式不正
EGDE_COMMON_DECODE_FORMAT_PRIOD1_ERR	0x80002002	Decode 時のアドレス書式 (format)が不正(ピリオド指定があるべきところがない)
EGDE_COMMON_DECODE_FORMAT_PRIOD2_ERR	0x80002003	Decode 時のアドレス書式 (format)が不正(ピリオド後の指定がない)

注釈

- 引数で当たられたバイナリ(バイト配列)・フォーマット文字列が空の場合は null を返します。
- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- アドレス書式は「[表 5 アドレス表記](#)」を参照してください。

EgEncode

Address format に則ってバイナリデータ列の所定位置にデータを書き出します:

シンタックス

```
int EgEncode (  ref byte[]    byDst,
                byte[]      bySrc,
                string      addrformat,
                ref uint     actualsize);
```

引 数

- byDst*

[OUT] 書き込み先配列
- bySrc*

[IN] 書き込みたい変数
- addrformat*

[IN] フォーマット(EgDecode の説明参照)
- actualsize*

[OUT] 書き込まれたバイトサイズ

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータ不正
EGDE_COMMON_DECODE_FORMAT_CMPIDENT_ERR	0x80002001	アドレス書式不正
EGDE_COMMON_DECODE_FORMAT_PRIOD1_ERR	0x80002002	Encode 時のアドレス書式 (format)が不正(ピリオド指定があるべきところがない)
EGDE_COMMON_DECODE_FORMAT_PRIOD2_ERR	0x80002003	Encode 時のアドレス書式 (format)が不正(ピリオド後の指定がない)

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- アドレス書式は「[表 5 アドレス表記](#)」を参照してください。

EgMakeAddress

オフセットとサイズの情報から address 書式を作成します:

シンタックス

```
string EgMakeAddress ( int      offset,  
                       short    size );
```

引 数

<i>offset</i>	[IN] オフセットバイト値
<i>size</i>	[IN] データバイトサイズ値

戻り値

アドレス文字列（生成できない場合は空）

EgParseAddress

address 書式からオフセットとサイズに分解します:

シンタックス

```
int32_t EgParseAddress ( string      addrformat,
                        ref int      offset,
                        ref uint     size,
                        ref short    bit,
                        ref byte     typemode);
```

引 数

- addressformat** [IN] 元になるアドレス書式文字列
- offset** [OUT] (ref) 取り出せたオフセットバイト値
- size** [OUT] (ref) 取り出せたバイトサイズ値
- bit** [OUT] (ref) 取り出せたビット番号値
- typemode** [OUT] (ref) 取り出せたデータ型区分番号

戻り値

ステータス値

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	正常終了時
EDGE_PARAM_INVALID	0x80000004	パラメータ不正
EGDE_COMMON_DECODE_FORMAT_CMPIDENT_ERR	0x80002001	アドレス書式不正
EGDE_COMMON_DECODE_FORMAT_PRIOD1_ERR	0x80002002	アドレス書式(format)が不正(ピリオド指定があるべきところがない)
EGDE_COMMON_DECODE_FORMAT_PRIOD2_ERR	0x80002003	アドレス書式(format)が不正(ピリオド後の指定がない)
EGDE_COMMON_DECODE_FORMAT_PRIOD3_ERR	0x80002004	アドレス書式(format)が不正(ピリオド値下限オーバー)
EGDE_COMMON_DECODE_FORMAT_PRIOD4_ERR	0x80002005	アドレス書式(format)が不正(ピリオド値上限オーバー)

注釈

- 上記以外のステータス値は「[3.4. エラーコードリスト](#)」を参照してください。
- アドレス書式は「[表 5 アドレス表記](#)」を参照してください。

3.4. エラーコードリスト

API 関数コールに失敗した場合、RT-edge API は以下に示すエラーコードを返却します:

定義名	エラーコード	説明
EDGE_SUCCESS	0x00000000	関数は正常に終了しました
EDGE_LOCATION_ERROR	0x80000001	システムエラー(ロケーションハンドルが取得できません)
EDGE_KERNEL_NOT_RUNNING	0x80000002	システムエラー(カーネルが起動していません)
EDGE_ROOTPROCESS_ERROR	0x80000003	システムエラー(ルートプロセスが取得できません)
EDGE_PARAM_INVALID	0x80000004	パラメータが不正です
EDGE_THREAD_ERROR	0x80000005	システムエラー(スレッドが開始できません)
EDGE_RTCD_NAME_INVALID	0x80000100	名称が不正です
EDGE_RTCD_NAME_LENGTH_ERROR	0x80000101	名称の文字数が制限値をオーバーしています
EDGE_RTCD_RECORD_LENGTH_ERROR	0x80000102	レコード長が制限値をオーバーしています
EDGE_RTCD_RECORD_NUM_ERROR	0x80000103	レコード数が制限値をオーバーしています
EDGE_RTCD_WRITE_MODE_ERROR	0x80000105	メールボックス/共有メモリの書き込みモードが不正です。
AEDGE_RTCD_EXIST_ERROR	0x80000106	同名のメールボックス/共有メモリが存在しています
EDGE_RTCD_NOT_EXIST_ERROR	0x80000107	システムエラー(共有メモリが存在しません)
EDGE_RTCD_NAME_EXIST_ERROR	0x80000108	既に同じ名前が使われています
EDGE_RTCD_ALLOCATE_ERROR	0x80000109	システムエラー(メモリの確保に失敗しました)
EDGE_RTCD_FREE_ERROR	0x8000010A	システムエラー(メモリの解放に失敗しました)
EDGE_RTCD_CREATE_MEM_HANDLE_ERROR	0x8000010B	システムエラー(メモリハンドルの生成に失敗しました)
EDGE_RTCD_DELETE_MEM_HANDLE_ERROR	0x8000010C	システムエラー(メモリハンドルの削除に失敗しました)
EDGE_RTCD_CATALOG_ERROR	0x8000010D	システムエラー(カタログ登録に失敗しました)
EDGE_RTCD_UNCATALOG_ERROR	0x8000010E	システムエラー(カタログ削除に失敗しました)
EDGE_RTCD_LOOKUP_ERROR	0x8000010F	システムエラー(ルックアップに失敗しました)
EDGE_RTCD_MAPPING_ERROR	0x80000110	システムエラー(マッピング処理が失敗しました)
EDGE_RTCD_UNMAPPING_ERROR	0x80000111	システムエラー(アンマッピングが失敗しました)
EDGE_RTCD_ACCESS_MODE_ERROR	0x80000112	アクセスモードエラー(アクセスに失敗しました)
EDGE_RTCD_ALREADY_OPEN_READ_ERROR	0x80000113	既に読み込みアクセスでオープンされています
EDGE_RTCD_ALREADY_OPEN_WRITE_ERROR	0x80000114	既に書き込みアクセスでオープンされています(シングルモード時)
EDGE_RTCD_MEMINFO_INVALID	0x80000115	システムエラー(メモリ情報が不正です)
EDGE_RTCD_HANDLE_INVALID	0x80000116	システムエラー(ハンドルが不正です)
EDGE_RTCD_BUFFER_INVALID	0x80000117	バッファが不正です。最大値・制限値を確認してください
EDGE_RTCD_BUFFER_SIZE_INVALID	0x80000118	バッファサイズが不正です。最大値・制限値を確認してください
EDGE_RTCD_WRITE_SIZE_INVALID	0x80000119	書き込みサイズが不正です。
EDGE_RTCD_RECORD_HEADER_INVALID	0x8000011A	システムエラー(RTCD ヘッダが不正です)
EDGE_RTCD_INFO_INVALID	0x8000011B	システムエラー(RTCD 情報が不正です)
EDGE_RTCD_NO_UPDATE	0x8000011C	システムエラー(データ更新がありません)
EDGE_RTCD_BUFFER_FULL_ERROR	0x8000011D	バッファフルエラー
EDGE_RTCD_CREATE_LOGMBOX_ERROR	0x8000011E	システムエラー(ログ用 MAILBOX 生成失敗)
EDGE_RTCD_PROPERTY_NOT_SUPPORT	0x8000011F	システムエラー(タグ生成エラー(指定したプロパティはサポートしていません))

EDGE_RTCD_CREATE_SEMAPHORE1_ERROR	0x8000200	システムエラー(セマフォ生成エラー(排他用セマフォ))
EDGE_RTCD_CREATE_SEMAPHORE2_ERROR	0x8000201	システムエラー(セマフォ生成エラー(書込通知用セマフォ))
EDGE_RTCD_DELETE_SEMAPHORE1_ERROR	0x8000202	システムエラー(セマフォ削除エラー(排他用セマフォ))
EDGE_RTCD_DELETE_SEMAPHORE2_ERROR	0x8000203	システムエラー(セマフォ削除エラー(書込通知用セマフォ))
EDGE_RTCD_CATALOG_SEMAPHORE1_ERROR	0x8000204	システムエラー(セマフォカタログエラー(排他用セマフォ))
EDGE_RTCD_CATALOG_SEMAPHORE2_ERROR	0x8000205	システムエラー(セマフォカタログエラー(書込通知用セマフォ))
EDGE_RTCD_SEMAPHORE_TIMEOUT	0x8000206	システムエラー(セマフォ取得タイムアウト)
EDGE_RTCD_WAIT_SEMAPHORE_ERROR	0x8000207	システムエラー(セマフォ取得エラー)
EDGE_RTCD_RELEASE_SEMAPHORE_ERROR	0x8000208	システムエラー(セマフォ解放エラー)
EDGE_RTCD_OPEN_SEMAPHORE1_ERROR	0x8000209	システムエラー(セマフォオープンエラー(排他用セマフォ))
EDGE_RTCD_OPEN_SEMAPHORE2_ERROR	0x800020A	システムエラー(セマフォオープンエラー(書込通知用セマフォ))
EDGE_RTCD_NOT_EXIST_SEMAPHORE_ERROR	0x800020B	システムエラー(セマフォが存在しません)
EDGE_RTCD_LOOKUP_SEMAPHORE_ERROR	0x800020C	システムエラー(セマフォルックアップエラー)
EDGE_RTCD_SEMAPHORE1_EXIST_ERROR	0x800020D	システムエラー(既に同じセマフォが存在します(排他用セマフォ))
EDGE_RTCD_SEMAPHORE1_NAME_EXIST_ERROR	0x800020E	システムエラー(既に同じ名前が使われています(排他用セマフォ))
EDGE_RTCD_SEMAPHORE2_EXIST_ERROR	0x800020F	システムエラー(既に同じセマフォが存在します(書込通知用セマフォ))
EDGE_RTCD_SEMAPHORE2_NAME_EXIST_ERROR	0x8000210	システムエラー(既に同じ名前が使われています(書込通知用セマフォ))
EDGE_RTCD_ARG_DATA_SIZE_ERROR	0x8000300	指定した引数の変数サイズが既定のサイズより大きい,または小さいためバッファが壊れます
EDGE_TTABLE_MEMORRY_ERROR	0x8000400	メモリの作成ができません(カーネルが起動しているか確認してください)
EDGE_TTABLE_EXIST_ERROR	0x8000401	システムエラー(既に同じテーブル名が存在します)
EDGE_TTABLE_CREATE_FILEMAPPING_ERROR	0x8000402	システムエラー(ファイルマッピング生成エラー)
EDGE_TTABLE_MAP_VIEW_ERROR	0x8000403	システムエラー(ビューマップエラー)
EDGE_TTABLE_MEMINFO_INVALID	0x8000404	システムエラー(テーブルメモリ情報が不正です)
EDGE_TTABLE_UNMAP_VIEW_ERROR	0x8000405	システムエラー(ビューアンマップエラー)
EDGE_TTABLE_CREATE_MEM_HANDLE_ERROR	0x8000406	システムエラー(メモリハンドル生成失敗)
EDGE_TTABLE_DELETE_MEM_HANDLE_ERROR	0x8000407	システムエラー(メモリハンドル削除失敗)
EDGE_TTABLE_OPEN_FILEMAPPING_ERROR	0x8000408	システムエラー(ファイルマッピングオープンエラー)
EDGE_TTABLE_ISEXIST_TAGNAME	0x8000409	既に同じ名前のオブジェクトが登録済みです
EDGE_TTABLE_NOEXISTT_TAGNAME	0x800040A	指定した名前がテーブルに存在しません
EDGE_TTABLE_HASH_ERROR	0x800040B	システムエラー(ハッシュ値が不正です)
EDGE_TTABLE_INDEX_ERROR	0x800040C	システムエラー(Index 値が不正です)
EDGE_TTABLE_DATA_ERROR	0x800040D	システムエラー(データ値が存在しません)
EDGE_TTABLE_NOEMPTY	0x800040E	テーブルに空きがありません
EDGE_TTABLE_VALUE_OVERFLOW	0x800040F	書き込み/読込データサイズが不正です
EDGE_TTABLE_NAME_EXIST_ERROR	0x8000410	既に同じ名前のオブジェクトが登録済みです
EDGE_TTABLE_CATALOG_ERROR	0x8000411	システムエラー(カタログ登録に失敗しました)
EDGE_TTABLE_LOOKUP_ERROR	0x8000412	システムエラー(ルックアップに失敗しました)
EDGE_TTABLE_MAPPING_ERROR	0x8000413	システムエラー(マッピング処理が失敗しました)

EDGE_TTABLE_NOT_EXIST_ERROR	0x80000414	システムエラー(対象のテーブルが存在しません)
EDGE_TTABLE_FREE_ERROR	0x80000415	システムエラー(メモリの解放に失敗しました)
EDGE_TTABLE_TYPE_ERROR	0x80000416	EgTag の型タイプが範囲外です
EDGE_TTABLE_DATASIZE_ERROR	0x80000417	EgTag のデータサイズが範囲外です
EDGE_TTABLE_DUPLICATE_ERROR	0x80000418	EgTag の重複登録に失敗しました
EDGE_TTABLE_NOEXISTTT_DSETNAME	0x80000501	指定された EgDataset 名は登録されていません
EDGE_TTABLE_NOEXISTTT_DSETITEM	0x80000502	指定された EgDataset 名には EgTag が登録されていません
EDGE_TTABLE_TAG_IS_NULL	0x80000503	EgDataset 内の次の EgTag が見つかりませんでした
EDGE_TTABLE_SERVICE_INDEX_NULL	0x80000505	Service 名の Index 化がされていない
EDGE_TTABLE_SYSTEM_CONFIG_NULL	0x80000506	オブジェクト数制限値共有メモリが作成できない
EDGE_FW_INIT_ERROR	0x80001001	フレームワーク処理の初期化に失敗しました
EDGE_FW_NAME_INVALID	0x80001002	サービス名称 (プロセス名)が不正です
EDGE_FW_XML_FORMAT_ERROR	0x80001003	サービスコンフィグファイルのフォーマットが異常です
EDGE_FW_COLLECTOR_ALREADY_WORKING	0x80001004	コレクタは既に動作しています。
EDGE_FW_OBJECT_NOTEXIST	0x80001005	指定したオブジェクトは存在しません
EDGE_FW_MESSAGE_NOTEXIST	0x80001006	指定したツール間通信メッセージは存在しません
EDGE_FW_DUPLICATE_OBJECT	0x80001007	同名のオブジェクトが存在します
EDGE_FW_OBJECT_SIZE_OVER	0x80001008	オブジェクト数が制限値を超えます/超えています
EGDE_COMMON_DECODE_FORMAT_CMPIDENT_ERR	0x80002001	アドレス書式(format)が不正です(比較エントリの識別子)
EGDE_COMMON_DECODE_FORMAT_PRIOD1_ERR	0x80002002	アドレス書式(format)が不正です(ピリオド指定があるべきところがない)
EGDE_COMMON_DECODE_FORMAT_PRIOD2_ERR	0x80002003	アドレス書式(format)が不正です(ピリオド後の指定がない)
EGDE_COMMON_DECODE_FORMAT_PRIOD3_ERR	0x80002004	アドレス書式(format)が不正です(ピリオド値下限オーバー)
EGDE_COMMON_DECODE_FORMAT_PRIOD4_ERR	0x80002005	アドレス書式(format)が不正です(ピリオド値上限オーバー)
EGDE_COMMON_DECODE_FORMAT_DST_BUFFERR	0x80002006	アドレス書式(format)が不正です(dst バッファ超過)
EGDE_COMMON_DECODE_FORMAT_OFFSET1_ERR	0x80002007	アドレス書式(format)が不正です(不適切なオフセット位置)
EGDE_COMMON_DECODE_FORMAT_OFFSET2_ERR	0x80002008	アドレス書式(format)が不正です(src バッファを offset 指定が超過)
EDGE_TAGPTR_CREATE_ERROR	0x80004001	タグポインタ(文字列型/バイト配列型 EgTag)用ヒープ領域の生成ができない
EDGE_TAGPTR_MAP_ERROR	0x80004002	タグポインタ用ヒープ領域のマッピングができない
EDGE_TAGPTR_EXIST_ERROR	0x80004003	既にタグポインタ用ヒープ領域が存在します
EDGE_TAGPTR_COUNT_OVER	0x80004004	タグポインタ登録数が最大数のため、タグポインタが登録できない
EDGE_TAGPTR_NO_FREE_SPACE	0x80004005	タグポインタ用ヒープ領域の空き容量がないため、タグポインタが登録できない
EDGE_TAGPTR_SIZE_ZERO	0x80004006	サイズに 0 が指定された
EDGE_TAGPTR_SIZE_UNMATCH	0x80004007	リンクタグのサイズが不一致です
EDGE_TAGPTR_SIZE_OVERFLOW	0x80004008	書き込み/読み込みデータサイズが不正です
EDGE_TAGPTR_TYPE_UNMATCH	0x80004009	タグタイプが不一致です
EDGE_TAGPTR_OFFSET_OVERFLOW	0x8000400A	オフセットが範囲外です
EDGE_TAGPTR_BUFFER_NULL	0x8000400B	バッファに NULL が指定されました
EDGE_TAGPTR_LOCK_TIMEOUT	0x8000400C	指定時間内に書き込み/読み込みができませんでした
EDGE_TAGTRIG_NOT_EXIST_EGTAGTRIGMGR	0x80005001	タグトリガー管理用テーブルが存在しません
EDGE_TAGTRIG_FW_NOT_USE	0x80005002	フレームワークを利用していません

EDGE_TAGTRIG_FRAMEWORK_MAX	0x80005003	フレームワークに登録可能なタグトリガー数が最大です
EDGE_TAGTRIG_TAG_MAX	0x80005004	タグに登録可能なトリガー対象サービス数が最大です
EDGE_TAGTRIG_SERVICE_MAX	0x80005005	サービスに登録可能なトリガー数が最大です
EDGE_TAGTRIG_ALREADY_ENTRY	0x80005006	既に登録されているタグ名です
EDGE_TAGTRIG_NO_ENTRY	0x80005007	トリガーフラグを設定する対象のタグがありません
EDGE_FIFO_UNDERRUN	0x80006001	FIFO 型タグに登録されたデータがない
EDGE_FIFO_OVERRUN	0x80006002	FIFO 型タグに登録できるデータ件数を超過した
EDGE_FIFO_TAG_ERROR	0x80006003	FIFO 型タグの内部整合性エラー
-	0xffffffff	必要なアプリケーション・ライブラリが正しくインストール されていません。インストール構成を見直してください。

4. 付録

4.1. メッセージ通信用システムメッセージ一覧

表 4 システムメッセージ一覧

定義名	Mess No	引数	説明
EM_RETURN_RECEIVE_OK	1	-	返答用メッセージ
EgService 制御関連			
EM_SERVICE_STOP	101	-	受け取ったサービスはサービスを終了させる
EM_SERVICE_START	102	-	受け取ったサービスはデータ更新を開始する
EM_SERVICE_PAUSE	103	-	受け取ったサービスはデータ更新を一時停止する
EM_SERVICE_UPDATE	104	-	受け取ったサービスはデータをリフレッシュする
EgService が参照する EgTag・EgCollector・EgTagTrigger 関連			
EM_SERVICE_ADD_TAG	111	TagRefName	EgService が参照する EgTag リストに参照用 EgTag 情報を追加する(フレームワーク内)
EM_SERVICE_ADD_OUT_TAG	114	TagRefName	EgService が参照する出力用 EgTag リストに参照用 EgTag 情報を追加する(フレームワーク内)
EM_SERVICE_ADD_IN_TAG	117	TagRefName	EgService が参照する入力用 EgTag リストに参照用 EgTag 情報を追加する(フレームワーク内)
EM_SERVICE_ADD_CLCTREF	121	CollectorRefName	EgService が参照する EgCollector (FIFO) リストに参照用 EgCollector 情報を追加する(フレームワーク内)
EM_SERVICE_ADD_MBOX	124	MBoxName, ColSize, RowSize	EgService が参照する EgMailBox を作成する
EM_SERVICE_ADD_TAGTRIGGER	126	TagName	EgService が参照する EgTagTrigger リストに EgTag 情報を追加する(フレームワーク内)
EM_TAGTRIGGER_NOTIFICATION	250	構造体 EG TAGTRIGGER INFO	EgService が参照する EgTagTrigger リスト内の EgTag の値が変化した際に EgService に対して EgTagTrigger 情報を通知する
EgCollector 関連			
EM_COLLECTOR_CREATE	201	Name,	EgCollector を作成する(フレームワーク内) メールボックスを作成するための引数必須
EM_COLLECTOR_START_ALL	202	-	EgCollector すべてをスタートさせる(フレームワーク内)
EM_COLLECTOR_STOP_ALL	203	-	EgCollector すべてをストップさせる(フレームワーク内)
EM_COLLECTOR_START_ONE	204	CollectorName	指定した EgCollector をスタートさせる
EM_COLLECTOR_STOP_ONE	205	CollectorName	指定した EgCollector をストップさせる
EM_COLLECTOR_WRITED_DATA	206	Data	(サービス間通信では使用されない) Collector がデータを追加する際のメッセージ番号
その他			
EM_READ_REQUEST_SERVICE_XML	1000	PATH	EgService が保持するサービスコンフィグファイルを EgBoot サービスに読み込みしてもらうためのメッセージ
EM_READ_REQUEST_SERVICE_XML_END	1001		サービスごとフレームワークからのサービスコンフィグファイル読み込み命令受信時、EgBoot がサービスコンフィグファイルを読み終わったときにサービスに送信するメッセージ
EM_READ_SERVICE_XML_END	1002		EgBoot がサービスコンフィグファイルを読み終わった際に依頼元のサービスへ送るメッセージ
EM_CORE_CREATE_SERVICE_END	1003		サービスコンフィグファイル内の Service タグを読み終わった際に Create 時のみ対象のサービスに送るメッセージ
EM_DEBUG_WRITE_SERVICEOBJ	9000		サービス内で登録されているオブジェクト情報をログメールボックスに出力

上記以外のシステムメッセージは、サービスごとのマニュアルを参照してください。

4.2. Address 表記

EgTag の Address 表記は IEC 表記に似せて作られています。バイナリデータ列を使った値の読み書きを簡略化できるユーティリティ API 関数で使します。EgEncode()でバイナリデータ列への書き込み、または EgDecore()でバイナリデータ列からのデータ抽出を行う際に使します。データはリトルエンディアンとして参照、書き込みされます。

フォーマット形式は以下で構成されます。

%[①][②].[③]

①: サイズ接頭語 ("X", "B", "W", "D", "L")

②: バイトのメモリアドレス

③: ビット位置 (サイズ接頭語が X:シングルビットサイズのと時のみ定義)

表 5 アドレス表記

サイズ 接頭語	型	説明	例	例説明
X	Boolean	シングルビットサイズ	%X2.4	オフセット 2 の 4 ビット目の値
B	SByte	バイトサイズ(1 バイト)	%B12	オフセット 12 から 1 バイト分の値
	Byte			
W	Int16	ワードサイズ(2 バイト)	%W52	オフセット 52 から 2 バイト分の値
	UInt16			
D	Int32	ダブルワードサイズ(4 バイト)	%D3	オフセット 3 から 4 バイト分の値
	UInt32			
	Float			
L	Int64	ロングワードサイズ(8 バイト)	%L7	オフセット 7 から 8 バイト分の値
	UInt64			
	Double			

4.3. その他構造体

EGTAGTRIGINFO 構造体

```
typedef struct
{
    uint16_t      arrNo;           // ローカルメンバ変数(egTagTriggers)に対応する 配列 No
    union {                       // 変更値 (ただしSegment型、String型は対象外)
        union EgTrigValues
        {
            bool      boolVal;
            float      fltVal;
            double     dblVal;
            char       cVal;
            int16_t     iVal;
            int32_t     lVal;
            int32_t     intVal;
            int64_t     llVal;
            uint8_t     bVal;
            uint16_t     uiVal;
            uint32_t     ulVal;
            uint32_t     uintVal;
            uint64_t     ullVal;
        };
        intptr_t val;
    };
} EGTAGTRIGINFO,* LPEGTAGTRIGINFO;
```

EGTAGTRIGINFOS 構造体 TagTrigger スレッド間データ受け渡し用 1 メッセージ

```
typedef struct
{
    uint16_t      TblCount;           // テーブルの配列数
    EGTAGTRIGINFO pTblTagTriggers[1]; // タグトリガー構造体テーブル
} EGTAGTRIGINFOS,* LPEGTAGTRIGINFOS;
```

S.M.A.R.T.情報全般の構造体 (C#)

```
public struct DiskExecInfo
{
    public Dictionary<int, DiskInfo> diskInfos; // <No, ディスク情報> の連想配列
}
```

S.M.A.R.T.属性の構造体 (C#)

```
public struct SmartAttribute
{
    public int      ID;           // S.M.A.R.T.情報の ID キーとなる
    public string   Status;       // Cur と Thr からチェックされた Status 値
    public string   Cur;          // 現在値
    public string   Wor;          // WORST 値
    public string   Thr;          // Threshold 値
    public string   RawValues;    // 生値
    public string   AttributeName; // 名前・文字列
}
```

ディスク情報の構造体 (C#)

```
public struct SmartAttribute
{
    public string      No;           // No
    public string      Model;        // Model 名
    public string      Firmware;     // Firmware 名
    public string      SerialNumber; // SerialNumber 名
    public string      DiskSize;     // Disk サイズ
    public string      BufferSize;   // Buffer サイズ
    public string      QueueDepth;   // -
    public string      NumberOfSectors; // -
    public string      RotationRate;  // 回転数
    public string      Interface;     // インタフェース
    public string      MajorVersion;  // メジャーバージョン
    public string      MinorVersion;  // マイナーバージョン
    public string      TransferMode;  // 転送モード
    public string      PowerOnHours;  // 使用時間
    public string      PowerOnCount;  // 電源投入回数
    public string      HostWrites;    // 総書込量(ホスト)
    public string      Temperature;   // 温度
    public string      HealthStatus;  // 健康状態
    public string      Features;      // -
    public string      APMLevel;      // -
    public string      AAMLevel;      // -
    public string      DriveLetter;   // ドライブレター
    public Dictionary<int, SmartAttribute> SmartAttributes; // S.M.A.R.T.情報
}
```

ディスクによって取得できないものがあります。ご注意ください。

更新履歴

版	日付	更新説明
1	2021.6	● 初版作成
2	2021.8	● RT-edge Ver.3.3.0 向け変更内容反映
3	2021.10	● RT-edge Ver.3.4.0 向け変更内容反映
4	2022.6	● EgMailboxOpenSC を追加 ● エラーコードリストの変更（更新部分の追加）
5	2023.1	● RT-edge Ver.3.5.1 向け変更内容反映
6	2023.4	● RT-edge Ver.3.5.2 向け変更内容反映
7	2024.8	● 保守用の API を追加
8	2025.3	● Tag の値初期値更新 API を追加
9	2025.8	● RTCD の名称を「RT-edge Object」に変更 ● RT-edge Ver.3.8.3 向け変更内容反映

REALTIME SERVICE for Windows

RT-edge API リファレンスマニュアル

発行元：株式会社マイクロネット

TEL: +81(0)299-90-1733

FAX: +81(0)299-92-8557

- ・ 本書の著作権は、マイクロネットに帰属します。
- ・ 本書の内容、及び付属のソフトウェアの全部または一部を無断で転載することは禁止しております。
- ・ 本製品の内容については、将来予告なしに変更することがあります。
- ・ 本製品の内容について万が一ご不審な点や記載もれなどお気づきの点がございましたら、お手数ですが、当社までご連絡ください。
- ・ Windows XP、Windows 7、Windows 8、Windows 10 等、Windows は、米国 Microsoft Corporation における登録商標です。
- ・ Visual Studio、Visual C++等は、米国、およびその他の国における Microsoft Corporation の登録商標です。
- ・ INtime は米国 TenAsys における登録商標です。
- ・ その他、記載されている会社名、製品名は、各社の商標又は登録商標です