DSHEng5 装置／ホスト通信エンジン・ライブラリ(GEM+GEM300)

ソフトウェア・パッケージ

DSHEng5 GEM 通信エンジン・クラス説明書

Vol－2

変数情報関連クラス

（EC, SV, DVVAL, CE, Report, Alarm）

２０１９年１２月（改訂-1）

株式会社データマップ

文書番号　DSHENG5-19-30322-01

［取り扱い注意］

- この資料ならびにソフトウェアの一部または全部を無断で使用、複製することはできません。
- 本説明書に記述されている内容は予告なしで変更される可能性があります。
- Windows は米国 Microsoft Corporation の登録商標です。
- ユーザーが本ソフトウェアの使用によって生じた遺失履歴、（株）データマップの予見の有無を問わず発生した特別損害、付随的損害、間接損害およびその他の拡大損害に対して責任を負いません。

【改訂履歴】

番号	改訂日付	項 目	概 略
1.	2019-06-28	初版	
2.	2019.12.16	訂正と記述追加	誤字等の訂正 説明の追加など。仕様上は変更はない。
3.			
4.			

目　次

１．はじめに

DSHEng5 GEM 通信エンジン説明書は、Vol-1 から６までの６つの Volume に分けられています。
本説明書の Vol 番号は**2** です。

本説明書では、DSHEng5 通信エンジンのプログラミングで使用できる装置変数 (EC,SV,DV) とレポート (CE,Report) とアラーム(Alarm)情報関連クラスについて機能、コンストラクタ、プロパティ、メソッドなどについて説明します。

Vol 番号	文書番号	内容
Vol-1	DSHENG5-19-30321-00	エンジン起動・停止、通信確立関連クラス (EngAPI、GEM 通信確立、予約装置変数関連)
Vol-2	**DSHENG5-19-30322-00**	**変数情報関連クラス** **(EC, SV, DVVAL, CE, Report, Alarm)**
Vol-3	DSHENG5-19-30323-00	プロセス情報関連クラス (PP, FPP, RECIPE, PRJ, CJ, CARRIER, SUBSTRATE)
Vol-4	DSHENG5-19-30324-00	SECS-II メッセージ送信クラス
Vol-5	DSHENG5-19-30325-00	SECS-II 通信メッセージ情報保存クラス
Vol-6	DSHENG5-19-30326-00	SECS-II 通信メッセージエンコード／デコード処理クラス

２．変数情報管理クラス

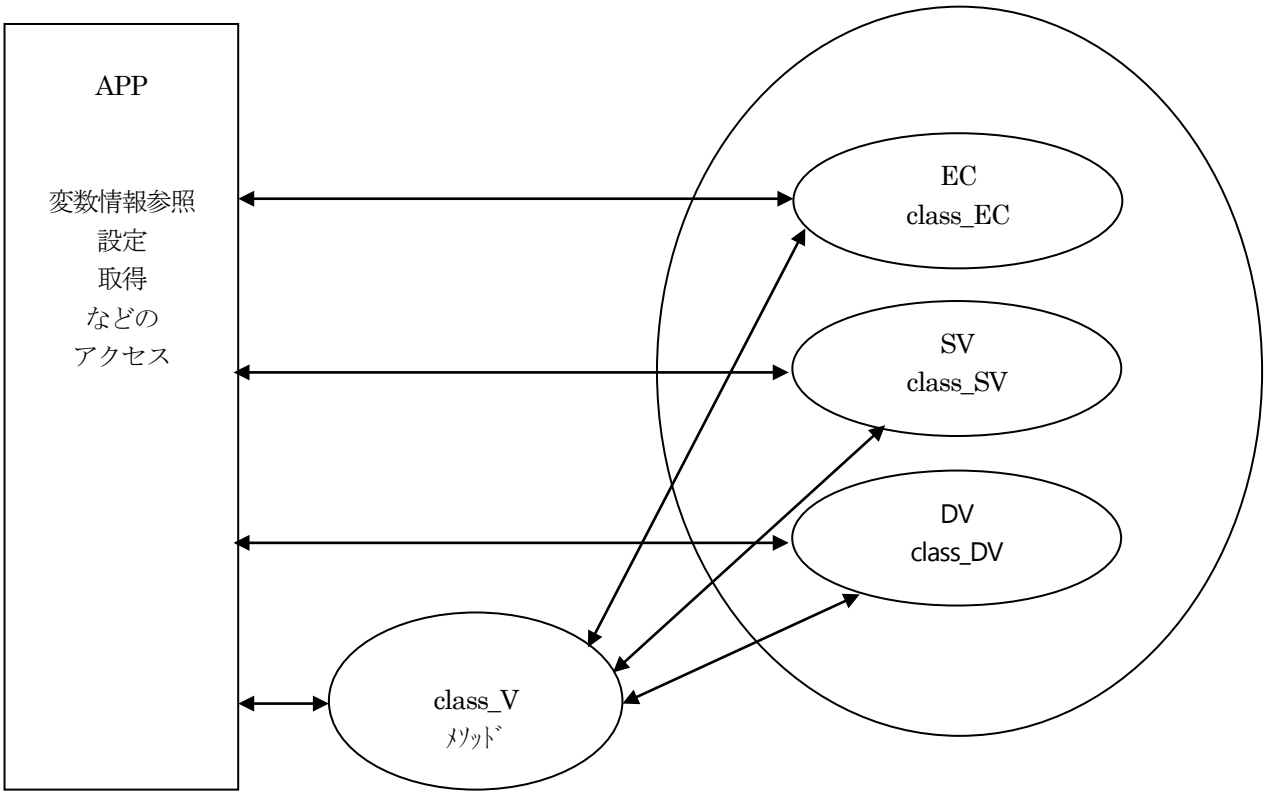
装置変数には以下の３つの種類の変数によって構成されます。

（１）EC － 装置定数
（２）SV － 装置状態変数
（３）DV － DVVAL　データ値変数

これら変数の個々は、それぞれ**固有の整数の ID と名前**を持っています。

変数情報は、**TV_INFO** クラスのインスタンスの中に保存されます。
（TV_INFO クラスについては 3.1 で説明します。）

下図は、各変数とクラス名、class_EC, class_SV そして class_DV を示しています。
そして、これら３つの変数全体にアクセスできるクラスである calss_V を示しています。



APP は変数の参照時、基本的に変数 ID をキーにして変数を特定します。そして、管理されている変数の情報領域をアクセスして変数情報を TV_INFO クラスのインスタンスの中に取得して参照します。

class_V クラスについては、APP が、ID をわかっているが、EC,SV,DV のどの変数であるかわからないときに、その変数を参照するために設けられています。

class_V は、ID から変数の種類 EC,SV,DV のうちのどれかを識別して変数管理情報にアクセスします。

次ページに各変数の format の表記方法を一覧表に示します。

表４ 変数のフォーマットと表記一覧表

番号	format(16 進)	DSHEng5 の表記	意味	
1	00	ICODE_L	リスト	
2	20	ICOE_B	２進	
3	24	ICODE_BOOLEAN	真理値	
4	40	ICODE_A	ASCII 文字	
5	44	ICODE_J	JIS8 文字	
6	60	ICODE_I8	符号付き８バイト整数	
	64	ICODE_I1	符号付き１バイト整数	
7	68	ICODE_I2	符号付き２バイト整数	
8	70	ICODE_I4	符号付き４バイト整数	
9	80	ICODE_F8	８バイト浮動小数点	
10	90	IOCDE_F4	４バイト浮動小数点	
11	A0	ICODE_U8	符号無し８バイト整数	
12	A4	ICODE_U1	符号無し１バイト整数	
13	A8	ICODE_U2	符号無し２バイト整数	
14	B0	ICODE_U4	符号無し４バイト整数	

①これらフォーマットの定義は HSMS クラスで定数として定義されています。

②ユーザがプログラムの中で使用する際は、**HSMS.ICODE_A**　のように表現してください。

```
例　if ( itemcode == HSMS.ICODE_A ){
        …
        ;
    }
```

③ICODE_L　には、その下に０個以上のアイテムがリンクされます。

④ICODE_A, ICODE_J の配列は長さ０文字以上の文字列になります。

⑤③、④以外のフォーマットでは、０個以上の配列データになります。
配列サイズ（要素数）は、プログラムによって変更することができます。
また、配列位置を指定して、その位置のデータの設定／取得もできます。

以下、class_EC, class_SV, class_DV, class_V, TV_INFO などのクラスの詳細について説明します。

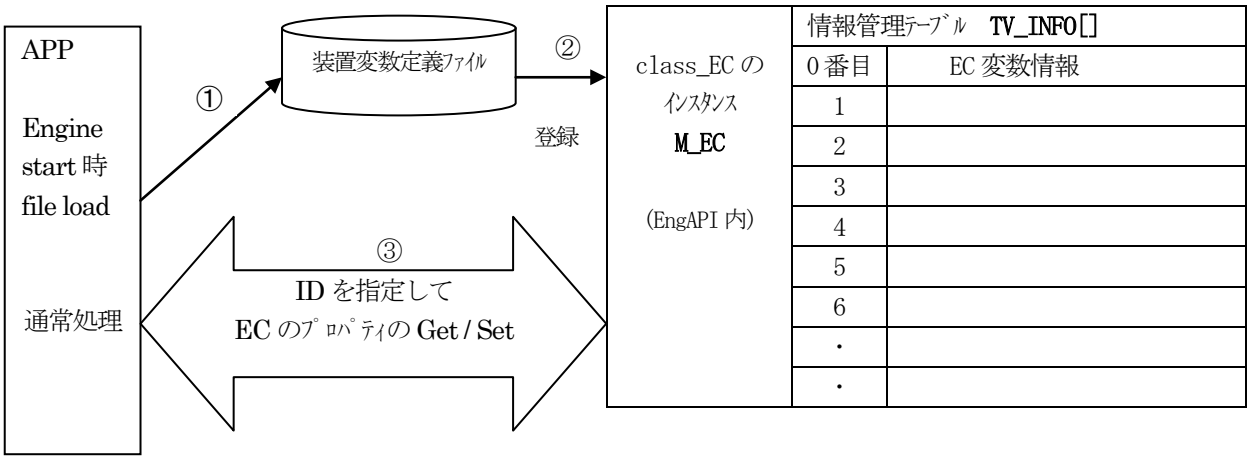
２．１　class_EC - 装置定数クラス

class_EC クラスは全ての**装置定数**の登録、参照、管理サービスを行うためのクラスです。

装置定数（EC）情報は DSHEng5 が開始された際に、装置変数定義ファイルの内容を読み込み、class_EC クラスの中に登録します。その class_EC のインスタンスが **EngAPI. M_EC** です。M_EC は EC 情報の集合です。

APP は、登録されたすべての変数 ID に対して参照（取得、設定）することができます。

EC 情報の構成と参照については概略以下の通りです。



（1）APP は、 EngAPI クラスを使って、エンジンをスタートさせ、装置変数定義ファイルに定義されている全 EC 変数定義情報を読み込み、class_EC クラスのインスタンス **M_EC** の中に登録します。
（M_EC については、６．変数情報保存クラスを参照）

（2）各 ECID の情報はそれぞれ **M_EC** の中に保存され、上図の右の表にあるように ECID 登録順に M_EC インスタンス内の情報管理テーブルに保存されます。

（3）（1）、（2）の後、APP は 各 EC に対して、EC 値の設定／取得などのメソッドを使って各 EC が有するプロパティ情報にアクセスすることができます。（DSHEng5 が開始終了後）

（4）APP が　id＝EC_abc の変数情報を TV_INFO info に取得する場合は、以下のようにプログラミングします。

```
UInt32 EC_abc = 100;
TV_INFO info = new TV_INFO()
int result = EngAPI.M_EC.get( EC_abc, ref info);     // EC_abc の情報を info に取得
if ( result ==0){
   〈処理する〉
   info.Dispose();
}
```

以上の説明内容は、装置変数 SV, DV(Data Value) についても全く同じになります。

２．１．１　コンストラクタ

	名前	説明
1	public class_EC(int max_id)	インスタンスを生成します。 DSHEng5 が開始時に生成します。 引数 max_id は登録できる ID の最大数

class_EC クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)

引数 max_id は管理できる ID の最大数を指定します。プロパティ v_info_tab[] の配列サイズになります。

(EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス表 **M_EC** を生成します。)

２．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1	TV_INFO[] v_info_tab	EC 装置変数の登録テーブル	

TV_INFO クラスについては、**６．１　TV_INFO クラス**　の説明を参照ください。

２．１．３　メソッド

DSHEng5 通信エンジンは、開始処理の中で、**EngAPI** クラス内に classEC のインスタンスを生成します。
インスタンス名は**M_EC** です。M_EC は、static（静的）　メモリ内に生成されます。

すべての EC 変数が、M_EC インスタンスの中に登録され、管理されることになります。

APP は、この**M_EC** インスタンスに対してメソッドを実行することによって特定 ID の EC 変数情報をアクセスすることになります。

EC_Mdln　format-A の値を取得する例は以下のようなコーディングになります。
（EC_Mdln の値は装置定数であり、変数定義ファイルで定義される ID を表す定数です。

```
string value = new string();
int result = EngAPI.M_EC.get_value(EC_MDLN, ref value);
```

APP が使用できる class_EC クラスのメソッドは下記一覧表のとおりです。

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int get_id_name_list()	登録されている ID、名前リストを取得します。
4	public int get_id_by_name()	変数名からその ID を取得します
5	public int get_name()	変数 ID の名前を取得します
6	public int set()	指定 ID に変数情報を設定します。
7	public int get()	指定 ID の変数情報を取得します。 （TV_INFO ： 変数情報用ｸﾗｽ）
8	public int get_format()	指定 ID の変数値フォーマットを取得します。 （TV_INFO ： 変数情報用ｸﾗｽ）
9	public int get_size()	指定 ID の変数値のサイズを取得します。
10	public int get_units()	指定 ID の変数値の物理単位を取得します。
11	public int set_value()	当該変数インスタンスに値を設定します。 （formmat 別に準備されています）
12	public int get_value()	当該変数インスタンスの値を取得します。 （formmat 別にメソッドが準備されてます）
13	public int get_nominal()	指定 ID の規定値を取得します。 （formmat 別に準備されています）
14	public int get_min_value()	指定 ID の最小値を取得します。（制限値） （formmat 別に準備されている）

15	public int get_max_value()	指定 ID の最大値を取得します。(制限値) (formmat 別に準備されています)
16	public int resize_V_array()	指定 ID の配列変数の配列サイズを変更します。
17	public int resize_V_Linklist()	format が L(List)変数にリンクされる変数 ID 数を変更します。
18	public int set_ V_Linklist()	format が L(List)変数にリンクされる変数の ID を 1 個追加設定します。
19	public int set_ V_Linklist()	format が L(List)変数にリンクされる変数の ID を 1 個設定します。(配列位置指定による)
20	public int set_ V_Linklist_all()	format が L(List)変数に全リンク変数 ID を一度に設定します。(リンクできる全変数 ID を)
21	public int set_limit_info()	指定 ID 変数にリミット情報を設定します。 (TLIMIT_INFO : ﾘﾐｯﾄ情報ｸﾗｽ)
22	public int get_limit_info()	指定 ID 変数のリミット情報を取得します。 (TLIMIT_INFO : ﾘﾐｯﾄ情報ｸﾗｽ)
23	public int del_limit_info()	指定 ID 変数のリミット情報を削除します。
24	public int check_val()	指定 ID の値が最小、最大値の範囲にはいっているかどうかを調べます。

２．１．３．１　get_id_count() - 登録されている ID 数の取得

DSHEng5 内に登録されている ECID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されている ECID 数を取得します。

２．１．３．２　get_id_list() - 登録されている ID リストの取得

DSHEng5 内に登録されている ECID のリストを取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大容量

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list 内に取得します。

戻り値は、取得した ID 数です。

２．１．３．３　get_id_name_list() - 登録されている ID,名前リストの取得

EC 変数に登録されているすべての ID とその名前をそれぞれのリストに取得します。

【構文】

public int get_id_name_list(UInt32[] id_list, string[] name_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

name_list

　　変数名を保存するリスト

max_size

　　両方の list の最大サイズ

【戻り値】

返却値	意　味
ID 数	取得した ID 数

【説明】

登録されている ID とその名前をそれぞれ id_list, name_list リストに取得します。
戻り値は、取得した ID 数です。

２．１．３．４　get_id_by_name() - 変数名から ID を取得

変数名から変数 ID を取得します。

【構文】

```
public int get_id_by_name(string vname, ref UInt32 id)
```

【引数】

vname

　　変数名

id

　　ID 格納用領域

【戻り値】

返却値	意　味
0	取得できた。
-1	変数名で指定された変数はなかった。

【説明】

変数名から変数 ID を取得します。
取得できた場合は 0 を、変数名が見つからなかった場合は (-1) を返却します。

２．１．３．５　get_name() - ID から変数名を取得

変数 ID から変数名を取得します。

【構文】

```
public string get_name(UInt32 id)
```

【引数】

id

　　変数 ID

【戻り値】

返却値	意　味
変数名	取得できた。
“”	ID が見つからなかった。

【説明】

変数 ID から変数名を取得します。

２．１．３．６　set()－変数情報の設定

TV_INFO クラスのインスタンスに保存されている変数情報を、指定された ID の変数情報に設定します。

【構文】

public int set(UInt32 id, TV_INFO info)

【引数】

id

　　設定先の変数 ID

info

　　設定したい変数情報

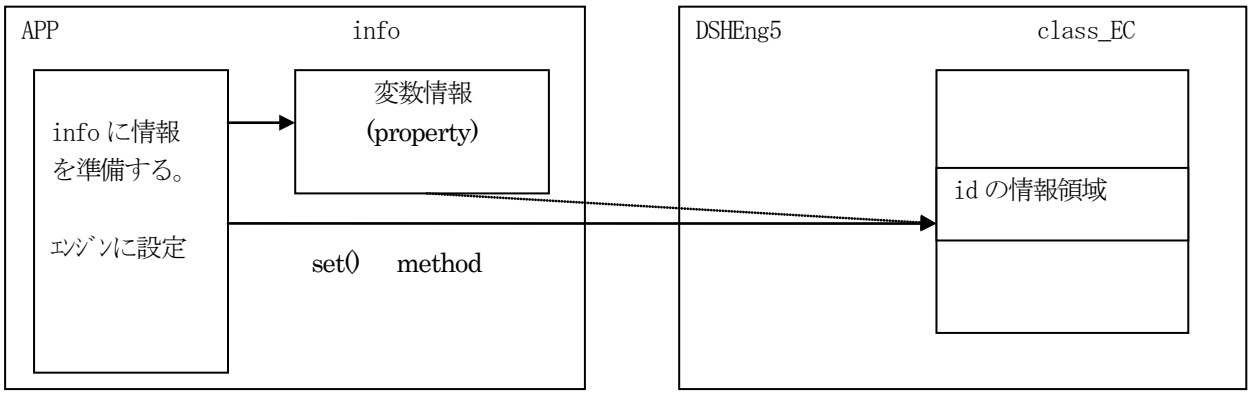
【戻り値】

返却値	意　味
0	設定できた。
-1	①ID が登録されていなかった。 ②info 内の変数値が最小、最大値の範囲を超えていた。

【説明】

info の変数情報の内容を id で指定された EC 変数に設定します。
もし、当該 ID に最小(min)，最大(max)値が設定されていれば値のチェックを行います。

APP は、set メソッドを実行する前に、V_INFO クラスのインスタンス info の中に設定したい変数情報を保存しておく必要があります。



コーディング例は次の通りです。

```
TV_INFO info = new TV_INFO();
UInt32 id = EC_xxxx;        (ECID)

  〈info に設定したい情報を設定します。〉
int result = EngAPI.M_EC.set( id, info);
info.Dispose();
```

TV_INFO クラスについては、**3.1 TV_INFO** を参照して下さい。

２．１．３．７　get()－変数情報の取得

指定された ID の変数情報を EC 情報管理クラスのインスタンス(EngAPI.M_EC)から取得します。

【構文】

public int get(UInt32 id, ref TV_INFO info)

【引数】

id

取得したい変数 ID

info

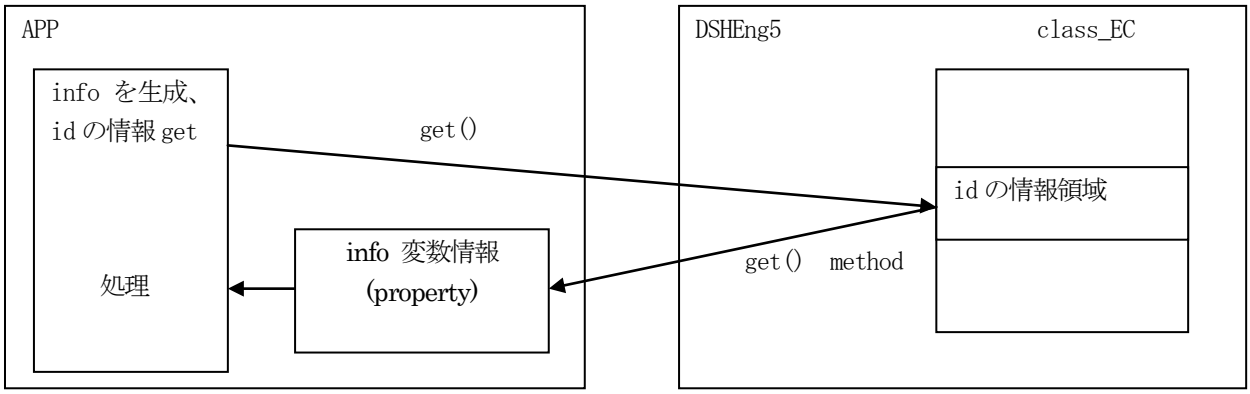
取得変数情報の保存インスタンス

【戻り値】

返却値	意　味
0	取得できた。
-1	取得できなかった。

【説明】

id で指定された EC 変数情報を info で指定された TV_INFO クラスのインスタンスに取得します。



コーディング例は次の通りです。

```
TV_INFO info = new TV_INFO();
UInt32 id = EC_xxxx;        (ECID)
int result = EngAPI.M_EC.get( id, ref info);
if ( result ==0)
{
    〈処理〉
    info.Dispose();
}
```

TV_INFO クラスについては、3.1 を参照して下さい。

２．１．３．８　get_format() - Format の取得

指定された変数 ID のデータフォーマットを取得します。

【構文】

public int get_format(UInt32 id)

【引数】

id

変数 ID

【戻り値】

返却値	意　味
(-1)以外	取得した format。
(-1)	ID が見つからなかった。

【説明】

引数で指定された変数 ID のデータフォーマットを取得します。

２．１．３．９　get_size() - 変数データサイズの取得

指定された変数 ID のデータサイズを取得します。

【構文】

public int get_size(UInt32 id)

【引数】

id

変数 ID

【戻り値】

返却値	意　味
(-1)以外	取得したデータサイズ。 format=A の場合、文字列ﾊﾞｲﾄｻｲｽﾞ。
(-1)	ID が見つからなかった。

【説明】

引数で指定された変数 ID のデータサイズ（配列サイズ）を取得します。

２．１．３．10　get_units() - 物理単位の取得

登録されている変数 ID から物理単位（units）を取得します。

【構文】

public string get_units(UInt32 id)

【引数】

id

変数 ID

【戻り値】

返却値	意　味
物理単位名	取得できた。
“”	ID が見つからなかった。

【説明】

引数で指定された変数 ID の物理単位を取得します。

２．１．３．11　set_value() - 変数値の設定

メソッドの引数で与えられたデータを指定された変数 ID の変数値に設定します。

【構文】

（1）基本のメソッドの構文は次のようになります。

```
public  int set_value(UInt32 id, IntPtr value)
public  int set_value(UInt32 id, IntPtr value, int size)
public  int set_value(UInt32 id, IntPtr value, int pos, int size)
```

設定値は IntPtr が指すメモリから取り出してデータフォーマットに合わせて設定します。

（2）データフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B U1	set_value(UInt32 id, byte value) set_value(UInt32 id, byte value, int pos) set_value(UInt32 id, byte[] value, int size)	
Boolean	set_value(UInt32 id, bool value) set_value (UInt32 id, bool value, int pos) set_value(UInt32 id, bool[] value, int size)	
I1	set_value(UInt32 id, Int8 value) set_value(UInt32 id, Int8 value, int pos) set_value(UInt32 id, Int8[] value, int size)	
U2	set_value(UInt32 id, UInt16 value) set_value(UInt32 id, UInt16 value, int pos) set_value(UInt32 id, UInt16[] value, int size)	
I2	set_value(UInt32 id, Int16 value) set_value(UInt32 id, Int16 value, int pos) set_value(UInt32 id, Int16[] value, int size)	
U4 L	set_value(UInt32 id, UInt32 value) set_value(UInt32 id, UInt32 value, int pos) set_value(UInt32 id, UInt32[] value, int size)	
I4	set_value(UInt32 id, Int32 value) set_value(UInt32 id, Int16 value, int pos) set_value(UInt32 id, Int32[] value, int size)	
U8	set_value(UInt32 id, UInt64 value) set_value(UInt32 id, UInt64 value, int pos) set_value(UInt32 id, UInt64[] value, int size)	
I8	set_value(UInt32 id, Int64 value) set_value(UInt32 id, Int64 value, int pos) set_value(UInt32 id, Int64[] value, int size)	
F4	set_value(UInt32 id, float value) set_value(UInt32 id, float value, int pos) set_value(UInt32 id, float[] value, int size)	

F8	set_value(UInt32 id, double value) set_value(UInt32 id, double value, int pos) set_value(UInt32 id, double[] value, int size)	
A	set_value(UInt32 id, string value)	string の配列はｻﾎﾟｰﾄしません。

【引数】

id

　　変数 ID

value

　　変数データ（配列データ）

size

　　データ数

pos

　　設定したい配列位置

【戻り値】

返却値	意　味
0	設定できた。
1	変数の定義サイズより size が小さかった。
(-1)	ID が登録されていなかった。または pos 位置エラーであった。
(-2)	size が変数の定義サイズより大きかった。

【説明】

引数 value で与えられたデータを id で指定された変数の値に設定します。

変数値が 1 個の場合は、value の値をそのまま設定します。

変数値が配列データの場合は、value 配列データの size で指定された数だけのデータを設定します。
ただし、変数値の規定配列サイズ(=要素数)を asize とした場合、asize と size の値によって、以下のように処理を行います。

（1）asize = size　　：　asize 分のデータを設定します。
（2）asize < size　　：　データの設定は行いません。そして、戻り値 = (-2)を返します。
（3）asize > size　　：　size 分だけのデータを設定します。そして、戻り値 = 1　を返します。

２．１．３．12　get_value() - 変数値の取得

登録されている変数 ID の変数値をメソッドの引数に指定されたメモリに取得します。

【構文】

（1）基本のメソッドの構文は次のようになります。

```
public  int get_value(UInt32 id, IntPtr value)
public  int get_value( UInt32 id, int pos, IntPtr value)
public  int get_value(UInt32 id, IntPtr value, int size)
```

値は、エンジンが管理している変数情報領域から取得します。

（2）変数フォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B U1	get_value(UInt32 id, ref byte value) or get_value(UInt32 id, ref byte value, int pos) set_value(UInt32 id, byte[] value, int size)	
Boolean	get_value(UInt32 id, ref bool value) get_value(UInt32 id, ref bool value, int pos) get_value(UInt32 id, bool[] value, int size)	
I1	get_value(UInt32 id, ref Int8 value) get_value(UInt32 id, ref Int8 value, int pos) get_value(UInt32 id, Int8[] value, int size)	
U2	get_value(UInt32 id, ref UInt16 value) get_value(UInt32 id, ref UInt16 value, int pos) get_value(UInt32 id, UInt16[] value, int size)	
I2	get_value(UInt32 id, ref Int16 value) get_value(UInt32 id, ref Int16 value, int pos) get_value(UInt32 id, Int16[] value, int size)	
U4	get_value(UInt32 id, ref UInt32 value) get_value(UInt32 id, ref UInt32 value, int pos) get_value(UInt32 id, UInt32[] value, int size)	
I4 L	get_value(UInt32 id, ref Int32 value) get_value(UInt32 id, ref Int32 value, int pos) get_value(UInt32 id, Int32[] value, int size)	
U8	get_value(UInt32 id, ref UInt64 value) get_value(UInt32 id, ref UInt64 value, int pos) get_value(UInt32 id, UInt64[] value, int size)	
I8	get_value(UInt32 id, ref Int64 value) get_value(UInt32 id, ref Int64 value, int pos) get_value(UInt32 id, Int648[] value, int size)	
F4	get_value(UInt32 id, ref float value) get_value(UInt32 id, ref float value, int pos) get_value(UInt32 id, float[] value, int size)	

F8	get_value(UInt32 id, ref double value) get_value(UInt32 id, ref double value, int pos) get_value(UInt32 id, double[] value, int size)	
A	get_value(UInt32 id, ref string value)	string の配列は ｻﾎﾟｰﾄしません。

【引数】

id

　　値を取得したい変数 ID

value

　　変数データまたは配列データの格納領域

size

　　取得したいデータ数

pos

　　配列の取得位置

【戻り値】

返却値	意　味
0	取得できた。
1	size が変数の定義サイズより小さかった。
2	size が変数の定義サイズより大きかった。
(-1)	ID が登録されていなかった。または配列の取得位置エラー

【説明】

id で指定された変数の値を value に取得します。

変数値が 1 個の場合は、変数値をそのまま value に取得します。

変数値が数値データで、しかも配列データの場合は、size で指定された数だけ value 配列データを取得します。
ただし、変数値の配列サイズを asize とした場合、asize と size の値によって、以下の処理を行います。

（1）asize = size　　：　asize 分のデータを取得します。
（2）asize < size　　：　asize 分のデータを取得し、戻り値 = 2 を返します。
（3）asize > size　　：　size 分だけのデータを取得し、戻り値 = 1 を返します。

変数が配列データの場合、value の配列サイズは登録変数データを充分格納できるサイズでなければなりません。

２．１．３．13　get_nominal()－変数規定値の取得

登録されている変数 ID の規定値（nominal）をメソッドの引数に指定された領域に取得します。

【構文】

（1）基本のメソッドは次に構文になります。

public　int get_nominal(UInt32 id, IntPtr value)

規定値は、エンジンが管理している変数情報領域から取得します。

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B, U1	get_nominal(UInt32 id, ref byte value)	
Boolean	get_nominal(UInt32 id, ref bool value)	
I1	get_nominal(UInt32 id, ref Int8 value)	
U2	get_nominal(UInt32 id, ref UInt16 value)	
I2	get_nominal(UInt32 id, ref Int16 value)	
U4	get_nominal(UInt32 id, ref UInt32 value)	
I4、L	get_nominal(UInt32 id, ref Int32 value)	
U8	get_nominal(UInt32 id, ref UInt64 value)	
I8	get_nominal(UInt32 id, ref Int64 value)	
F4	get_nominal(UInt32 id, ref float value)	
F8	get_nominal(UInt32 id, ref double value)	
A	get_nominal(UInt32 id, ref string value)	

【引数】

id

　　規定値を取得したい変数 ID

value

　　変数規定値の格納領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	ID が登録されていなかった。

【説明】

id で指定された変数の規定値を引数 value に取得します。

２．１．３．14　get_min() - 変数最小値の取得

登録されている変数 ID の最小値（min）をメソッドの引数に指定された領域に取得します。

【構文】

（1）基本のメソッドは次に構文になります。

```
public  int get_min(UInt32 id, IntPtr value)
```

最小値は、エンジンが管理している変数情報領域から取得します。

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B, U1	get_min(UInt32 id, ref byte value)	
Boolean	get_min(UInt32 id, ref bool value)	
I1	get_min(UInt32 id, ref Int8 value)	
U2	get_min(UInt32 id, ref UInt16 value)	
I2	get_min(UInt32 id, ref Int16 value)	
U4	get_min(UInt32 id, ref UInt32 value)	
I4, L	get_min(UInt32 id, ref Int32 value)	
U8	get_min(UInt32 id, ref UInt64 value)	
I8	get_min(UInt32 id, ref Int64 value)	
F4	get_min(UInt32 id, ref float value)	
F8	get_min(UInt32 id, ref double value)	
A	get_min(UInt32 id, ref string value)	

【引数】

id

最小値を取得したい変数 ID

value

変数最小値の格納領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	ID が登録されていなかった。

【説明】

id で指定された変数の最小値を引数 value に取得します。

２．１．３．15　get_max() - 変数最大値の取得

登録されている変数 ID の最大値（max）をメソッドの引数に指定された領域に取得します。

【構文】

（1）基本のメソッドは次に構文になります。

```
public  int get_max(UInt32 id, IntPtr value)
```

最大値は、エンジンが管理している変数情報領域から取得します。

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B, U1	get_max(UInt32 id, ref byte value)	
Boolean	get_max(UInt32 id, ref bool value)	
I1	get_max(UInt32 id, ref Int8 value)	
U2	get_max(UInt32 id, ref UInt16 value)	
I2	get_max(UInt32 id, ref Int16 value)	
U4	get_max(UInt32 id, ref UInt32 value)	
I4, L	get_max(UInt32 id, ref Int32 value)	
U8	get_max(UInt32 id, ref UInt64 value)	
I8	get_max(UInt32 id, ref Int64 value)	
F4	get_max(UInt32 id, ref float value)	
F8	get_max(UInt32 id, ref double value)	
A	get_max(UInt32 id, ref string value)	

【引数】

id

最大値を取得したい変数 ID

value

変数最大値の格納領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	ID が登録されていなかった。

【説明】

id で指定された変数の最大値を引数 value に取得します。

２．１．３．16　resize_V_array() - 変数の配列サイズの変更

指定された ID の変数データ配列サイズを変更します。

【構文】

public static int resize_V_array(uint vid, int new_size)

【引数】

vid

　　変数 ID

new_size

　　新しい配列サイズ

【戻り値】

返却値	意　味
0	変更できた。
(-1)	変更できなかった。 (ID が見つからなかった、または format が　A, L であった)

【説明】

vid で指定された数値変数のデータ配列サイズを　new_size に変更します。

変更されると、配列要素の値はすべてクリア (=0) されます。

変数フォーマット A, L 以外の変数が本メソッドの対象になります。

(例) 変数 ID　100, format = ICODE_I4, asize (配列サイズ) =1 の変数の配列サイズ 16 の配列に変更する場合は以下のようにコーディングします。

```
UInt32  vid = 100;
int result = EngAPI.M_EC.resize_V_array( vid, 16);
```

これで、変数 vid=100 の変数値の配列サイズが 16 になります。

(value[1]　==>　value[16])

２．１．３．17　resize_V_Linklist()－L 変数のリンク配列サイズの変更

L-フォーマットの変数のリンク VID 格納リストの配列サイズを変更します。

【構文】

public static int resize_V_Linklist(uint vid, int new_size)

【引数】

vid

　　L-フォーマットの変数 ID

new_size

　　新しいリンク配列サイズ

【戻り値】

返却値	意　味
0	変更できた。
(-1)	変更できなかった。 (ID が見つからなかった、または format が　L 以外であった)

【説明】

vid で指定されたデータ配列の中には、その vid にリンクされる他の変数(EC,SV,DV)の ID が格納されます。

vid で指定された数値変数のデータ配列サイズを　new_size に変更します。
変更されると、配列データの各要素の値はすべてクリア (=0) されます。

もし、配列サイズを 0 にしたい場合は、new_size=0 に設定してください。(リンク変数 ID がない)

vid で指定された変数のフォーマットは、”L” でなければなりません。

(例) 変数 ID　100, format = ICODE_L, asize (配列サイズ) =1 の変数 ID 配列サイズを 5 個にします。
そのあと、vid = 200, 201, 202, 203, 204 のを vid=100 のリンクリストの追加する
場合は以下のようにコーディングします。

```
UInt32  vid = 100;                    // format =ICODE_L
UInt32[]  link_vid_list = { 200, 201, 202, 203, 204, 205 };
int result = EngAPI.M_EC.resize_V_linklist( vid, 5);          // linklist を要素数 5 にする
..
result = EngAPI.M_EC.set_V_linklist_all( vid, link_vid_list, 5);
```

これで vid 変数に lik_vid_list[]内の変数 ID が設定されます。

２．１．３．18　add_V_Linklist() - の変数のリンクリストに変数 ID を追加

L-フォーマットの変数のリンク変数 ID リストに 1 個の変数 ID を追加します。

【構文】

public int add_V_Linklist(uint id, uint link_vid)

【引数】

vid

　　L-フォーマットの変数 ID

link_vid

　　vid 変数に追加リンクしたい変数 ID

【戻り値】

返却値	意　味
0	設定できた。
(-1)	以下の１つの条件 (1)vid が登録されていなかった。 (2)vid のフォーマットが L ではなかった。 (3)link_vid が登録されていなかった。 (4)リストが満杯であった。

【説明】

vid で指定された L-フォーマット変数の VID リンクリスト配列内に 1 個の変数 ID を追加します。
追加は、リストの先頭から順に行います。
追加は、配列リストのサイズ分になるまで実行してください。

追加できた場合、戻り値　0 を返却します。
追加できなかった場合は、上で示した戻り値を返却します。

２．１．３．19　set_V_Linklist() - 変数リンクリストの指定位置に vid 設定

L-フォーマットの変数の VID リンクリストの指定配列位置に変数 ID を設定します。

【構文】

public int set_V_Linklist(uint vid, uint link_vid, int pos)

【引数】

vid

　　L-フォーマットの変数 ID

link_vid

　　vid 変数にリンクしたい変数 ID

pos

　　リンクリスト配列位置(0,..)

【戻り値】

返却値	意　味
0	設定できた。
(-1)	以下の１つの条件 (1)vid が登録されていなかった。 (2)vid のフォーマットが L ではなかった。 (3)link_vid が登録されていなかった。 (4)pos が配列リスト外の位置であった。

【説明】

vid で指定された L-フォーマット変数のリンクリストに link_vid を設定します。
設定する配列位置は pos で指定します。

設定できた場合、戻り値　0 を返却します。

設定できない条件を検出した場合は、上で示した戻り値を返却します。

２．１．３．20　set_V_Linklist_all() - の変数のリンクリストに全変数 ID を設定

L-フォーマットの変数の VID リンクリストにまとめてリンク変数 ID を設定します。

【構文】

public int set_V_Linklist_all(uint id, uint[] link_vid_list, int size)

【引数】

vid

　　L-フォーマットの変数 ID

link_vid_list

　　vid 変数にリンクしたい変数 ID の配列

size

　　設定するリンク VID の数　（ >= 0 ）

【戻り値】

返却値	意　味
0	設定できた。
(-1)	エラー、以下の中の１つの条件 (1)vid が登録されていなかった。 (2)vid のフォーマットが L ではなかった。 (3)link_vid_list 内の vid が登録されていなかった。 (4)size=0 または設定したいサイズが設定できるサイズを超えていた。

【説明】

vid で指定された変数のリンク VID 配列に、引数 link_vid_list に含まれる size 個分の変数 ID を設定します。

以下の場合は、エラーを返却します。

　　①引数 vid 変数のフォーマットが L でなかった。

　　②引数の size=0 であった。

　　③引数 size が設定できるサイズを超えていた。

正常に設定できた場合、0 を返却します。

２．１．３．21　set_limit_info()－の変数リミット情報の設定

IDを指定して変数にリミット監視情報を設定します。

【構文】

public int set_limit_info(UInt32 id, TLIMIT_INFO info)

【引数】

id

　　変数ID

info

　　リミット情報クラス TLIMIT_INFOのインスタンス

【戻り値】

返却値	意　味
0	設定できた。
(-1)	(1)vidが登録されていなかった。 (2)vidのフォーマットが 数値のフォーマットでなかった。 　　(L, A, J, BOOLEAN,はLimit監視対象にならない)

【説明】

変数のリミット監視のためのリミット情報を設定します。

リミット情報は、TLIMIT_INFOクラスのインスタンスです。 TLIMIT_INFOクラスについては ６．11 で説明します。

設定できない条件を検出した場合は、上で示した戻り値を返却します。

２．１．３．22　get_limit_info() - の変数リミット情報の取得

ID を指定して変数のリミット情報を取得します。

【構文】

public int get_limit_info(UInt32 id, ref TLIMIT_INFO info)

【引数】

id

変数 ID

info

取得したリミット情報を保存するための TLIMIT_INFO のクラスのインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	(1)vid が登録されていなかった。 (2)リミット情報が無かった。

【説明】

変数のリミット情報を取得します。

２．１．３．23　del_limit info() - の変数リミット情報の取得

ID を指定して変数のリミット情報を削除します。

【構文】

public int del_limit_info(UInt32 id)

【引数】

id

変数 ID

【戻り値】

返却値	意　味
0	削除できた。(常時)

【説明】

変数のリミット情報を削除します。

返却値は、変数 ID が未登録、リミット情報が無かったなどの場合でも 0 を返します。

２．２　class_SV - 装置状態変数クラス

class_SV クラスは GEM が定める装置状態変数に関する設定、参照などのサービスを行うためのクラスです。

APP は、登録されたすべての変数 ID に対して参照（取得、設定）することができます。

変数の構成と参照については　**2. class_EC** で説明した内容と同じです。
また、プロパティ、メソッドについても同様に、classs_EC の説明と同様になりますので、そちら参照ください。

class_SV のインスタンスは、**EngAPI** クラスのプロパティ **M_SV** になります。
APP は、このインスタンス、**M_SV** に対してメソッドを実行することによって特定 ID の SV 変数にアクセスすることができます。

SV_CarID　format-A の値を取得する例は以下のようなコーディングになります。

```
string value = new string();
int result = EngAPI.M_SV.get_value( SV_CarID, ref value);
```

２．３　class_DV - 装置データ値変数クラス

class_DV クラスは GEM が定める装置データ値(DVVAL)変数に関する設定、参照などのサービスを行うためのクラスです。

APP は、登録されたすべての変数 ID に対して設定、参照することができます。

変数の構成と参照については **2.１　class_EC** で説明した内容と同じです。
また、プロパティ、メソッドについても同様に、classs_EC の説明と同様になりますので、そちら参照ください。

class_DV のインスタンスは、**EngAPI** クラスのプロパティ **M_DV** になります。
APP は、このインスタンス、**M_DV** に対してメソッドを実行することによって特定 ID の DV 変数にアクセスすることができます。

２．４　class_V クラス － 変数（EC, SV, DV）共通

class_V は、EC,SV,DV 全装置変数情報に、変数の種別を意識しないで、アクセスする手段を与えるクラスです。

アクセスは本クラスの中に準備されているメソッドを使って行うことができます。

APP が利用できる一部を除くすべてのメソッドに、引数として変数 ID が含まれます。

２．４．１　コンストラクタ

本クラスは、プロパティが特に有しておらず、APP に提供するメソッドはすべて **static のメソッド**になっていますので、インスタントを生成する必要はありません。

コンストラクタだけが、準備されています。

```
public class_V(){;}
```

２．４．２　プロパティ

本クラスにはプロパティはありません。

２．４．３　メソッド

APP が使用できる class_V クラスでは class_EC クラスで説明したメソッドと同様の機能を有するメソッドが準備されています。

class_V に含まれるメソッドは static のものです。
そのため、変数にアクセスするためには、class_V.set_value() のように実行します。

例えば、変数 ID が format –A、名前が EC_Mdln 変数の値を取得するには以下のようにコーディングします。

```
string str = new string();
int result = class_V.get_value( EC_Mdln, ref str);
```

（**class_V クラス名が競合**する場合は、 DSH_ENG.class_V と表記してください。）

すべて、static 、静的メソッドになります。

基本のメソッドの構文は次のようになります。

```
public static int set_value(UInt32 id, IntPtr value)
public static int set_value(UInt32 id, IntPtr value, int size)
public static int set_value(UInt32 id, IntPtr value, int pos, int size)
```

IntPtr が指すメモリに格納されている変数値を、変数 ID の id のプロパティ変数値(value)に設定します。

APP が使用できる class_V クラスのメソッドは次の一覧表のとおりです。

	メソッド名	説明
1	public static int get_id_count()	登録されている ID 数を取得します。 (EC, SV, DV すべて)
2	public static int get_id_list()	登録されている ID リストを取得します。 (EC, SV, DV すべて)
3	public static int get_id_name_list()	登録されている ID,名前リストを取得します。 (EC, SV, DV すべて)
4	public static int get_id_by_name()	変数名からその ID を取得します
5	public static int get_name()	変数 ID の名前を取得します
6	public static int set()	変数情報を指定 ID に設定します。
7	public static int get()	指定 ID の変数情報を取得します。 (TV_INFO : 変数情報用ｸﾗｽ)
8	public static int get_format()	指定 ID の変数値フォーマットを取得します。 (TV_INFO : 変数情報用ｸﾗｽ)
9	public static int get_size()	指定 ID の変数値のサイズを取得します。
10	public static int get_units()	指定 ID の変数値の物理単位を取得します。
11	public static int set_value()	当該変数インスタンスに値を設定します。 (formmat 別に準備されています)
12	public static int get_value()	当該変数インスタンスの値を取得します。 (formmat 別に準備されてます)
13	public static int get_nominal()	指定 ID の規定値を取得します。 (formmat 別に準備されている)
14	public static int get_min_value()	指定 ID の最小値を取得します。(制限値) (formmat 別に準備されている)
15	public static int get_max_value()	指定 ID の最大値を取得します。(制限値) (formmat 別に準備されている)
16	public static int resize_V_array()	指定 ID の配列変数の配列を変更します。
17	public static int resize_V_Linklist()	format が L(List)変数にリンクされる変数 ID 数を変更します。
18	public static int add_V_Linklist()	format が L(List)変数にリンクされる変数の ID を 1 個追加設定します。
19	public static int set_V_Linklist()	format が L(List)変数にリンクされる変数の ID を 1 個設定します。(配列位置指定による)
20	public static int add_V_Linklist_all()	format が L(List)変数に全リンク変数 ID を一度に設定します。(リンクできる全変数 ID を)
21	public static int set_limit_info()	指定 ID 変数にリミット情報を設定します。 (TLIMIT_INFO : ﾘﾐｯﾄ情報ｸﾗｽ)

22	public static int get_limit_info()	指定 ID 変数のリミット情報を取得します。 （TLIMIT_INFO ： ﾘﾐｯﾄ情報ｸﾗｽ）
23	public static int del_limit_info()	指定 ID 変数のリミット情報を削除します。

２．４．３．１　get_id_count()－登録されているID数の取得

DSHEng5内に登録されているすべての装置変数ID数を取得します。
EC, SV, DVの合計ID数になります。

【構文】

public static int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID数	登録ID数

【説明】

登録されている変数ID数を取得します。

２．４．３．２　get_id_list()－登録されているIDリストの取得

DSHEng5内に登録されている変数IDのリストを取得します。
EC,SV, DV順に全変数IDを取得します。

【構文】

public static int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list
　　IDを保存するリスト
max_size
　　id_list配列の最大容量

【戻り値】

返却値	意　味
ID数	id_listリストに取得したID数

【説明】

登録されている全変数IDをid_list内に取得します。
戻り値は、取得したID数です。

２．４．３．３　get_id_name_list() - 登録されている ID,名前リストの取得

変数変数に登録されているすべての ID とその名前をそれぞれのリストに取得します。
EC,SV, DV 順に全変数の ID と変数名を取得します。

【構文】

```
public static int get_id_name_list(UInt32[] id_list, string[] name_list, int max_size)
```

【引数】

id_list
　　ID を保存するリスト
name_list
　　変数名を保存するリスト
max_size
　　両方の list の最大サイズ

【戻り値】

返却値	意　味
ID 数	取得した ID 数

【説明】

登録されているすべての ID とその名前をそれぞれ id_list, name_list 内に取得します。
戻り値は、取得した ID 数です。

２．４．３．４　get_id_by_name() - 変数名から ID を取得

登録されている全変数の中から、指定された変数名の変数 ID を取得します。

【構文】

```
public static int get_id_by_name(string vname, ref UInt32 id)
```

【引数】

vname

　　変数名

id

　　ID 格納用領域

【戻り値】

返却値	意　味
0	取得できた。
-1	変数名で指定された変数はなかった。

【説明】

変数名から変数 ID を取得します。
取得できた場合は 0 を、変数名が見つからなかった場合は (-1)を返却します。

２．４．３．５　get_name() - ID から変数名を取得

登録されている全変数 ID から変数名を取得します。

【構文】

```
public static string get_name(UInt32 id)
```

【引数】

id

　　変数 ID

【戻り値】

返却値	意　味
変数名	取得できた。
“”	ID が見つからなかった。

【説明】

変数 ID から変数名を取得します。

２．４．３．６　set() - 変数情報の設定

指定された ID に、TV_INFO クラスのインスタンスに設定された変数情報を設定します。

【構文】

public static int set(UInt32 id, TV_INFO info)

【引数】

id

　　設定先の変数 ID

info

　　設定したい変数情報

【戻り値】

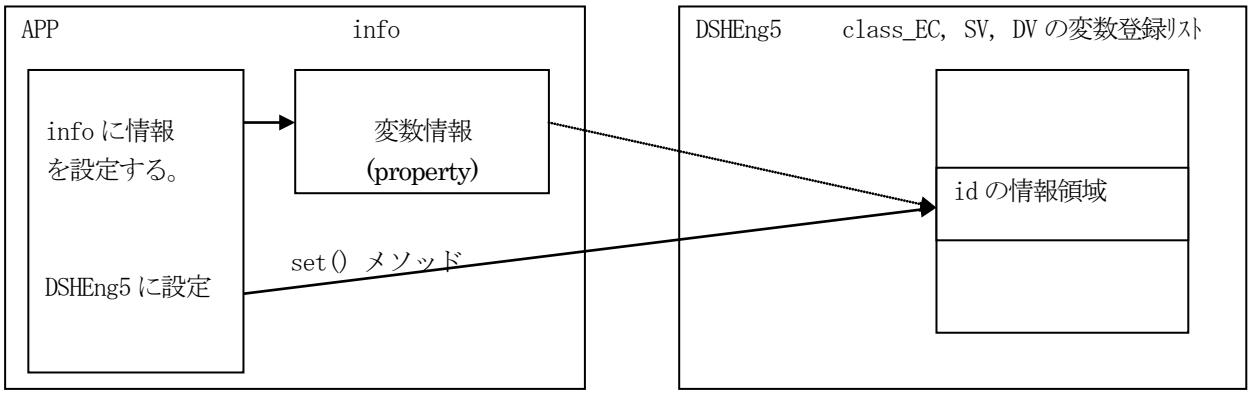
返却値	意　味
0	設定できた。
-1	①ID が登録されていなかった。 ②最小、最大値の範囲を超えていた。

【説明】

info 内の変数情報を id で指定された変数に設定します。

設定前に、もし、当該 ID に最小(min)，最大(max)値の設定があれば、チェックを行います。

APP は TV_INFO クラスのインスタンス info の中に、変数情報を設定した後、本 set メソッドを実行することになります。



コーディング例は次の通りです。

```
TV_INFO info = new TV_INFO();
UInt32 id = EC_xxxx;        (ECID)
  <info に設定したい情報を設定します。>

int result = class_V.set( id, info);       // EC_xxxx の変数に情報を設定
info.Dispose();
```

TV_INFO クラスについては、**6.1 TV_INFO** を参照して下さい。

２．４．３．７　get() - 変数情報の取得

指定された ID の変数情報を引数 info で指定されたインスタンスに取得します。

【構文】

```
public static int get(UInt32 id, ref TV_INFO info)
```

【引数】

id

　　取得したい変数 ID

info

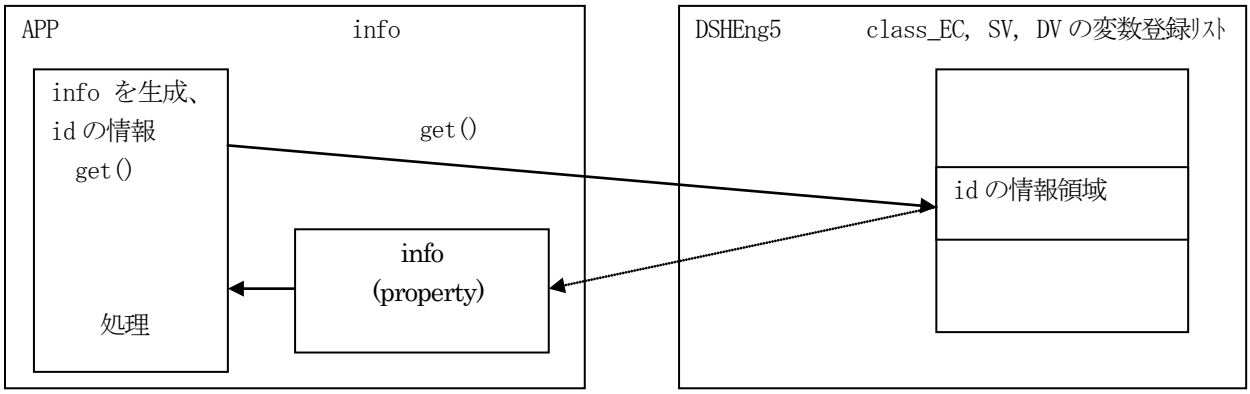
　　取得変数情報保存用インスタンス

【戻り値】

返却値	意　味
0	取得できた。
-1	取得できなかった。

【説明】

引数 id で指定された変数の情報を、引数 info に取得します。



コーディング例は次の通りです。

```
TV_INFO info = new TV_INFO();
UInt32 id = EC_xxxx;        (ECID)
int result = class_V.get( id, ref info);
if ( result ==0)
{
    〈処理〉
    info.Dispose();
}
```

TV_INFO クラスについては、6.1 を参照して下さい。

２．４．３．８　get_format() - フォーマットの取得

指定された変数 ID のフォーマットを取得します。

【構文】

```
public static int get_format(UInt32 id)
```

【引数】

id

変数 ID

【戻り値】

返却値	意　味
(-1)以外	取得した format。
(-1)	ID が見つからなかった。

【説明】

変数 ID のフォーマットを取得します。

２．４．３．９　get_size() - 変数データサイズの取得

指定された変数 ID のデータサイズを取得します。

【構文】

```
public static int get_size(UInt32 id)
```

【引数】

id

変数 ID

【戻り値】

返却値	意　味
(-1)以外	取得したデータサイズ。 format=A の場合、文字列ﾊﾞｲﾄｻｲｽﾞになる。
(-1)	ID が見つからなかった。

【説明】

変数 ID からデータサイズ（配列サイズ）を取得します。

２．４．３．10　get_units() - 物理単位の取得

指定された変数 ID の変数値の物理単位（units）を取得します。

【構文】

public static string get_units(UInt32 id)

【引数】

id

変数 ID

【戻り値】

返却値	意　味
物理単位名	取得できた。
“”	ID が登録されていなかった。

【説明】

変数 ID から物理単位（文字列）を取得します。

２．４．３．11　set_value() - 変数値の設定

メソッドの引数で与えられたデータを変数 ID が指定する変数値に設定します。

【構文】

（1）基本のメソッドの構文は次のようになります。

```
public static  int set_value(UInt32 id, IntPtr value)
public static  int set_value(UInt32 id, IntPtr value, int size)
public static  int set_value(UInt32 id, IntPtr value, int pos, int size)
```

設定値は IntPtr が指すメモリから取り出し設定します。

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B U1	set_value(UInt32 id, byte value) set_value(UInt32 id, byte value, int pos) set_value(UInt32 id, byte[] value, int size)	
Boolean	set_value(UInt32 id, bool value) set_value (UInt32 id, bool value, int pos) set_value(UInt32 id, bool[] value, int size)	
I1	set_value(UInt32 id, Int8 value) set_value(UInt32 id, Int8 value, int pos) set_value(UInt32 id, Int8[] value, int size)	
U2	set_value(UInt32 id, UInt16 value) set_value(UInt32 id, UInt16 value, int pos) set_value(UInt32 id, UInt16[] value, int size)	
I2	set_value(UInt32 id, Int16 value) set_value(UInt32 id, Int16 value, int pos) set_value(UInt32 id, Int16[] value, int size)	
U4	set_value(UInt32 id, UInt32 value) set_value(UInt32 id, UInt32 value, int pos) set_value(UInt32 id, UInt32[] value, int size)	
I4, L	set_value(UInt32 id, Int32 value) set_value(UInt32 id, Int16 value, int pos) set_value(UInt32 id, Int32[] value, int size)	
U8	set_value(UInt32 id, UInt64 value) set_value(UInt32 id, UInt64 value, int pos) set_value(UInt32 id, UInt64[] value, int size)	
I8	set_value(UInt32 id, Int64 value) set_value(UInt32 id, Int64 value, int pos) set_value(UInt32 id, Int64[] value, int size)	
F4	set_value(UInt32 id, float value) set_value(UInt32 id, float value, int pos) set_value(UInt32 id, float[] value, int size)	

F8	set_value(UInt32 id, double value) set_value(UInt32 id, double value, int pos) set_value(UInt32 id, double[] value, int size)	
A	set_value(UInt32 id, string value)	string の配列はｻﾎﾟｰﾄしません。

【引数】

id

　　設定したい変数 ID

value

　　設定したい変数データまたは配列データ

size

　　データ数

pos

　　設定したい配列位置

【戻り値】

返却値	意　味
0	設定できた。
1	変数の定義サイズより size が小さかった。
(-1)	ID が登録されていなかった。または pos 位置エラーであった。
(-2)	size が変数の定義サイズより大きかった。

【説明】

value で与えられたデータを id で指定された変数値に設定します。

変数値が 1 個の場合は、value の値をそのまま設定します。

変数値が配列データの場合は、size で指定された数だけの value データを配列に設定します。
ただし、変数値の配列サイズ-を asize とした場合、asize と size の値によって、以下の処理を行います。

（1）asize = size　　：　asize 分のデータを設定します。
（2）asize < size　　：　データの設定は行いません。そして、戻り値 = (-2)を返します。
（3）asize > size　　：　size 分だけのデータを設定します。そして、戻り値 = 1　を返します。

２．４．３．12　get_value() - 変数値の取得

登録されている変数 ID の変数値をメソッドの引数に指定された領域に取得します。

【構文】

（1）基本のメソッドの構文は次のようになります。

```
public static  int get_value(UInt32 id, IntPtr value)
public static  int get_value( UInt32 id, int pos, IntPtr value )
public static  int get_value(UInt32 id, IntPtr value, int size)
```

値は、エンジンが管理している変数値領域から取得します。

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B U1	get_value(UInt32 id, ref byte value) or get_value(UInt32 id, ref byte value, int pos) set_value(UInt32 id, byte[] value, int size)	
Boolean	get_value(UInt32 id, ref bool value) get_value(UInt32 id, ref bool value, int pos) get_value(UInt32 id, bool[] value, int size)	
I1	get_value(UInt32 id, ref Int8 value) get_value(UInt32 id, ref Int8 value, int pos) get_value(UInt32 id, Int8[] value, int size)	
U2	get_value(UInt32 id, ref UInt16 value) get_value(UInt32 id, ref UInt16 value, int pos) get_value(UInt32 id, UInt16[] value, int size)	
I2	get_value(UInt32 id, ref Int16 value) get_value(UInt32 id, ref Int16 value, int pos) get_value(UInt32 id, Int16[] value, int size)	
U4	get_value(UInt32 id, ref UInt32 value) get_value(UInt32 id, ref UInt32 value, int pos) get_value(UInt32 id, UInt32[] value, int size)	
I4, L	get_value(UInt32 id, ref Int32 value) get_value(UInt32 id, ref Int32 value, int pos) get_value(UInt32 id, Int32[] value, int size)	
U8	get_value(UInt32 id, ref UInt64 value) get_value(UInt32 id, ref UInt64 value, int pos) get_value(UInt32 id, UInt64[] value, int size)	
I8	get_value(UInt32 id, ref Int64 value) get_value(UInt32 id, ref Int64 value, int pos) get_value(UInt32 id, Int648[] value, int size)	
F4	get_value(UInt32 id, ref float value) get_value(UInt32 id, ref float value, int pos) get_value(UInt32 id, float[] value, int size)	

F8	get_value(UInt32 id, ref double value) get_value(UInt32 id, ref double value, int pos) get_value(UInt32 id, double[] value, int size)	
A	get_value(UInt32 id, ref string value)	string の配列は ｻﾎﾟｰﾄしません。

【引数】

id

　　値を取得したい変数 ID

value

　　変数データまたは配列データの格納領域

size

　　取得したいデータ数

【戻り値】

返却値	意　味
0	取得できた。
1	size が変数の定義サイズより小さかった。
2	size が変数の定義サイズより大きかった。
(-1)	ID が登録されていなかった。または配列の取得位置エラー

【説明】

id で指定された変数の値を value に取得します。

変数値が 1 個の場合は、変数値をそのまま value に取得します。

変数値が数値フォーマットで配列データの場合は、size で指定された数だけ value 配列データを取得します。
ただし、変数値の配列サイズはプロパティの asize と size の値によって、以下の処理を行います。

（1）asize = size　　：　asize 分のデータを取得します。
（2）asize < size　　：　asize 分のデータを取得し、戻り値 = 2 を返します。
（3）asize > size　　：　size 分だけのデータを取得し、戻り値 = 1 を返します。

変数が配列データの場合、value の配列サイズは登録変数データを充分格納できるサイズでなければなりません。

２．４．３．13　get_nominal()－変数規定値の取得

登録されている変数 ID の規定値（nominal）をメソッドの引数に指定された領域に取得します。

【構文】

（1）基本のメソッドは次に構文になります。

```
public static  int get_nominal(UInt32 id, IntPtr value)
```

規定値は、id が登録されている変数情報から取得します。

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B, U1	get_nominal(UInt32 id, ref byte value)	
Boolean	get_nominal(UInt32 id, ref bool value)	
I1	get_nominal(UInt32 id, ref Int8 value)	
U2	get_nominal(UInt32 id, ref UInt16 value)	
I2	get_nominal(UInt32 id, ref Int16 value)	
U4	get_nominal(UInt32 id, ref UInt32 value)	
I4, L	get_nominal(UInt32 id, ref Int32 value)	
U8	get_nominal(UInt32 id, ref UInt64 value)	
I8	get_nominal(UInt32 id, ref Int64 value)	
F4	get_nominal(UInt32 id, ref float value)	
F8	get_nominal(UInt32 id, ref double value)	
A	get_nominal(UInt32 id, ref string value)	

【引数】

id

規定値を取得したい変数 ID

value

変数規定値の格納領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	ID が登録されていなかった。

【説明】

id で指定された変数の規定値を引数 value に取得します。

２．４．３．14　get_min() - 変数最小値の取得

登録されている変数 ID の最小値（min）をメソッドの引数に指定された領域に取得します。

【構文】

（1）基本のメソッドは次に構文になります。

```
public static  int get_min(UInt32 id, IntPtr value)
```

最小値は、id が登録されている変数情報から取得します。

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B, U1	get_min(UInt32 id, ref byte value)	
Boolean	get_min(UInt32 id, ref bool value)	
I1	get_min(UInt32 id, ref Int8 value)	
U2	get_min(UInt32 id, ref UInt16 value)	
I2	get_min(UInt32 id, ref Int16 value)	
U4	get_min(UInt32 id, ref UInt32 value)	
I4, L	get_min(UInt32 id, ref Int32 value)	
U8	get_min(UInt32 id, ref UInt64 value)	
I8	get_min(UInt32 id, ref Int64 value)	
F4	get_min(UInt32 id, ref float value)	
F8	get_min(UInt32 id, ref double value)	
A	get_min(UInt32 id, ref string value)	

【引数】

id

　　最小値を取得したい変数 ID

value

　　変数最小値の格納領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	ID が登録されていなかった。

【説明】

id で指定された変数の最小値を引数 value に取得します。

２．４．３．15　get_max() - 変数最大値の取得

登録されている変数 ID の最大値（max）をメソッドの引数に指定された領域に取得します。

【構文】

（1）基本のメソッドは次に構文になります。

```
public static int get_max(UInt32 id, IntPtr value)
```

最大値は、id が登録されている変数情報から取得します。

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B, U1	get_max(UInt32 id, ref byte value)	
Boolean	get_max(UInt32 id, ref bool value)	
I1	get_max(UInt32 id, ref Int8 value)	
U2	get_max(UInt32 id, ref UInt16 value)	
I2	get_max(UInt32 id, ref Int16 value)	
U4	get_max(UInt32 id, ref UInt32 value)	
I4, L	get_max(UInt32 id, ref Int32 value)	
U8	get_max(UInt32 id, ref UInt64 value)	
I8	get_max(UInt32 id, ref Int64 value)	
F4	get_max(UInt32 id, ref float value)	
F8	get_max(UInt32 id, ref double value)	
A	get_max(UInt32 id, ref string value)	

【引数】

id

　　最大値を取得したい変数 ID

value

　　変数最大値の格納領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	ID が登録されていなかった。

【説明】

id で指定された変数の最大値を引数 value に取得します。

２．４．３．16　resize_V_array() - 変数の配列サイズの変更

指定された ID の変数のデータ配列サイズを変更します。

【構文】

public static int resize_V_array(uint vid, int new_size)

【引数】

vid

　　変数 ID

new_size

　　新しい配列サイズ

【戻り値】

返却値	意　味
0	変更できた。
(-1)	変更できなかった。 (ID が見つからなかった、または format が　A, L であった)

【説明】

vid で指定された数値変数のデータ配列サイズを　new_size に変更します。

変更されると、配列要素は、すべてクリア (=0) されます。

本メソッドの変数のフォーマットは、A, L 以外のものが対象となります。

２．４．３．17　resize_V_Linklist() - L 変数のリンク配列サイズの変更

L-フォーマットの変数のリンク VID 格納リストの配列サイズを変更します。

【構文】

public static int resize_V_Linklist(uint vid, int new_size)

【引数】

vid

　　L-フォーマットの変数 ID

new_size

　　新しいリンク配列サイズ

【戻り値】

返却値	意　味
0	変更できた。
(-1)	変更できなかった。 (ID が見つからなかった、または format が　L 以外であった)

【説明】

vid で指定されたデータ配列の中には、vid にリンクされる他の変数(EC,SV,DV)の ID が格納されます。

vid で指定された数値変数のデータ配列を　new_size のサイズに変更します。
変更されると、配列データの値はすべてクリア (=0) されます。

また、リンク VID を 0 個にしたい場合は、new_size=0 に設定してください。

vid で指定された変数のフォーマットは、L でなければなりません。

２．４．３．18　add_V_Linklist() - の変数のリンクリストに vid を追加

L-フォーマットの変数の VID リンクリストに変数 ID を追加します。

【構文】

public static int add_V_Linklist(uint id, uint link_vid)

【引数】

vid

　　L-フォーマットの変数 ID

link_vid

　　vid 変数にリンクしたい追加の変数 ID

【戻り値】

返却値	意　味
0	設定できた。
(-1)	以下の１つの条件 (1)vid が登録されていなかった。 (2)vid のフォーマットが L ではなかった。 (3)link_vid が登録されていなかった。 (4)リストが満杯であった。

【説明】

vid で指定された L-フォーマット変数の VID リンクリスト配列内に変数 ID を追加します。
追加は、リストの先頭から順に行います。
追加は、配列リストのサイズ分になるまで実行してください。

追加できた場合、戻り値　0 を返却します。
追加できなかった場合は、上で示した戻り値を返却します。

２．４．３．19　set_V_Linklist()- の変数のリンクリストに vid を設定

L-フォーマットの変数の VID リンクリストの指定配列位置にリンク変数 ID を設定します。

【構文】

```
public static int set_V_Linklist(uint vid, uint link_vid, int pos)
```

【引数】

vid

　　L-フォーマットの変数 ID

link_vid

　　vid 変数にリンクしたい変数 ID

pos

　　リンクリスト配列の位置(0,..)

【戻り値】

返却値	意　味
0	設定できた。
(-1)	以下の１つの条件 (1)vid が登録されていなかった。 (2)vid のフォーマットが L ではなかった。 (3)link_vid が登録されていなかった。 (4)pos が配列リスト外の位置であった。

【説明】

vid で指定された L-フォーマット変数のリンクリスト内に変数 ID、link_vid を設定します。
設定する配列位置は pos で指定します。
(link_vid_list[pos] = link_vid)

設定できた場合、戻り値　0 を返却します。

設定できない条件を検出した場合は、上で示した戻り値を返却します。

２．４．３．20　set_V_Linklist_all() - の変数のリンクリストに全 vid を設定

L-フォーマットの変数の VID リンクリストに、まとめてリンク変数 ID を設定します。

【構文】

```
public static int set_V_Linklist_all(uint id, uint[] link_vid_list, int size)
```

【引数】

vid
　　L-フォーマットの変数 ID
link_vid_list
　　vid 変数にリンクしたい変数 ID の配列（size 分の配列）
size
　　設定するリンク VID の数　（ >= 0 ）

【戻り値】

返却値	意　味
0	設定できた。
(-1)	以下の１つの条件 (1)vid が登録されていなかった。 (2)vid のフォーマットが L ではなかった。 (3)link_vid_list 内の vid が登録されていなかった。

【説明】

最初に、vid で指定された変数のリンク VID 配列の要素数を size にリサイズします。
そして、size > 0 の場合は、link_vid_list 配列に含まれるリンク変数 ID を vid のリンク VID 配列領域に設定します。
size = 0 の指定の場合は、vid 変数の配列は空になります。

設定できない条件を検出した場合は、上で示した戻り値を返却します。

２．４．３．21　set_limit_info()－の変数リミット情報の設定

変数 ID を指定してリミット情報を設定します。

【構文】

public static int set_limit_info(UInt32 id, TLIMIT_INFO info)

【引数】

id

　　変数 ID

info

　　リミット情報クラス TLIMIT_INFO のインスタンス

【戻り値】

返却値	意　味
0	設定できた。
(-1)	以下の１つの条件 (1)vid が登録されていなかった。 (2)vid のフォーマットが 数値のフォーマットでなかった。 　　(L, A, J, BOOLEAN,はLimit 監視対象にならない)

【説明】

変数のリミット監視のためのリミット情報を設定します。

リミット情報は、TLIMIT_INFO クラスのインスタンスです。 TLIMIT_INFO クラスについては、別途　3.11 で説明します。

設定できない条件を検出した場合は、上で示した戻り値を返却します。

２．４．３．22　get_limit_info() - の変数リミット情報の取得

変数 ID を指定して変数のリミット情報を取得します。

【構文】

```
public static int get_limit_info(UInt32 id, ref TLIMIT_INFO info)
```

【引数】

id

　　変数 ID

info

　　取得したリミット情報を保存するための TLIMIT_INFO のクラスのインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	以下の１つの条件 (1)vid が登録されていなかった。 (2)リミット情報が無かった。

【説明】

変数のリミット情報を取得します。

２．４．３．23　del_limit info() - の変数リミット情報の取得

変数 ID を指定して変数のリミット情報を削除します。

【構文】

```
public static int del_limit_info(UInt32 id)
```

【引数】

id

　　変数 ID

【戻り値】

返却値	意　味
0	削除できた。(常時)

【説明】

id で指定された変数のリミット情報を削除します。

返却値は、変数 ID が未登録、リミット情報が無いなどの場合でも 0 を返します。

３．EC, SV, DVVAL - 装置変数関連情報保存クラス

装置変数 EC,SV,DV の個別情報は TV_INFO クラスのインスタンスの中に保存されます。

TV_INFO クラスには、ID, 値などの変数情報を保存する プロパティが存在します。
変数値の保存には TV_VALUE クラスを使用します。

以下、装置変数に関連する情報を保存する TV_INFO クラスについて説明します。

３．１　TV_INFO - 装置変数情報保存クラス

EC, SV, DV の装置変数情報を保存するクラスです。

３．１．１　コンストラクタ

TV_INFO クラスのインスタンスを生成します。

インスタンス v_info を生成する例です。

（1）空のインスタンスを生成

```
TV_INFO v_info = new TV_INFO();
```

（2）V_123 の変数 ID のインスタンスを生成します。

```
UInt32 V_123 = 100;
TV_INFO v_info = new TV_INFO(V_123);
```

これで、V_123 の変数情報が v_info に設定されることになります。

（注）V_123 の変数が登録されていない場合は空のインスタンスを生成します。

３．１．２　プロパティ

下表に示すプロパティを所有しています。

	プロパティ名	説明
1	public UInt32 vid	変数 ID です。
2	public string name	変数名です。
3	public int format	変数値のフォーマットです。 B, A, I1, I2...などです。
4	public int asize	変数値のサイズです。 ①format が数値データの場合はデータサイズです。 配列サイズになります。 ②format=A の場合はﾊﾞｲﾄ数です。 ③format=L の場合は当該変数にリンクされる変数 ID 数 になります。
5	public IntPtr value	変数データの値です。 format によって、asize 分のメモリの中に値が保存されます。
6	public IntPtr nominal	規定値です。装置変数定義ファイルで定義されます。
7	public IntPtr min	value プロパティの取りうる最小値になります。 （format-l, A, J には適用されません）
8	public IntPtr max	min と同様に value の最大値になります。
9	public string units	物理単位名です。
10	public UInt32[] link_vid_list	format=L の変数で、L にリンクされている変数 ID の配列 リストです。
11	public string[] link_vname_list	link_vid_list 配列内に設定されている変数の名前リストです。
12	public int link_v_size	link_vid_list に準備された変数 ID リストの配列サイズです。
13	public int link_v_count	link_vid_list の中に設定されている変数 ID 数です。
14	public UInt32[] ceid_list	変数値 value の値が変化したタイミングに送信する S6F11 の ID の配列です。
15	public string[] ce_name_list	ceid_list 配列に設定されている CEID の名前リストです。
16	public int ce_count	ce_list[]に設定されている CEID です。
17	public TLIMIT_INFO limit	変数のリミット情報です。 L を除く format についてのみ使用されます。

３．１．３　メソッド

変数情報保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int set_value()	value の値をインスタンス内に設定します。 (formmat 別に準備されています)
4	public int get_value()	当該変数インスタンスの値を取得します。 （formmat 別に準備されてます)
5	public int copy()	変数インスタンスをコピーします。 2 種類あります。

３．１．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．１．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）

３．１．３．３　set_value() - 変数値の設定

メソッドの引数で与えられたデータを変数 ID が指定する変数値に設定します。

【構文】

（1）基本のメソッドの構文は次のようになります。

```
public static int set_value(IntPtr value)
```

設定値は IntPtr が指す領域から取り出し設定します。

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B U1	public int set_value(byte value) public int set_value(byte[] value, int size)	
Boolean	public int set_value(bool value) public int set_value(bool[] value, int size)	
I1	public int set_value(Int8 value) public int set_value(Int8[] value, int size)	
U2	public int set_value(UInt16 value) public int set_value(UInt16[] value, int size)	
I2	public int set_value(Int16 value) public int set_value(Int16[] value, int size)	
U4	public int set_value(UInt32 value) public int set_value(UInt32[] value, int size)	
I4,L	public int set_value(Int32 value) public int set_value(Int32[] value, int size)	
U8	public int set_value(UInt64 value) public int set_value(UInt64[] value, int size)	
I8	public int set_value(Int64 value) public int set_value(Int64[] value, int size)	
F4	public int set_value(float value) public int set_value(float[] value, int size)	
F8	public int set_value(double value) public int set_value(double[] value, int size)	
A	public int set_value(string value)	string の配列はｻﾎﾟｰﾄしません。

【引数】

value

設定した変数データまたは配列データ

【戻り値】

返却値	意　味
0	設定できた。
1	変数の定義サイズより size が小さかった。
(-1)	pos 位置エラーであった。
(-2)	size が変数の定義サイズより大きかった。

【説明】

value で与えられたデータを当該インスタンスの変数値に設定します。

変数値が 1 個の場合は、value の値をそのまま設定します。

変数値が配列データの場合は、size で指定された数だけの value 配列データを設定します。
ただし、変数値の配列サイズ-を asize とした場合、asize と size の値によって、以下の処理を行います。

（1）asize = size　　：　asize 分のデータを設定します。
（2）asize < size　　：　データの設定は行いません。そして、戻り値 = (-2)を返します。
（3）asize > size　　：　size 分だけのデータを設定します。そして、戻り値 = 1　を返します。

３．１．３．４　get_value() - 変数値の取得

当該インスタンスの変数値をメソッドの引数に指定されたメモリに取得します。

【構文】

（1）基本のメソッドの構文は次のようになります。

```
public  int get_value( IntPtr value)
public  int get_value(IntPtr value, int size)
```

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B U1	public int get_value(ref byte value) public int get_value(byte[] value, int size)	
Boolean	public int get_value(ref bool value) public int get_value(bool[] value, int size)	
I1	public int get_value(ref Int8 value) public int get_value(sbyte[] value, int size)	
U2	public int get_value(ref UInt16 value) public int get_value(UInt16[] value, int size)	
I2	public int get_value(ref Int16 value) public int get_value(Int16[] value, int size)	
U4	public int get_value(ref UInt32 value) public int get_value(UInt32[] value, int size)	
I4, L	public int get_value(ref Int32 value) public int get_value(Int32[] value, int size)	
U8	public int get_value(ref UInt64 value) public int get_value(UInt64[] value, int size)	
I8	public int get_value(ref Int64 value) public int get_value(Int64[] value, int size)	
F4	public int get_value(ref float value) public int get_value(float[] value, int size)	
F8	public int get_value(ref double value) public int get_value(double[] value, int size)	
A	public int get_value(ref string value)	string の配列は ｻﾎﾟｰﾄしません。

【引数】

value

変数データまたは配列データの格納メモリ

size

取得するデータ数

【戻り値】

返却値	意　味
0	取得できた。
1	size が変数の定義サイズより小さかった。
2	size が変数の定義サイズより大きかった。
(-1)	ID が登録されていなかった。または配列の取得位置エラー

【説明】

id で指定された変数値を value に取得します。

変数値が 1 個の場合は、変数値をそのまま value に取得します。

変数値が数値データで、しかも配列データの場合は、size で指定された数だけ value 配列データを取得します。
ただし、変数値の配列サイズ-を asize とした場合、asize と size の値によって、以下の処理を行います。

（1）asize = size　　：　asize 分のデータを取得します。
（2）asize < size　　：　asize 分のデータを取得し、戻り値 = 2 を返します。
（3）asize > size　　：　size 分だけのデータを取得し、戻り値 = 1 を返します。

変数が配列データの場合、value の配列サイズは登録変数データを充分格納できるサイズでなければなりません。

３．１．３．５　copy() - TV_INFO のコピー

TV_INFO クラスのインスタンスを別の TV_INFO のインスタンスにコピーします。
２種類の構文があります。

（1）コピー先が当該インスタンスの場合
（2）コピー先もコピー元も当該インスタンスでない場合

【構文】

```
public int copy( TV_INFO src_info)
public static int copy(ref TV_INFO dst_info, TV_INFO src_info)
```

【引数】

dst_info
　　コピー先インスタンス
src_info
　　コピー元インスタンス

【戻り値】

返却値	意　味
0	コピーできた。
(-1)	コピーできなかった。

【説明】

TV_INFO クラスのインスタンスを別の TV_INFO のインスタンスにコピーします。

コピーする主なプロパティは以下のとおりです。

```
vid, name, format, asize, value, units, nominal, min, max,
ink_v_size, link_v_count,
ce_list, ce_count, ce_name_list
limit
```

引数として dst_info が指定されない場合は、当該インスタンスがコピー先になります。
引数 dst_info が指定された場合は、static のメソッドです。

３．２　TVID_LIST - 変数 ID 保存配列リストクラス

変数 ID を保存する配列リストです。

３．２．１　コンストラクタ

省略

３．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	配列に保存されている ID 数
2	public UInt32[] id_list	ID が保存される配列リスト

３．２．３　メソッド

変数 ID 保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void clear()	すべてのプロパティの値をクリアします。
2	public void add()	配列リストに変数 ID を 1 個追加します。

３．２．３．１　clear() - インスタンスのクリア

プロパティの内容をクリアします。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

プロパティの count = 0 にし、vid_list リストを空にします。

３．２．３．２　add() - 変数 ID を追加

変数 ID 配列リストに変数 ID を１個追加します。

【構文】

public int add(UInt32 vid)

【引数】

vid

追加する変数 ID

【戻り値】

追加した後のプロパティ count の値を返却します。

【説明】

変数 ID 配列リスト、プロパティ vid_list に引数 vid を追加し、count +1 します。

追加後、配列リストに保存されている変数 ID 数を返却します。

３．３　TVID_VAL_LIST - 変数 ID と値保存配列リスト

変数 ID を保存する配列リストです。
本クラスは、“S2F15　EC データ送信”の処理に使用します。

３．３．１　コンストラクタ

TDV_VAL_LIST クラスのインスタンスの生成を行います。

３．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	配列に保存されている ID 数
2	public TVID_VAL[] list	ID と値が保存される配列リスト

３．３．３　メソッド

変数 ID 保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int add ()	配列リストに変数 ID と値を追加します。

３．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）

３．３．３．３　add()- 変数 ID と値を追加

変数 ID 配列リストに変数 ID を１個追加します。

【構文】

public int add(TV_INFO vinfo)
public int add(TVID_VAL info)

【引数】

vinfo
　　装置変数情報クラスのインスタンス
info
　　変数データ情報

【戻り値】

追加した後のプロパティ count の値を返却します。

【説明】

変数 ID と値の配列リスト、TVID_VAL[] list に引数 で与えられたインスタンスの情報を追加し、count +1 します。

引数が **TV_INFO** vinfo の場合は、変数情報クラス vinfo の内容を list に追加設定します。
引数が **TVID_VAL** info の場合は、変数 ID、データ情報 info の内容を list に追加します。

追加後、配列リストに保存されている変数 ID 数を返却します。

３．４　TV_VALUE クラス

１個の変数値情報を保存するためのクラスです。
プロパティには format-L にリンクされる VID リストも含みます。

３．４．１　コンストラクタ

TV_VALUE クラスのインスタンスの生成を行います。

３．４．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public UInt32 vid	変数 ID
2	public int format	変数データのフォーマット
3	public int size	変数データのサイズ
4	public IntPtr value	変数値保存メモリ
5	public TV_VALUE_LIST V_L_list	L-フォーマット。リンク変数値情報保存リスト コンストラクタでは null が設定されます。

３．４．３　メソッド

変数 ID 保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int add_link_TV_VALUE()	V_L_list にリンク変数値情報を１個リンクリストに追加します。
4	public int set_link_TV_VALUE()	V_L_list に変数情報リストを設定します。 （複数の TV_VALUE のインスタンスを設定します）
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．４．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．４．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）も開放します。）

３．４．３．３　add_link_TV_VALUE()－リンクリストに変数値情報を追加

変数リンクリストにリンクしたい変数値情報を VID リンクリストに追加します。

【構文】

public int add_link_TV_VALUE(TV_VALUE val_info)

【引数】

val_vinfo

追加したい装置変数情報クラスのインスタンス

［戻り値】

返却値	意　味
＞ 0	追加できた。追加した後の count の値を返却
(-1)	①当該クラスの format が L でなかった。 ②val_info が null であった。

【説明】

引数 val_info で与えられたインスタンスの情報をプロパティ V_L_list に追加します。そして、プロパティ count を　+1 します。

追加後、配列リストに保存されている変数 ID 数を返却します。

追加できなかった場合は (-1)を返却します。

３．４．３．４　set_link_TV_VALUE()－リンク配列リストの設定

変数 ID リンクリストにリンクしたい複数の変数値情報を設定します。

【構文】

public int set_link_TV_VALUE(TV_VALUE_LIST list)

【引数】

list

設定したい装置変数情報リストのインスタンス　(TV_VALUE_LIST クラスは、3.5 を参照)

［戻り値】

返却値	意　味
= 0	設定できた。
(-1)	①当該クラスの format が L でなかった。 ②list が null であった。

【説明】

変数 ID と変数値の配列リストである V_L_list に引数 list で与えられた TV_VALUE_LIST クラスのインスタンスの情報を設定します。

設定できた場合は　0 を返却します。

設定できなかったばあは(-1)を返却します。

３．４．３．５　copy() - インスタンスのコピー

TV_VALUE クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TV_VALUE dst, TV_VALUE src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

TV_VALUE インスタンス src を dst にコピーします。

３．５　TV_VALUE_LIST - 変数値保存配列リストクラス

TV_VALUE クラスの配列リストのクラスです。
format-L の変数にリンクされる変数の配列リストです。

３．５．１　コンストラクタ

クラスのインスタンスの生成を行います。

３．５．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	保存されている TV_VALUE のインスタンス数
2	public TV_VALUE[] list	TV_VALUE の配列リスト

３．５．３　メソッド

変数 ID 保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	プロパティの値をクリアします。
3	public int add()	１個 TV_VALUE のインスタンスを配列リストに追加します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．５．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．５．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）も開放します。）

３．５．３．３　add() - リンク配列に TV_VALUE を追加

配列リスト list に TV_VALUE のインスタンスを１個追加します。

【構文】

public int add(TV_VALUE vinfo)

【引数】

v_vinfo

追加したい TV_VALUE クラスのインスタンス

[戻り値】

返却値	意　味
＞ 0	追加できた。追加した後の count の値を返却
(-1)	v_info が null であった。

【説明】

TV_VALUE 配列リストである list に引数 v_info で与えられたインスタンスの情報を追加します。そして、プロパティ count を　+1 します。
追加後、配列リストに保存されている TV_VALUE のインスタンス数を返却します。
追加できなかった場合は (-1) を返却します。

３．５．３．４　copy() - インスタンスのコピー

TV_VALUE_LIST クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TV_VALUE_LIST dst, TV_VALUE_LIST src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

TV_VALUE_LIST クラスのインスタンス src を dst にコピーします。

３．６　TSV_NAME_LIST - SV名と物理単位保存配列リストクラス

SV変数のID、名前と物理単位を保存するための配列リストです。S1F11の応答メッセージS1F12メッセージのデコード／エンコード処理時に使用します。

３．６．１　コンストラクタ

クラスのインスタンスの生成を行います。

３．６．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	配列リストに保存されているTSV_NAMEのインスタンス数
2	public TSV_NAME [] name_list	名前と物理単位保存用の配列リスト

３．６．３　メソッド

変数ID保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	プロパティの値をクリアします。
3	public int add()	１個SV変数に対する変数名と物理単位名をname_list配列リストに追加します。

３．６．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
Dispose()の後、このインスタンスを使用することはできません。

３．６．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．６．３．３　add() - 配列に名前と物理単位名を追加

配列リスト name_list に 1 個の SV の名前と物理単位名を追加します。
また、引数として TSV_NAME クラスのインスタンスを指定して追加することができます。

【構文】

public int add(string name, string units)
public int add(TSV_NAME info)

【引数】

name
　　SV 変数名
units
　　物理単位名
info
　　名前，物理単位名保存クラスのインスタンス

［戻り値】

返却値	意　味
＞ 0	追加できた。追加した後の count の値を返却
(-1)	info が null であった。

【説明】

引数に name，units を指定する場合と、TSV_NAME クラスのインスタンスを指定する場合があります。
TSV_NAME のクラスにはプロパティとして SV 名と物理単位名が含まれます。

追加した後、count+1 にして、その値を返却します。
追加できなかった場合は (-1)を返却します。

３．７　TSV_NAME - SV 名と物理単位情報保存クラス

SV 変数の名前と物理単位保存用クラスです。S1F11 の応答メッセージ S1F12 メッセージのデコード／エンコード処理時に使用します。

３．７．１　コンストラクタ

クラスのインスタンスの生成を行います。

３．７．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public UInt32 svid	SVID
2	public string name	SV 変数名
3	public string units	物理単位名

３．７．３　メソッド

変数 ID 保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	プロパティの内容をすべてクリアします。
3	public void set_id()	SVID を設定します。
4	public void set()	変数名、物理単位名を設定します。

３．７．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
Dispose()の後、このインスタンスを使用することはできません。

３．７．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．７．３．３　set_id() - 変数 ID の設定

SVID を設定します。

【構文】

public void set_id(UInt32 vid)

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイ svid に引数の vid を設定します。。

５．７．３．４　set() - 変数名、物理単位の設定

インスタンスの中に変数名と物理単位を設定します。

【構文】

public int set(UInt32 id, TV_INFO info)

【引数】

name

SV 変数名

units

SV の物理単位

【戻り値】

なし。

【説明】

引数で与えられた変数名と物理単位をプロパティに設定します。

３．８　TEC_NAME_LIST - EC 変数情報保存配列リストクラス

EC 変数の ID、名前、物理単位名等の情報を保存する配列リストです。S2F29 の応答メッセージ S2F30 メッセージのデコード／エンコード処理時に使用します。

３．８．１　コンストラクタ

クラスのインスタンスの生成を行います。

３．８．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	配列リストに保存されている TEC_NAME のインスタンス数
2	public TEC_NAME [] name_list	名前と物理単位保存用の配列リスト

３．８．３　メソッド

変数 ID 保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	プロパティの値をクリアします。システムの資源（メモリ）の開放も行います。
3	public int add()	１個 EC 変数に対する変数名と物理単位名などの情報を name_list 配列リストに追加します。

３．８．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
Dispose()の後、このインスタンスを使用することはできません。

３．８．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）

３．８．３．３　add()－配列に EC 情報を追加

配列リスト name_list に 1 個の EC の名前と物理単位名などの情報を追加します。
また、引数として TEC_NAME クラスのインスタンスを指定して追加することができます。

【構文】

public int add(TEC_NAME info)

【引数】

info

EC 変数の名前, 物理単位名等の情報保存クラスのインスタンス

［戻り値】

返却値	意　味
＞ 0	追加できた。追加した後の count の値を返却
(-1)	info が null であった。

【説明】

引数に TEC_NAME クラスのインスタンスを指定します。
TEC_NAME クラスについては、3.9 を参照ください。

追加した後、count+1 にして、その値を返却します。
追加できなかった場合は (-1)を返却します。

３．９　TEC_NAME - EC 変数情報保存クラス

１個の EC 変数の名前と物理単位当の保存用クラスです。S2F29 の応答メッセージ S2F30 メッセージのデコード／エンコード処理時に使用します。

３．９．１　コンストラクタ

クラスのインスタンスの生成を行います。

３．９．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public UInt32 ecid	ECID
2	public string name	EC 変数名
3	public int format	データ format
4	public int asize	データサイズ
5	public IntPtr ecmax	最大値
6	public IntPtr ecmin	最小値
7	public IntPtr nominal	規定値
8	public string units	物理単位名

３．９．３　メソッド

変数 ID 保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	clear()	プロパティの値をクリアします。
3	set_name_info()	EC 情報をプロパテイに設定します。

３．９．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
Dispose()の後、このインスタンスを使用することはできません。

３．９．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．９．３．３　set_name_info() - EC 情報の設定

引数に指定された情報を当該インスタンスのプロパテイに設定します。

【構文】

```
public void set_name_info(string name, int format, int asize,
                IntPtr ecmin, IntPtr ecmax, IntPtr nominal, string units)
```

【引数】

name	EC 変数名
format	データフォーマット
asize	データサイズ
ecmin	最小値
ecmax	最大値
nominal	規定値
units	物理単位名

【戻り値】

なし。

【説明】

引数で与えられた情報をそれぞれのプロパティに設定します。

３．10　class_V_ope - 変数処理関連クラス

装置変数データ値のオペレーション処理を行うためのクラスです。
本クラスのメソッドは全て static 修飾子付きのメソッドです。
従って、メソッドの呼出し方法は、class_V_ope.method()のように行います。

３．10．１　コンストラクタ

なし。

３．10．２　プロパティ

なし。

３．10．３　メソッド

以下、説明するメソッドの変数情報クラス **TV_INFO クラスのインスタンスが１番目の引数**になります。
そして、そのインスタンスに関する処理になります。

本クラスで提供しりメソッドは下表のとおりです。

	メソッド名	説明
1	public static int set_value()	変数値を設定します。
2	public static int get_value()	変数値を取得します。
3	public static int get_nominal_value()	規定値を取得します。
4	public static int get_min_value()	最小値を取得します。
5	public static int get_max_value()	最大値を取得します。
6	public static int resize_V_array()	変数値の配列サイズを変更します。
7	public static int resize_V_Linklist()	format = L のリンク変数 ID の配列を変更します。
8	public static int add_V_Linklist()	format = L のリンク変数 ID の配列に変数 ID を追加します。
9	public static int set_V_Linklist()	format = L のリンク変数 ID の配列位置を指定して変数 ID を設定します。
10	public static int set_V_Linklist_all()	変数の VID リンクリストにまとめて変数 ID を設定します。

３．10．３．１ set_value() - 変数値の設定

TV_INFO クラスのインスタンスに引数で指定された値を設定します。

【構文】

public static int set(TV_INFO info, IntPtr value)
public static int set(TV_INFO info, string value)

【引数】

info
　　設定したい変数情報のインスタンス
value
　　設定したい値

【戻り値】

返却値	意　味
0	設定できた。
-1	①info が null であった。 ②最小、最大値の範囲を超えていた。

【説明】

info の変数値の内容を info のインスタンスに設定します。

format が L, A, BOOLEAN 以外の場合、最小値(min)、最大値(max)の設定範囲を超えていた場合は、設定されません。

３．10．３．２　get_value() - 変数値の取得

変数情報保存クラスのインスタンスの変数値を、引数に指定された領域に取得します。

【構文】

（1）基本のメソッドの構文は次のようになります。

```
public static int get_value(TV_INFO info, IntPtr value)
public static int get_value(TV_INFO info, IntPtr value, int pos )
public static int get_value(TV_INFO info, IntPtr value, int size)
```

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B U1	get_value(TV_INFO info, ref byte value) or get_value(TV_INFO info, ref byte value, int pos) set_value(TV_INFO info, byte[] value, int size)	
Boolean	get_value(TV_INFO info, ref bool value) get_value(TV_INFO info, ref bool value, int pos) get_value(TV_INFO info, bool[] value, int size)	
I1	get_value(TV_INFO info, ref Int8 value) get_value(TV_INFO info, ref Int8 value, int pos) get_value(TV_INFO info, Int8[] value, int size)	
U2	get_value(TV_INFO info, ref UInt16 value) get_value(TV_INFO info, ref UInt16 value, int pos) get_value(TV_INFO info, UInt16[] value, int size)	
I2	get_value(TV_INFO info, ref Int16 value) get_value(TV_INFO info, ref Int16 value, int pos) get_value(TV_INFO info, Int16[] value, int size)	
U4	get_value(TV_INFO info, ref UInt32 value) get_value(TV_INFO info, ref UInt32 value, int pos) get_value(TV_INFO info, UInt32[] value, int size)	
I4	get_value(TV_INFO info, ref Int32 value) get_value(TV_INFO info, ref Int32 value, int pos) get_value(TV_INFO info, Int32[] value, int size)	
U8	get_value(TV_INFO info, ref UInt64 value) get_value(TV_INFO info, ref UInt64 value, int pos) get_value(TV_INFO info, UInt64[] value, int size)	
I8	get_value(TV_INFO info, ref Int64 value) get_value(TV_INFO info, ref Int64 value, int pos) get_value(TV_INFO info, Int648[] value, int size)	
F4	get_value(TV_INFO info, ref float value) get_value(TV_INFO info, ref float value, int pos) get_value(TV_INFO info, float[] value, int size)	

F8	get_value(TV_INFO info, ref double value) get_value(TV_INFO info, ref double value, int pos) get_value(TV_INFO info, double[] value, int size)	
A	get_value(TV_INFO info, ref string value)	string の配列は ｻﾎﾟｰﾄしません。

【引数】

info

変数情報クラスのインスタンス

value

変数データまたは配列データの格納領域

size

取得したいデータ数

【戻り値】

返却値	意　味
0	取得できた。
1	size が、変数の定義サイズより小さかった。
(-1)	info の値が null であった。
(-2)	size が定義サイズより大きかった。

【説明】

変数情報保存クラスのインスタンスの変数値を value に取得します。

変数値が 1 個の場合は、変数値をそのまま value に取得します。

変数値が数値データで、しかも配列データの場合は、size で指定された数だけ value 配列データを取得します。
ただし、変数値の配列サイズ-を asize とした場合、asize と size の値によって、以下の処理を行います。

（1）asize = size　　：　asize 分のデータを取得します。
（2）asize < size　　：　データの取得は行いません。そして、戻り値 = (-2)を返します。
（3）asize > size　　：　size 分だけのデータを取得します。そして、戻り値 = 1　を返します。

変数が配列データの場合、value の配列サイズは登録変数データを充分格納できるサイズでなければなりません。

３．10．３．３　get_nominal_value() - 変数規定値の取得

変数情報保存クラスのインスタンスの規定値（nominal）を、引数に指定された領域に取得します。

【構文】

（1）基本のメソッドは次に構文になります。

```
public static int get_nominal_value(TV_INFO info, IntPtr value)
```

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B, U1	get_nominal_value(TV_INFO info, ref byte value)	
Boolean	get_nominal_value(TV_INFO info, ref bool value)	
I1	get_nominal_value(TV_INFO info, ref Int8 value)	
U2	get_nominal_value(TV_INFO info, ref UInt16 value)	
I2	get_nominal_value(TV_INFO info, ref Int16 value)	
U4	get_nominal_value(TV_INFO info, ref UInt32 value)	
I4 L	get_nominal_value(TV_INFO info, ref Int32 value)	
U8	get_nominal_value(TV_INFO info, ref UInt64 value)	
I8	get_nominal_value(TV_INFO info, ref Int64 value)	
F4	get_nominal_value(TV_INFO info, ref float value)	
F8	get_nominal_value(TV_INFO info, ref double value)	
A	get_nominal_value(TV_INFO info, ref string value)	

【引数】

info

変数情報クラスのインスタンス

value

変数規定値の格納領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	info の値が null であった。

【説明】

変数情報保存クラスのインスタンスの規定値を value に取得します。

３．10．３．４　get_min() - 変数最小値の取得

変数情報保存クラスのインスタンスの最小値（min）を引数に指定された領域に取得します。

【構文】

（1）基本のメソッドは次に構文になります。

```
public static int get_min(TV_INFO info, IntPtr value)
```

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B, U1	get_min(TV_INFO info, ref byte value)	
Boolean	get_min(TV_INFO info, ref bool value)	
I1	get_min(TV_INFO info, ref Int8 value)	
U2	get_min(TV_INFO info, ref UInt16 value)	
I2	get_min(TV_INFO info, ref Int16 value)	
U4	get_min(TV_INFO info, ref UInt32 value)	
I4	get_min(TV_INFO info, ref Int32 value)	
U8	get_min(TV_INFO info, ref UInt64 value)	
I8	get_min(TV_INFO info, ref Int64 value)	
F4	get_min(TV_INFO info, ref float value)	
F8	get_min(TV_INFO info, ref double value)	
A	get_min(TV_INFO info, ref string value)	

【引数】

info

変数情報保存クラスのインスタンス

value

変数最小値の格納領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	info の値が null であった。

【説明】

変数情報保存クラスのインスタンスの最小値を value に取得します。

３．10．３．５　get_max() - 変数最大値の取得

変数情報保存クラスのインスタンスの最大値（max）を引数に指定された領域に取得します。

【構文】

（1）基本のメソッドは次に構文になります。

```
public static int get_max(TV_INFO info, IntPtr value)
```

（2）変数のフォーマット別に以下のメソッドがあります。

ﾌｫｰﾏｯﾄ	メソッド	備考
B, U1	get_max(TV_INFO info, ref byte value)	
Boolean	get_max(TV_INFO info, ref bool value)	
I1	get_max(TV_INFO info, ref Int8 value)	
U2	get_max(TV_INFO info, ref UInt16 value)	
I2	get_max(TV_INFO info, ref Int16 value)	
U4	get_max(TV_INFO info, ref UInt32 value)	
I4	get_max(TV_INFO info, ref Int32 value)	
U8	get_max(TV_INFO info, ref UInt64 value)	
I8	get_max(TV_INFO info, ref Int64 value)	
F4	get_max(TV_INFO info, ref float value)	
F8	get_max(TV_INFO info, ref double value)	
A	get_max(TV_INFO info, ref string value)	

【引数】

info

変数情報保存クラスのインスタンス

value

変数最大値の格納領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	info の値が null であった。

【説明】

変数情報保存クラスのインスタンスの最大値を value に取得します。

３．10．３．６　resize_V_array() - 変数の配列サイズの変更

指定された変数情報保存クラスのインスタンスの変数値配列の要素数を変更します。

【構文】

public static int resize_V_array(TV_INFO vinfo, int new_size)

【引数】

vinfo

　　変数情報保存クラスのインスタンス

new_size

　　新しい配列サイズ

【戻り値】

返却値	意　味
0	変更できた。
(-1)	変更できなかった。 (ID が見つからなかった、または format が　A, L であった)

【説明】

vinfo で指定された数値変数のデータ配列サイズを　new_size のサイズに変更します。

変更されると、配列要素の値はすべてクリア (=0) されます。

本メソッドの変数のフォーマットは、A, L 以外のものが対象となります。

３．10．３．７　resize_V_Linklist()－Ｌ変数のリンク配列サイズの変更

L-フォーマットの変数のリンク VID 格納リストの配列サイズを変更します。

【構文】

public static int resize_V_Linklist(TV_INFO vinfo, int new_size)

【引数】

vinfo

　　変数情報保存クラスのインスタンス

new_size

　　新しいリンク配列サイズ

【戻り値】

返却値	意　味
0	変更できた。
(-1)	変更できなかった。 (ID が見つからなかった、または format が　L 以外であった)

【説明】

vinfo で指定されたデータ配列の中には、vid にリンクされる他の変数(EC,SV,DV)の ID が格納されます。

vinfo で指定された数値変数のデータ配列を　new_size のサイズに変更します。
変更後、配列データの値はすべてクリア (=0) されます。

また、リンク VID を 0 個にしたい場合は、new_size=0 に設定してください。

vinfo で指定された変数のフォーマットは、L でなければなりません。

３．10．３．８　add_V_Linklist() - の変数のリンクリストに vid を追加

L-フォーマットの変数の VID リンクリストに順番に変数 ID を追加します。

【構文】

public static int add_V_Linklist(TV_INFO vinfo, uint link_vid)

【引数】

vinfo

　　変数情報保存クラスのインスタンス

link_vid

　　vid 変数にリンクしたい追加の変数 ID

【戻り値】

返却値	意　味
0	設定できた。
(-1)	以下の１つの条件 (1)vinfo の変数 ID が登録されていなかった。 (2) vinfo の変数 ID のフォーマットが L ではなかった。 (3)link_vid が登録されていなかった。 (4)リストが満杯であった。

【説明】

vinfo で指定された L-フォーマット変数の VID リンクリスト配列内に 1 個の変数 ID を追加します。
追加は、リストの先頭から順に行います。
追加は、配列リストのサイズ分になるまで実行してください。

追加できた場合、戻り値　0 を返却します。
追加できなかった場合は、上で示した戻り値を返却します。

３．10．３．９　set_V_Linklist()- の変数のリンクリストに vid を設定

L-フォーマットの変数の VID リンクリストの指定配列位置にリンク変数 ID を設定します。

【構文】

```
public static int set_V_Linklist(TV_INFO vinfo, uint link_vid, int pos)
```

【引数】

vinfo
　　変数情報保存クラスのインスタンス
link_vid
　　vid 変数にリンクしたい変数 ID
pos
　　リンクリスト配列の位置(0,..)

【戻り値】

返却値	意　味
0	設定できた。
(-1)	以下の１つの条件 (1) vinfo の変数 ID が登録されていなかった。 (2) vinfo の変数 ID のフォーマットが L ではなかった。 (3)link_vid が登録されていなかった。 (4)pos が配列リスト外の位置であった。

【説明】

vinfo で指定された L-フォーマット変数のリンクリスト配列内に変数 ID、link_vid を設定します。
設定する配列の位置は pos で指定します。

設定できる条件であれば設定できた場合、戻り値　0 を返却します。

設定できない条件を検出した場合は、上で示した戻り値を返却します。

３．10．３．10　set_V_Linklist_all() - の変数のリンクリストに全 vid を設定

L-フォーマットの変数の VID リンクリストにまとめてリンク変数 ID を設定します。

【構文】

public static int set_V_Linklist_all(TV_INFO vinfo, uint[] link_vid_list, int size)

【引数】

vinfo

　　変数情報保存クラスのインスタンス

link_vid_list

　　vid 変数にリンクしたい変数 ID の配列（size 分の配列）

size

　　設定するリンク VID の数　（ >= 0 ）

【戻り値】

返却値	意　味
0	設定できた。
(-1)	以下の１つの条件 (1)vinfo の変数 ID が登録されていなかった。 (2)vinfo の変数 ID のフォーマットが L ではなかった。 (3)link_vid_list 内の vid が登録されていなかった。

【説明】

vinfo で指定された変数のリンク VID 配列の要素数を size に変更します。
そして、size > 0 の場合は、link_vid_list 配列に含まれるリンク変数 ID を vid のリンク VID 配列領域に設定します。
size = 0 の指定の場合は、vid 変数の配列は空になります。

設定できない条件を検出した場合は、上で示した戻り値を返却します。

３．11　TLIMIT_INFO - 装置変数リミット情報保存クラス

装置変数のリミット情報を保存クラスです。

３．11．１　コンストラクタ

TLIMIT_INFO クラスのインスタンスを生成します。

例 ： V_123 の変数 ID のインスタンスを生成します。

```
UInt32 V_123 = 100;
TLIMIT_INFO V_123 = new TLIMIT_INFO();
```

３．11．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明
1	public UInt32 vid	変数 ID です。(EC, SV, or DV)
2	public int format	変数値のフォーマットです。 B, A, I1, I1...など
3	public int asize	変数値のサイズ
4	lmt_state	リミット監視の状態です。 不定、確定、状態遷移
5	public int c_limitid	(内部処理用)
6	public int lmt_dir	(内部処理用)
7	public int limitid_count	リミット ID 情報配列リストに保存されている limit id 数 (limitid, lowerdb, upperdb)
8	public TLIMIT_ID_INFO[] limitid_list	リミット ID 情報配列リスト TLIMIT_ID_INFO クラスのインスタンスリスト

３．11．３　メソッド

変数リミット情報保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int set_vid()	変数 ID です。
4	public int add_limitid()	limitid_list[]配列リストにリミット ID 情報を追加します。
5	public int set_limitid()	limitid_list[]配列リストの配列位置を指定してリミット ID 情報を設定します。
6	public int copy()	TLIMIT_INFO クラスのインスタンスをコピーします。

３．11．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します
Dispose()の後、このインスタンスを使用することはできません。

３．11．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）

３．11．３．３　set_vid()－変数 ID の設定

メソッドの引数で与えられたデータを変数 ID が指定する変数の値に設定します。　そして変数 ID が有するデータフォーマットなどの情報を、当該インスタンスのプロパティに設定します。。

【構文】

public int set_vid(UInt32 vid)

【引数】

vid

リミット情報を所有する変数 ID です。

【戻り値】

返却値	意　味
0	設定できた。
-1	指定された変数 ID が登録されていなかった。

【説明】

リミット情報を所有する変数 ID として vid をプロパティ内に設定します。
そして、その変数が所有するプロパティ format, asize を取り出し、当該インスタンスのプロパティ内に設定します。

３．11．３．４　add_limitid()－リミット ID 情報の追加

メソッドの引数で与えられたリミット ID 情報を limitid_list 配列リストに追加します。

【構文】

public int add_limitid(int limit_id, IntPtr upperdb, IntPtr lowerdb)

【引数】

limit_id

リミット ID

upperdb

リミットの上限値

lowerdb

リミットの下限値

【戻り値】

返却値	意　味
0	追加できた。
-1	指定された変数 ID が登録されていなかった。

【説明】

TLIMIT_ID_INFO リミット ID 配列リストである limitid_list に１個のリミット ID 情報を追加します。
本メソッドを実行すると、limitid_list の中に順番にリミット ID 情報が保存されます。
保存は、limitid_list[limitid_count] 位置になります。追加後、limitid_count は +1 されます。

３．11．３．５　set_limitid() - リミットID情報の設定

メソッドの引数で与えられたリミットID情報をlimitid_list配列リストの指定された配列位置に追加します。

【構文】

```
public int set_limitid( int pos, int limit_id, IntPtr upperdb, IntPtr lowerdb)
```

【引数】

pos
　imitid_list配列リストの要素位置(0,1,2..)
limit_id
　リミットID
upperdb
　リミットの上限値
lowerdb
　リミットの下限値

【戻り値】

返却値	意　味
0	設定できた。
-1	指定された変数IDが登録されていなかった。 指定posが配列リスト外の位置であった。

【説明】

TLIMIT_ID_INFOリミットID配列リストであるlimitid_listのposで指定された要素位置に１個のリミットID情報を追加します。
保存は、limitid_list[pos] 位置になります。

３．11．３．６　copy()－TLIMIT_INFO クラスのコピー

TLIMIT_INFO クラスのインスタンスを別の TLIMIT_INFO のインスタンスにコピーします。

【構文】

public static int copy(ref TLIMIT_INFO dst_info, TLIMIT_INFO src_info)

【引数】

dst_info

　　コピー先インスタンス

src_info

　　コピー元インスタンス

【戻り値】

返却値	意　味
0	コピーできた。
(-1)	コピーできなかった。

【説明】

TLIMIT_INFO クラスのインスタンスを別の TLIMIT_INFO のインスタンスにコピーします。

引数として dst_info が指定されない場合は、当該インスタンスがコピー先になります。
引数 dst_info が指定された場合は、static のメソッドですので注意ください。

３．12　TLIMIT_LIST - 装置変数リミット情報リストクラス

複数の装置変数のリミット情報を保存する配列リストクラスです。

３．12．１　コンストラクタ

TLIMIT_LIST クラスのインスタンスを生成します。

３．12．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明
1	public int vid_count	
2	public TLIMIT_INFO[] limit_list	

３．12．３　メソッド

変数リミット情報保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int add ()	limit_list[]配列リストにリミット情報を追加します。
4	public static int set_limit_list_info()	TLIMIT_LIST クラスの情報を変数情報の中の TV_INFO クラスの limit プロパティに設定します。

３．12．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．12．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）

３．12．３．３　add()－リミット情報の追加

メソッドの引数で与えられた変数リミット情報を limit_list リストに追加します。

【構文】

public int add(TLIMIT_INFO info)

【引数】

info

変数リミット情報(TLIMIT_INFO のインスタンス)

【戻り値】

返却値	意　味
0	追加できた。
-1	指定された変数 ID が登録されていなかった。

【説明】

TLIMIT_INFO リミット情報の配列リストである limit_list に１個のリミット情報を追加します。
本メソッドを実行すると、limit_list の中に順番にリミットリミット情報が保存されます。
保存は、limit_list[vid_count] 位置になります。
追加後、vid_count は +1 されます。

３．12．３．４　set_limit_list_info()－リミット情報リストを装置変数に設定

TLIMIT_LIST クラスのインスタンスに含まれているリミット情報を装置変数情報内に設定します。
装置変数は、リストの各要素である TLIMIT_INFO クラス内のプロパティ vid に変数 ID が設定されています。

【構文】

public static int set_limit_list_info(TLIMIT_LIST list)

【引数】

list

リミット情報リスト、TLIMIT_LIST クラスのインスタンス

【戻り値】

返却値	意　味
0	設定
(-1)	設定できなかった。

【説明】

変数 ID の情報(TV_INFO のインスタンス)のプロパティ limit に limit_list の中のリミット情報を設定します。

TLIMIT_LIST クラスの limit_list 配列リストの各要素には､設定したいリミット情報 TLIMIT_INFO のインスタンスが保存されています。各要素の TLIMIT_INFO のインスタンス内には、装置変数 ID が含まれています。

３．13　TLIMIT_ID_INFO - リミット ID 情報クラス

リミット ID 情報を保存するクラスです。

３．13．１　コンストラクタ

TLIMIT_ID_INFO クラスのインスタンスを生成します。

３．13．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明
1	public int limit_id	リミット ID
2	public IntPtr upperdb	上限値
3	public IntPtr lowerdb	下限値

３．13．３　メソッド

変数リミット情報保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int set()	リミット ID、上限値、下限値を設定します。
4	public static int copy( )	TLIMIT_ID_INFO クラスのインスタンスをコピーします。

３．13．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．13．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）

３．13．３．３　set()－リミットID情報の設定

リミットID、上限値、下限値の値を設定します。

【構文】

public int set(int lmt_id, int fmt, IntPtr upper, IntPtr lower)

【引数】

lmt_id

　　変数リミットID

fmt

　　上下限値のフォーマット

upper

　　上限値

lower

　　下限値

【戻り値】

返却値	意　味
0	設定できた。
-1	①fmtが数値フォーマットではなかった。 ②upper，lowerの値が有効でなかった。

【説明】

引数で与えられたリミットID、上限値、下限値の値をそれぞれのプロパティに設定します。

３．13．３．４　copy()－TLIMIT_ID_INFOクラスのコピー

TLIMIT_D_INFOクラスのインスタンスを別のTLIMIT_IID_NFOのインスタンスにコピーします。

【構文】

public static int copy(ref TLIMIT_ID_INFO dst, TLIMIT_ID_INFO src, int format, int size)

【引数】

dst_info

　　コピー先インスタンス

src_info

　　コピー元インスタンス

【戻り値】

返却値	意　味
0	コピーできた。
(-1)	コピーできなかった。(srcが無効であった)

【説明】

TLIMIT_ID_INFOクラスのインスタンスを別のTLIMIT_ID_INFOのインスタンスにコピーします。

３．14　TVLIMIT_EVENT_INFO - リミットイベント情報クラス

リミットイベント情報を保存するクラスです。

リミットイベント情報は、DSHEEng5 エンジンが変数値のリミット監視で上限値から下限値の範囲を超えた瞬間を検出した際に APP のイベント・ハンドラーに渡すための情報です。

３．14．１　コンストラクタ

TVLIMIT_EVENT_INFO クラスのインスタンスを生成します。

３．14．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明
1	public UInt32 vid	リミット ID
2	public int format	上限値
3	public int asize	下限値
4	public string value	検出したときの変数値（文字列）
5	public int limitid	検出したリミット ID
6	public int dir	値の変化方向（0=上限を超えた、1=加減を超えた）

３．14．３　メソッド

変数リミット情報保存クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。

３．14．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．14．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）

３．15　class_TRACE － SV トレース情報管理クラス

class_TRACE クラスは GEM が定める SV トレース情報の登録と保存、参照、管理サービスを行うためのクラスです。トレースの対象変数は、SV 変数だけです。

トレース情報は、ホストからの S2F23 メッセージによって与えられます。

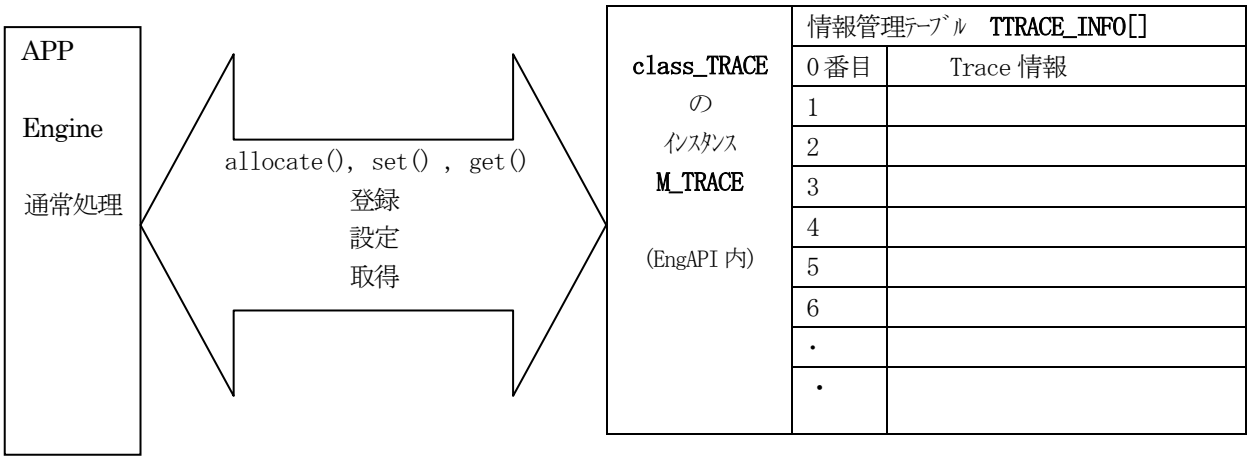
S2F23 メッセージの処理には２通りの方法があります。

（1）DSHEng5 が内部でデコードし、トレース情報を class_TRACE 管理登録し、そのあと、トレースを開始する方法

（2）APP が S2F23 を DSHEng5 から受取り、デコードし、トレース情報を class_TRACE 管理登録し、そのあと、トレースを開始する方法

以下、APP によって S2F23 を処理するケースについて説明します。

概略のトレース情報の構成と登録、設定、取得については以下の通りです。



３．15．１　コンストラクタ

	名前	説明
1	public class_TRACE(int max_id)	ｲﾝｽﾀﾝｽを生成します。 引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_TRACE クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)

引数 max_id は管理する ID の最大数を指定します。プロパティ tr_info_tab[] の配列サイズになります。

(本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_TRACE を生成します。)

３．15．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1	TTRACE_INFO[] tr_info_tab	トレース情報の登録テーブル	

TTRACE_INFO クラスについては、**３．16　TTRACE_INFO クラス**　の説明を参照ください。

３．15．３　メソッド

通信エンジン内に予め class_TRACE のインスタンスが準備されています。このインスタンスは、EngAPI クラスの中にあり、名前は M_TRACE です。
すべてのトレース情報が、このインスタンスの中に登録され、管理されることになります。

APP は、M_TRACE インスタンスに対してメソッドを実行することによって特定 ID のトレース情報をアクセスすることができます。

APP が使用できる class_TRACE クラスのメソッドは次の一覧表のとおりです。

	メソッド名	説明
1	public int allocate()	トレース ID を登録します。
2	public int set()	指定 ID のトレース情報を設定します。
3	public int get()	指定 ID のトレース情報を取得します。
4	public void delete_all_id()	登録されているすべてのトレース ID を削除します。
5	public void delete()	指定された ID を削除します。
6	public int enable()	指定された ID のトレースモニター処理を開始します。
7	public int get_id_count()	登録されている ID 数を取得します。
8	public int get_id_list()	登録されている ID リストを取得します。

３．15．３．１　allocate()－トレース ID の予約登録

指定されたトレース ID を予約登録します。

【構文】

public int allocate_id(string id)

【引数】

id

予約したいトレース ID

【戻り値】

返却値	意　味
0	予約登録できた。(新規登録)
1	ID が登録済であった。
(-1)	登録できなかった。

【説明】

指定されたトレース ID を管理下に登録します。
もし、既に登録されていた場合、トレース ID 以外のトレース情報をクリアします。戻り値=1 を返却します。
登録するスペースが無かった場合は、(-1)を返却します。

３．15．３．２　set()－トレース情報の設定

指定された ID のトレース情報を設定します。
未登録であれば、allocate します。そして設定します。

【構文】

public int set(string id, TTRACE_INFO src_info)

【引数】

id

設定したいトレース ID

src_info

設定したいトレース情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	src_info が null であった。

【説明】

指定された ID に、src_info で指定されたインスタンスの情報を設定します。

もし、ID が未登録の場合は、最初に allocate()メソッドを使って内部で登録します。
既に登録済の場合は、元の情報をクリアした後に設定します。

３．15．３．３　get() - トレース情報の取得

指定された ID のトレース情報を取得します。

【構文】

```
public int get(string id, ref TTRACE_INFO dst_info)
```

【引数】

id

　　取得したいトレース ID

dst_info

　　取得したトレース情報を保存するインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたトレース ID が登録されていなかった。

【説明】

指定された ID のトレース情報を dst_info のインスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

３．15．３．４　delete_all_id() 全トレース情報の消去

登録されているすべてのトレース情報を削除します。

【構文】

```
public void delete_all_id()
```

【引数】

なし。

【戻り値】

なし。

【説明】

登録されているすべてのトレース情報を削除します。
トレース実行中のものは終了させます。

３．15．３．５　delete()－トレース情報の削除

指定された ID のトレース情報を削除し、登録から外します。

【構文】

public void delete(string id)

【引数】

id

削除したいトレース ID

【戻り値】

なし。

【説明】

指定された ID のトレース情報を削除します。そして登録から外します。
トレースが実行中であれば実行終了させます。

３．15．３．６　enable()－トレース・サンプルの実行

指定された ID のトレースのサンプリングの実行を開始します。(デバッグ目的用)

【構文】

public int enable(string id)

【引数】

id

サンプリング実行したいトレース ID

【戻り値】

返却値	意　味
0	実行できた。
(-1)	指定トレース ID が登録されていなかった。

【説明】

指定された ID のトレースのサンプリングを実行します。
トレース情報の中の totsmp(総サンプル数)の値によって以下の意味になります。

totsmp ＞ 0 ：　サンプリングを開始します。
totsmp ＝ 0 ：　サンプリングを停止します。

サンプリングは、S2F23 メッセージによって送信されるトレース情報の受信によって実行されます。
開始／停止は totsmp の値によって判断されます。

３．15．３．７　get_id_count() - 登録されている ID 数の取得

DSHEng5 内に登録されているトレース ID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されているトレース ID 数を取得します。

３．15．３．８　get_id_list() - 登録されている ID リストの取得

DSHEng5 内に登録されているトレース ID リストを取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大容量

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list リスト内に取得します。

戻り値は、取得した ID 数です。

３．16　TTRACE_INFO- トレース情報保存クラス

TTRACE_INFO クラスは、１個のトレース情報の保存に使用されます。

３．16．１　コンストラクタ

３．16．１．１　コンストラクタ

TTRACE_INFO クラスのインスタンスを生成します。

例 : TR_123 のトレース ID のインスタンスを生成します。

```
string TR_123 = “TR_123”;
TTRACE_INFO TR_123 = new TTRACE_INFO();
```

３．16．１．２　デストラクタ

TTRACE_INFO のインスタンスを破棄します。
破棄する前に、そのインスタンスが使用している資源（Unmanaged Memory）をシステムに返却します。

３．16．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string trid	トレース ID
2	public int format	トレース ID のフォーマット format = A(ASCII)がデフォルト
3	public int dsper	サンプリング周期（文字列表記）
4	public int dsper_time	サンプリング周期（数値表記）
5	public int totsmp	合計サンプル数（数値） (=0 でサンプル終了)
6	public int repgsz	レポートグループのサイズ
7	public int gsz_fmt	repgsz のフォーマット
8	public int svid_count	トレース対象 SVID の数
9	public UInt32[] svid_list	トレース対象 SVID 保存リスト

３．16．３　メソッド

トレース情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_parameter()	トレース条件の設定 dsper, totsmp, repgsz
4	public int add_svid()	トレース対象 SVID リストに svid を追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．16．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．16．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．16．３．３　set_parameter()－トレース・パラメータの設定

トレース条件の以下のパラメータを設定します。
dsper, totsmp, repgsz の３個です。

【構文】

public void set_parameter(string dsper, int totsmp, int repgsz)

【引数】

dsper

サンプリング周期　書式　：　hhmmsscc(時分秒　+100ms x cc

totsmp

合計サンプル数　(=0 ならばサンプリング終了)

regsz

レポートグレープのサイズ

【戻り値】

なし。

【説明】

３つの引数それぞれをプロパティ、dsper, totsmp, regsz に設定します。

プロパティ　dsper_time には、dsper を 10ms 単位の整数値に変換した値を設定します。

３．16．３．４　add_svid()－SVID の追加

SV 変数 ID をプロパテイ sv_list 配列リストに追加設定します。

【構文】

public int add_svid(UInt32 svid)

【引数】

svid

追加する SVID

【戻り値】

返却値	意　味
0	追加できた。
(-1)	svid の SV が登録されていなかった。

【説明】

トレースしたい SV 変数 ID を sv_list 配列リストに１個追加します。
追加したあと、svid_count + 1 します。この svid_list 内の svid が１つのグループとしてサンプリングが行われます。

３．16．３．５　copy() - インスタンスのコピー

TTRACE_INFO クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TTRACE_INFO dst, TTRACE_INFO src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

TTRACE_INFO インスタンス src を dst にコピーします。

３．17　TTRACE_SV- トレース SV データ保存クラス

TTRACE_SV クラスは、サンプリングで得られた 1 個の SV データの保存に使用されます。
S6F1 メッセージの トレースデータ Encode / Decode の入力、出力用クラスとして使用されます。

３．17．１　コンストラクタ

TTRACE_SV クラスのインスタンスを生成します。

３．17．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public uint svid	SV 変数 ID
2	public int format	svid のフォーマット
3	public int asize	SV 値のサイズ
4	public IntPtr sv	SV 値保存メモリポインタ

３．17．３　メソッド

トレース情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	サンプリングデータを設定します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．17．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．17．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．17．３．３　set() - SV 値の設定

当該インスタンスにデータ値を設定します。

【構文】

public void set(int fmt, IntPtr sv, int size)

【引数】

fmt

　　データのフォーマット

sv

　　データ値のポインタ

size

　　データサイズ

【戻り値】

なし。

【説明】

当該インスタンス のプロパテイ format, asize, sv それぞれに引数の fmt, size, sv を設定します。

３．17．３．４　copy() - インスタンスのコピー

TTRACE_SV クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TTRACE_SV dst, TTRACE_SV src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

TTRACE_SV インスタンス src を dst にコピーします。

３．18　TTRACE_DATA- サンプリング結果データリスト保存クラス

TTRACE_DATA クラスは、サンプリングで得られた 1 個以上の SV データの保存に使用されます。

３．18．１　コンストラクタ

TTRACE_DATA クラスのインスタンスを生成します。

３．18．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string trid	トレース ID
2	public int smpln	sampling 番号
3	public int count	トレースによる SV 値リスト sv_list[]の数
4	public TTRACE_SV[] sv_list	トレース結果情報配列リスト

３．18．３　メソッド

トレース情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void add_sv()	サンプリングデータを追加します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．18．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．18．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．18．３．３　add_sv()－SV 値の追加

トレース結果データ 1 個をプロパティ sv_list[]に追加します。

【構文】

public void add_sv(TTRACE_SV info)

【引数】

sv_info

トレース結果情報 (SVID1 個分)

【戻り値】

なし。

【説明】

引数 info の内容を sv_list リストに追加します。
追加した後、count +1 します。

３．18．３．４　copy()－インスタンスのコピー

TTRACE_DATA クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TTRACE_DATA dst, TTRACE_DATA src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

TTRACE_DATA インスタンス src を dst にコピーします。

３．19　TTRACE_SV- サンプリング結果データ保存クラス

TTRACE_SV クラスは、サンプリングで得られた１個の SV データの保存に使用されます。

３．19．１　コンストラクタ

TTRACE_SV クラスのインスタンスを生成します。

３．19．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public uint svid	SVID
2	public int format	SV 値のフォーマット
3	public int asize	SV 値のサイズ
4	public IntPtr sv	SV 値保存メモリ

３．19．３　メソッド

トレース情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	サンプリングデータを設定します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．19．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイを clear()メソッドによってすべてクリアします。。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．19．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイをすべてクリアします。

３．19．３．３　set()－SV 値の設定

当該インスタンスにトレース結果データ 1 個を設定します。

【構文】

public void set(UInt32 svid, int fmt, IntPtr sv, int size)

【引数】

svid

　　SVID

fmt

　　SV 値のフォーマット

sv

　　SV 値が保存されているメモリ

size

　　sv 値のデータサイズ

【戻り値】

なし。

【説明】

当該インスタンスのプロパティに引数で与えられた情報を設定します。

３．19．３．４　copy()－インスタンスのコピー

TTRACE_SV クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TTRACE_SV dst, TTRACE_SV src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

TTRACE_SV インスタンス src を dst にコピーします。

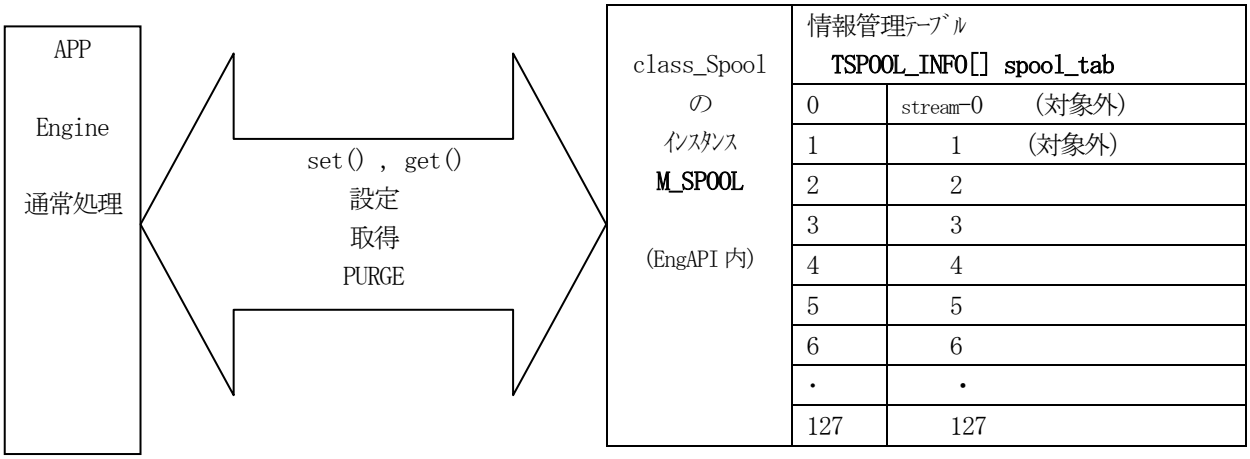
３．20　class_Spool － スプール情報管理クラス

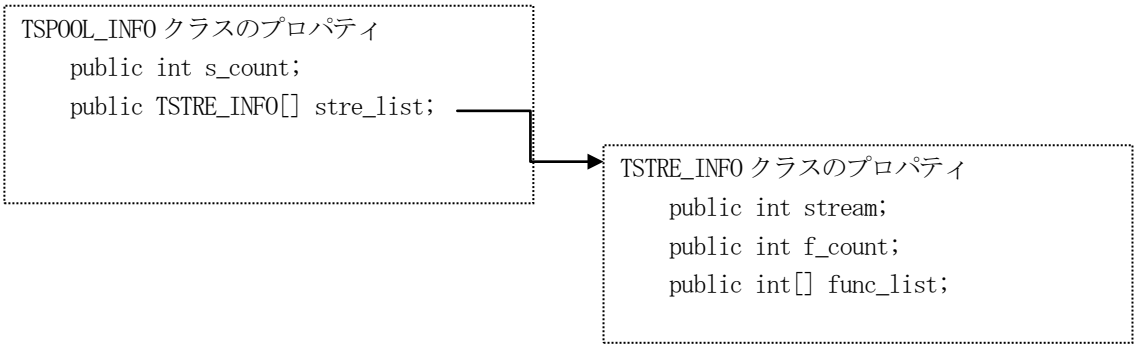
class_Spool クラスは GEM が定めるスプール情報の設定、保存、参照、管理サービスを行うためのクラスです。

class_Spool クラスのインスタンスは、DSHEng5 通信エンジン開始時に、EngAPI クラスの M_SPOOL の名前で生成されます。

スプール対象メッセージの情報は、ホストからの S2F43 メッセージによって与えられます。
S2F43 の処理は、通常 DSHEng5 が自動的に処理します。

概略のスプール情報の構成と設定、参照については以下の通りです。





３．20．１　コンストラクタ

	名前	説明
1	public class_Spool()	ｲﾝｽﾀﾝｽを生成します。 DSHEng5 が開始時に自動的に生成します。

class_Spool クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)

(本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_SPOOL を生成します。)

３．20．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1	TSPOOL_INFO[] spool_tab	スプール情報の登録テーブル 配列は SECS-II メッセージの stream 0 ～127 の要素から構成されます。 (使用されない stream も含まれています)	

TSPOOL_INFO クラスについては、**３．21　TSPOOL_INFO クラス**　の説明を参照ください。

３．20．３　メソッド

通信エンジン内に予め class_Spool のインスタンスが準備されています。このインスタンスは、EngAPI クラスの中にあり、名前は M_SPOOL です。
すべてのスプール情報が、このインスタンスの中に登録され、管理されることになります。

APP は、M_SPOOL インスタンスに対してメソッドを実行することによって特定 ID のスプール情報をアクセスすることができます。

APP が使用できる class_Spool クラスのメソッドは下記一覧表のとおりです。

	メソッド名	説明
1	public void Dispose()	プロパティをクリアし、インスタンスを消滅させます。
2	public void clear()	スプール情報を削除します。
3	public int add_spool_SF ()	spool_tab のプロパティに指定された stream, function のメッセージを追加します。
4	public int get()	Steram コードを指定して、Spool 対象となっている function コードを取得します。
5	public static void purge()	Spool され、送信待ちのメッセージをすべて消去します。
6	public static int copy()	インスタンスを別のインスタンスにコピーします。

3.20.3.1　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティ spool_tab の内容を clear()メソッドによってすべてクリアします。
Dispose()の後、このインスタンスを使用することはできません。

3.20.3.2　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()
public void clear(int s)

【引数】

s

stream コードです。

【戻り値】

なし。

【説明】

（1）clear()の場合は、当該インスタンスのプロパティ spool_tab の内容をすべてクリアします。
（spool_tab に含まれる情報をリセットします。）

（2）clear(int s)の場合は、s で指定された stream の function コードリストをクリアします。

３．20．３．３　add_spool_SF() - stream と function の追加

spoo_tab の中の stream に対する function リストに１個メッセージを追加します。

【構文】

public int add_spool_SF(int s, int func)

【引数】

s

stream コード

func

function コード

【戻り値】

返却値	意　味
0	追加できた。
(-1)	stream s が

【説明】

引数 stream s に対する function 配列リストに引数 func を追加します。

3.20.3.4　get() - スプール情報の取得

指定された ID のスプール情報を取得します。

【構文】

public int get(int s, int[] flist)

【引数】

s

取得したい stream s のスプール情報

flist

取得した function　code list を保存するリスト

【戻り値】

返却値	意　味
>=0	指定 stream の中で Spool 対象になっている function の数
(-1)	引数 s の値がスプール適用外であった。

【説明】

指定された s の stream でスプール対象になっている function コードを flist に取得します。

３．20．３．５　purge() - 送信待ち全スプールメッセージの消去

送信待ちにされているすべてのスプールされたれたメッセージを削除します。

【構文】

public void purge()

【引数】

なし。

【戻り値】

なし。

【説明】

送信待ちにされているすべてのスプールされたれたメッセージを削除します。

３．20．３．６　copy() - インスタンスのコピー

TSPOOL_INFO クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TSPOOL_INFO dst, TSPOOL_INFO src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src インスタンスから dst インスタンスへコピーします。static メソッドです。

３．21　TSPOOL_INFO - スプール情報保存クラス

TSPOOL_INFO クラスは、１個のスプール情報の保存に使用されます。

３．21．１　コンストラクタ

３．21．１．１　コンストラクタ

TSPOOL_INFO クラスのインスタンスを生成します。

３．21．１．２　デストラクタ

TSPOOL_INFO のインスタンスを破棄します。
破棄する前に、そのインスタンスが使用している資源（Unmanaged Memory）をシステムに返却します。

３．21．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int s_count	stream の数(128 固定)
2	public TSTRE_INFO[] stre_list	stream 別 function code 配列リスト stream=6 の function code は stre_list[6]に格納されます。

３．21．３　メソッド

スプール情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	プロパティをクリアし、インスタンスを消滅させます。
2	public void clear()	スプール情報を削除します。
3	public int add_spool_SF ()	spool_tab のプロパティに指定された stream, function のメッセージを追加します。
4	public int get()	Steram コードを指定して、Spool 対象となっている function コードを取得します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．21．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．21．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()
public void clear(int s)

【引数】

s
　　stream コードです。

【戻り値】

なし。

【説明】

（1）clear()の場合は、当該インスタンスのプロパテイ stre_list の内容をすべてクリアします。

（2）clear(int s)の場合は、s で指定されたストリームの stre_list[s]の内容をクリアします。

３．21．３．３　add_spool_SF() - stream と function の追加

spoo_tab の中の stream に対する function リストに 1 個メッセージを追加します。

【構文】

public int add_spool_SF(int s, int func)

【引数】

s

stream コード

func

function コード

【戻り値】

返却値	意　味
0	追加できた。
(-1)	stream s が

【説明】

引数 stream s に対する function 配列リストに引数 func を追加します。

３．21．３．４　get() - スプール情報の取得

指定された ID のスプール情報を取得します。

【構文】

public int get(int s, int[] flist)

【引数】

s

取得したい stream s のスプール情報

flist

取得した function　code list を保存するリスト

【戻り値】

返却値	意　味
>=0	指定 stream の中で Spool 対象になっている function の数
(-1)	引数 s の値がスプール適用外であった。

【説明】

指定された s の stream でスプール対象になっている function コードを flist に取得します。

３．21．３．５　copy() - インスタンスのコピー

TSPOOL_INFO クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TSPOOL_INFO dst, TSPOOL_INFO src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src インスタンスから dst インスタンスへコピーします。static メソッドです。

３．22　TSTRE_INFO - スプール Stream-Function 保存クラス

TSTRE_INFO クラスは、１個の stream コードの function リスト情報の保存に使用されます。

３．22．１　コンストラクタ

３．22．１．１　コンストラクタ

TSTRE_INFO クラスのインスタンスを生成します。

３．22．１．２　デストラクタ

TSTRE_INFO のインスタンスを破棄します。
破棄する前に、そのインスタンスが使用している資源（Unmanaged Memory）をシステムに返却します。

３．22．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int stream	stream code
2	public int f_count	設定されている function コードの数
3	public int[] func_list	function コードの配列リスト

３．22．３　メソッド

スプール情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	プロパティをクリアし、インスタンスを消滅させます。
2	public void clear()	スプール情報を削除します。

３．22．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．22．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

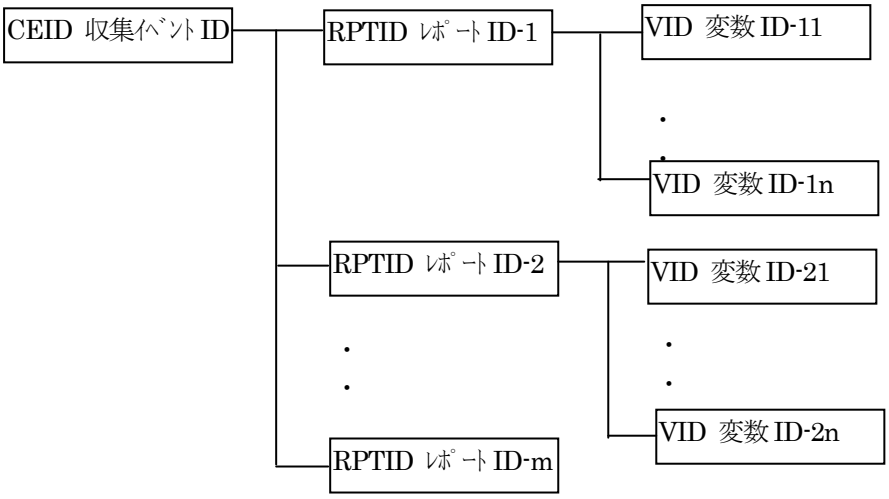
clear()の場合は、当該インスタンスのプロパテイ stre_list の内容をすべてクリアします。

４．CE - 収集イベント情報関連クラス

CE 収集イベント情報は装置から S6F11 メッセージを使ってホストに通報されます。

収集イベントは CE, Report, 装置変数によって構成されます。

CE、Report、変数 V との関係は下図に示す通りです。



（1）CE、Report、Variable（装置変数）はそれらの集合の中でそれぞれ**固有の ID(Identifier)と名前**を有しています。ID は通常 U4 フォーマットの符号無し整数値で表現されます。
（2）CE は 0 個以上の Report で構成されます。
（3）Report は 0 個以上の V(変数)で構成されます。

(注) S6F11 の送受信に関するクラスについては“**DSHEng5 エンジンクラス説明書 通信編**”で説明します。

関連クラスは下表のとおりです。

	クラス名	概要
1	class_CE	全 CE（収集イベント）情報管理クラス
2	TCE_INFO	CE 情報保存クラス
3	TCE_LIST	CE のリンク情報リスト （TCE_LINK[]）
4	TCE_LINK	CE にリンクするレポート ID リスト
5	TEDR_INFO	CE の Enable / Disable（有効/無効）情報リスト
6	TCE_CONTENT	CE のリンク・レポートの詳細情報保存リストクラス （TP_CONTEN[]）

４．１　class_CE - 全収集イベント情報管理クラス

通信エンジンに登録されている全CE(Collection Event)情報を一括して管理するためのクラスです。

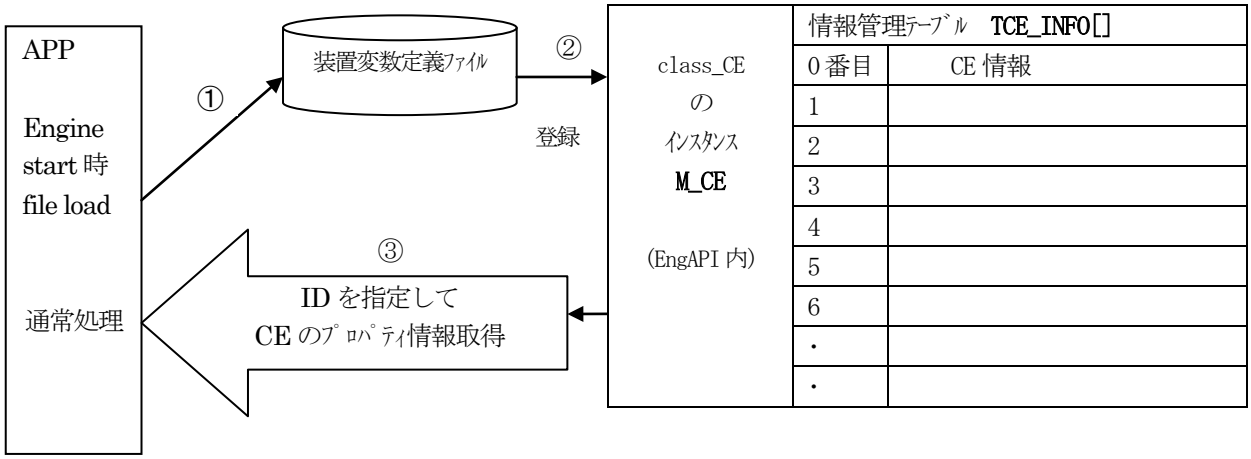
DSHEng5 開始時に、装置変数定義ファイル内に定義されているすべての CEID と情報を本 class_CE クラスのインスタンスに登録します。その class_CE のインスタンスが EngAPI. **M_CE** です。

各 CEID 情報の保存には **TCE_INFO** クラスを使用します。(後述します)

APP は本クラスのインスタンスに対しては、ID を指定して各インスタンスのプロパティを参照（取得）することができます。

プロパティの変更は以下の２つのケースだけが可能です。
① **S2F35** メッセージ（Link Event Report）の受信によるリンクレポート ID プロパティの変更
②メソッドによる ceed（有効/無効）の変更

CE 情報の構成と参照については概略以下の通りです。



４．１．１　コンストラクタ

	名前	説明
1	public class_CE(int max_id)	ｲﾝｽﾀﾝｽを生成します。 引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_CE クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)

引数 max_id は管理する ID の最大数を指定します。プロパティ ce_info_tab[] の配列サイズになります。

(本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_CE を生成します。)

４．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1	TCE_INFO[] ce_info_tab	CE の登録テーブル	

TCE_INFO クラスについては、**４．２　TCE_INFO クラス**　の説明を参照ください。

４．１．３　メソッド

通信エンジン内に予め class_CE のインスタンスが準備されています。このインスタンスは、EngAPI クラスの中に M_CE の名前で準備されており、この中にすべての CE 情報が登録され、管理されることになります。

APP は、このインスタンス、M_CE に対してメソッドを実行することによって特定 ID の CE 情報（プロパティ）にアクセスすることができます。

例えば、CE_ControlState　にリンクしている変数 ID を取得する場合、次のようなコーディングになります。

```
TCE_INFO info = new TCE_INFO();
UInt32[]  rpid_list = new UInt32[0];
int result = EngAPI.M_CE.get_rp_list( CE_ControlState, ref rpid_list);
```

APP が使用できる class_CE クラスのメソッドは下記一覧表のとおりです。

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int get_id_name_list()	登録されている ID,名前リストを取得します。
4	public int get_id_by_name()	名前からその ID を取得します。
5	public int get_name()	CEID の名前を取得します。
6	public int get()	指定 ID の CE 情報を取得します。 （TCE_INFO ： CE 情報用ｸﾗｽ）
7	public int get_rp_list()	CEID にリンクされているレポート ID リストを取得します。
8	public int get_rp_name_list()	CEID にリンクされているレポート ID の名前リストを取得します。
9	public int set_ceed()	ceed(有効/無効) を設定します。 （ceed=無効で当該イベント送信はされない）
10	public int get_ceed()	ceed(有効/無効) を取得します。

４．１．３．１　get_id_count() - 登録されている ID 数の取得

DSHEng5 内に登録されている CEID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されている CEID 数を取得します。

４．１．３．２　get_id_list() - 登録されている ID リストの取得

DSHEng5 内に登録されている CEID の ID リストを取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大容量

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list リスト内に取得します。

戻り値は、取得した ID 数です。

４．１．３．３　get_id_name_list() - 登録されている ID,名前リストの取得

ECCE に登録されているすべての ID とその名前をそれぞれのリストに取得します。

【構文】

public int get_id_name_list(UInt32[] id_list, string[] name_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

name_list

　　CE 名を保存するリスト

max_size

　　両方の list の最大サイズ

【戻り値】

返却値	意　味
ID 数	取得した ID 数

【説明】

登録されている ID とその名前をそれぞれ id_list, name_list リストに取得します。
戻り値は、取得した ID 数です。

４．１．３．４　get_id_by_name() - CE 名から ID を取得

登録されている CE 名から CEID を取得します。

【構文】

public int get_id_by_name(string vname, ref UInt32 id)

【引数】

vname

　　CE 名

id

　　ID 格納用領域

【戻り値】

返却値	意　味
0	取得できた。
-1	CE 名で指定された CE はなかった。

【説明】

CE 名から CEID を取得します。
取得できた場合は 0 を、CE 名が見つからなかった場合は (-1)を返却します。

４．１．３．５　get_name() - ID から CE 名を取得

登録されている CEID から CE 名を取得します。

【構文】

public string get_name(UInt32 id)

【引数】

id

CEID

【戻り値】

返却値	意　味
CE 名	取得できた。
“”	ID が見つからなかった。

【説明】

ID から CE 名を取得します。

４．１．３．６　get() - CE 情報の取得

DSHEng5 から、TCE_INFO クラスに設定された CE 情報を指定された ID の CE 情報を取得します。

【構文】

public int get(UInt32 id, ref TCE_INFO info)

【引数】

id

取得したい CEID

info

取得 CE 情報保存用ンスタンス

【戻り値】

返却値	意　味
0	取得できた。
-1	取得できなかった。

【説明】

id で指定された CE 情報を info で指定された TCE_INFO クラスのインスタンスに取得します。

TCE_INFO クラスについては、4.2 を参照して下さい。

４．１．３．７　get_rp_list() - リンクされているレポート ID リストの取得

指定された CEID にリンクされているレポート ID リストを取得します。

【構文】

public int get_rp_list(UInt32[] id, ref UInt32[] rpid_list)

【引数】

id

CEID です。

rpid_list

レポート ID を保存するための配列リストです。

【戻り値】

返却値	意　味
>=0	rpid_list リストに取得したレポート ID 数
(-1)	指定された ID の CE が無かった。

【説明】

指定された CEID にリンクされているレポート ID リストを取得します。

４．１．３．８　get_rp_name_list() - リンクされているレポート名リストの取得

指定された CEID にリンクされているレポート ID と名前リストを取得します。

【構文】

public int get_rp_name_list(UInt32[] id, ref UInt32[] rpid_list, ref string[] name_list)

【引数】

id

CEID です。

rpid_list

レポート ID を保存するための配列リストです。

name_list

レポート名を保存する配列リストです。

【戻り値】

返却値	意　味
>=0	id_list(name_list)リストに取得した ID 数
(-1)	指定された ID の CE が無かった。

【説明】

id で指定された CE にリンクされている変数の ID と名前をそれぞれ vid_list, name_list に取得します。

４．１．３．９　set_ceed()－イベントの有効／無効の設定

指定されたCEIDの有効／無効の設定を行います。

【構文】

```
public int set_ceed( UInt32 id, bool f)
```

【引数】

id

設定したいCEIDです。

f

設定したいceedの値です。

f = true ： 有効、　f= false ： 無効

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたIDのCEが無かった。

【説明】

idで指定されたCEのイベントを有効か、無効に設定します。
ceedの値は、TCE_INFOクラスの プロパティ名、enabledに設定されます。

有効であれば、そのIDのイベントの事象が発生すれば、当該CEIDのS6F11の送信が行われます。
無効に設定されると、当該IDのイベントの事象が発生しても、S6F11は送信されないことになります。

４．１．３．10　get_ceed() - イベントの有効／無効の取得

指定されたCEIDの有効／無効の状態を取得します。

【構文】

```
public int get_ceed( UInt32 id, ref bool f)
```

【引数】

id

取得したいCEIDです。

f

取得したceedを保存する領域

f = true ： 有効　f = false ： 無効

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたIDのCEが無かった。

【説明】

idで指定されたCEのイベントを有効/無効の状態を取得します。

４．２　TCE_INFO - CE 情報保存クラス

TCE_INFO クラスは、１個の CE 情報の保存に使用されます。

４．で説明した class_CE クラスの中で ce_info_tab 配列リストがありましたが、その配列要素になります。

４．２．１　コンストラクタ

TCE_INFO クラスのインスタンスを生成します。

例 ： CE_123 の CEID のインスタンスを生成します。

```
UInt32 CE_123 = 100;
TCE_INFO CE_123 = new TCE_INFO();
```

４．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public UInt32 ceid	CEID
2	public bool enabled	有効/無効のフラグ true=有効、false=無効
3	public string name	CEID の名前
4	public int rp_count	リンクされているレポートの数
5	public UInt32[] rpid_list	リンクされているレポート ID 配列リスト
6	public string[] rpname_list	リンクされているレポート ID の名前リスト

(注) レポートリンク情報の変更は、DSHEng5 が行います。

４．２．３．　メソッド

CE 情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int copy() public static int copy()	インスタンスを別のインスタンスにコピーします。 （static メソッドの２種類あります。）

４．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

４．２．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）も開放します。）

４．２．３．３　copy() - TCE_INFO 情報コピー

TCE_INFO の情報を別の TCE_INFO クラスのインスタンスにコピーします。
２種類のものがあります。

【構文】

```
public int copy(TCE_INFO src_info)
public static int copy(ref TCE_INFO dst_info, TCE_INFO src_info)
```

【引数】

src_info
　　コピー元の TCE_INFO クラスのインスタンス
dst_info
　　コピー先の TCE_INFO クラスのインスタンス

[戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src_ info が null であった。

【説明】

１番目のメソッドは、当該インスタンスが他のインスタンスからのコピーに使用します。

２番目のメソッドは、当該インスタンスが、他の２つインスタンス間でのコピーになります。

４．３　TCE_LIST - CE のリンク情報リスト

TCE_LIST クラスは、Report リンクリストを配列に持つレポートリンク情報配列リストを保存するために使用します。

レポートの配列リストの要素は後述する TCE_LIST クラスになります。

４．３．１　コンストラクタ

TCE_LIST クラスのインスタンスを生成します。

４．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	配列リストのサイズ
2	public TCE_LINK[] ce_list	1 個の CEID にリンクされる Report 情報配列リスト

４．３．３　メソッド

TCE_LIST クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void add_ce_link()	TCE_LINK クラスのインスタンスを ce_list に追加します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

４．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスのプロパティをすべて消去、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
破棄します。
Dispose()の後、このインスタンスを使用することはできません。

４．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

４．３．３．３　add_ce_link()- リンク情報の追加

アラーム ID 配列リストにアラーム ID を１個追加します。

【構文】

public void add_ce_link(TCE_LINK link)

【引数】

link

追加するリンク情報

【戻り値】

なし。

【説明】

プロパティ ce_list に引数 link の CE リンク情報を追加し、プロパティ count +1 します。
TCE_LINK クラスについては、次節の ４．４ を参照ください。

４．３．３．４　copy()-TCE_LIST 情報コピー

TCE_LIST の情報を別の TCE_LIST クラスのインスタンスにコピーします。

【構文】

public static int copy(ref TCE_LIST dst, TCE_LIST src

【引数】

src

コピー元の TCE_LIST クラスのインスタンス

dst

コピー先の TCE_LIST クラスのインスタンス

[戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src インスタンスから dst インスタンスへコピーします。static メソッドです。

４．４　TCE_LINK - CE にリンクするレポート ID リスト

TCE_LINK クラスは、1 個の CE にリンクするレポート ID の配列リスト情報を保存するクラスです。

４．４．１　コンストラクタ

TCE_LINK クラスのインスタンスを生成します。

４．４．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public UInt32　ceid	CEID
2	public int rp_count	リンクレポート配列リストのサイズ
3	public UInt32[] rpid_list	リンクレポート ID の配列リスト

４．４．３　メソッド

TCE_LINK クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void add_rpid()	rpid_list にレポート ID を追加します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

４．４．３．１　Dispose() - インスタンスの破棄

当該インスタンスのプロパティをすべて消去、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
破棄します。
Dispose()の後、このインスタンスを使用することはできません。

４．４．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

４．４．３．３　add()－レポートIDを追加

リンクレポートID配列リストにレポートIDを１個追加します。

【構文】

public int add(UInt32 rpid)

【引数】

rpid

追加するレポートID

【戻り値】

追加した後のプロパティ rp_count の値を返却します。

【説明】

リンクレポートID配列リスト、プロパティ rp_list に引数 rpid を追加し、プロパティ count　+1 します。
追加後、配列リストに保存されているレポートID数を返却します。

４．４．３．４　copy()－TCE_LIST情報コピー

TCE_LINK の情報を別の TCE_LINK クラスのインスタンスにコピーします。

【構文】

public static int copy(ref TCE_LINK dst, TCE_LINK src)

【引数】

src

コピー元の TCE_LINK クラスのインスタンス

dst

コピー先の TCE_LINK クラスのインスタンス

[戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src インスタンスから dst インスタンスへコピーします。static メソッドです。

４．５　TEDER_INFO - CE の Enable / Disable (有効/無効)情報リスト

CE の有効/無効情報を保存するクラスです。
S2F37 メッセージ（ Enable / Disable Event Report）をデコードした結果を保存します。

４．５．１　コンストラクタ

TEDER クラスのインスタンスを生成します。

４．５．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public bool ceed	Enable / Disable 設定フラグ true = Enable,　false = Disable
2	public int count;	Enable/Disable 設定対象 CEID 数
3	public UInt32[] id_list	設定対象 CEID の配列リスト

４．５．３　メソッド

TEDER_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void clear()	すべてのプロパティの値をクリアします。 ceed = false になります。
2	public int add()	id_list に指定された CEID を追加します。

４．５．３．１　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイをすべてクリアします。

４．５．３．２　add() - CEID を追加

CEID 配列リストに CEID を１個追加します。

【構文】

public int add(UInt32 ceid)

【引数】

ceid

追加する CEID

【戻り値】

追加した後のプロパティ count の値を返却します。

【説明】

CEID 配列リスト、プロパティ id_list に引数 ceid を追加し、プロパティ count +1 します。
追加後、配列リストに保存されている CEID 数を返却します。

４．６　TCE_CONTENT - CE のリンク・レポートの詳細情報保存リスト

TCE_CONTENT クラスは、CEID にリンクされているレポート情報を保存します。
そして、それらレポートにリンクされている変数 ID、情報の詳細（値）も含みます。

４．３．１　コンストラクタ

TCE_CONTENT のインスタンスを生成します。

４．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public uint ceid	CEID です。
2	public int rp_count	リンクされているレポートの数です。
3	public TRP_CONTENT[] rp_list	リンクされているレポート情報リストです。 (5.4　TRP_CONTENT 参照)

４．３．３　メソッド

レポート情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int get_content()	TCE_INFO クラスのリンク情報から、変数値を含む情報を取得します。(S6F11 をエンコードするための情報を)

４．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスのプロパティをすべて消去、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドを使ってすべてクリアした後、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

４．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）

４．３．３．３　get_content() - CE のリンク情報の詳細を取得_

指定されたCEIDのレポートリンク情報ならびにそのレポートにリンクされている変数IDと値の情報を取得します。
これは、S6F11 送信時に使用されます。

【構文】

public int get_content(TCE_INFO ce_info, ref TCE_CONTENT cinfo)

【引数】

ce_info

TCE_INFO クラスのインスタンス - CE 定義情報が保存されています。

cinfo

TCE_CONTENT クラスのインスタンス - この中にレポート、変数リンク情報を取得します。

[戻り値】

返却値	意　味
= 0	取得できた。
(-1)	①ce_info が null であった。 ②CE リンク情報の中に登録されていない Report ID または 変数 ID が含まれていた。

【説明】

ce_info の ceid にリンクされているレポート ID、そのレポート ID にリンクされている変数 ID の変数情報を cinfo インスタンス内に取得します。
取得した情報を設定する cinfo のプロパティの中に、TRP_CONTENT クラスの配列リスト rp_list があります。

例えば、ce_info のレポート ID リストの先頭に、rpid-1 の ID を有するレポートがリンクされており、その rpid-1 に 2 個の変数 ID　vid-1、vid-2 がリンクされているとすると、以下のような処理を行います。

（1）ce_info から rpid-1 を取り出し、cinfo のプロパティ rp_list[0]に TRP_CONTENT クラスのインスタントを生成します。
（2）次に、rpid-1 にリンクされている変数が 2 個あるので、rp_list[0]のインスタンスのプロパティ v_list リストに TV_CONTENT クラスのインスタンスを生成します。

TRP_CONTENT	format	value	v_count	link_list
v_list[0] =vid-1	U2	100	0	null
v_list[1] =vid-2	L	2	2	TV_CONTENT link_list[2]

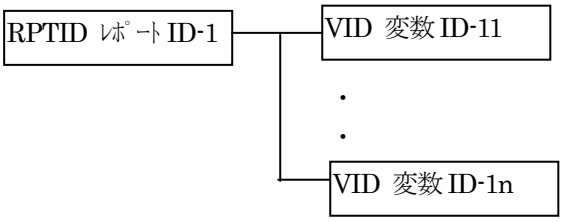
v_list[1]の link_list 内容が下の表

TV_CONTENT	format	value	v_count	link_list	
vid-2-0	U1	12	0	null	
vid-2-1	B	5	0	null	

５．　Report - レポート情報関連クラス

レポート情報は、装置からホストに S6F11 メッセージを通知する際に使用されます。

レポート情報は装置変数によって構成（リンク）されます。
Report と変数 V との関係は次に示す通りです。



Report は 0 個以上の V(変数)で構成されます。

関連クラスは下表のとおりです。

	クラス名	概要
1	class_Report	全レポート情報管理クラス
2	TRP_INFO	レポート情報保存クラス
3	TRP_LIST	レポートにリンクされる全変数分のリンク情報リストクラス(TRP_LINK[])
4	TRP_LINK	レポート（個別）の変数リンクリスト
5	TRP_CONTENT	レポートの変数リンクリストさ要塞保存リスト
6	TV_CONTENT	変数の変数リンク詳細保存リスト

以下、Report 情報関連クラスの詳細について説明します。

５．１　class_Report - 全レポート情報管理クラス

通信エンジンに登録されている全 Report 情報を一括して管理するためのクラスです。

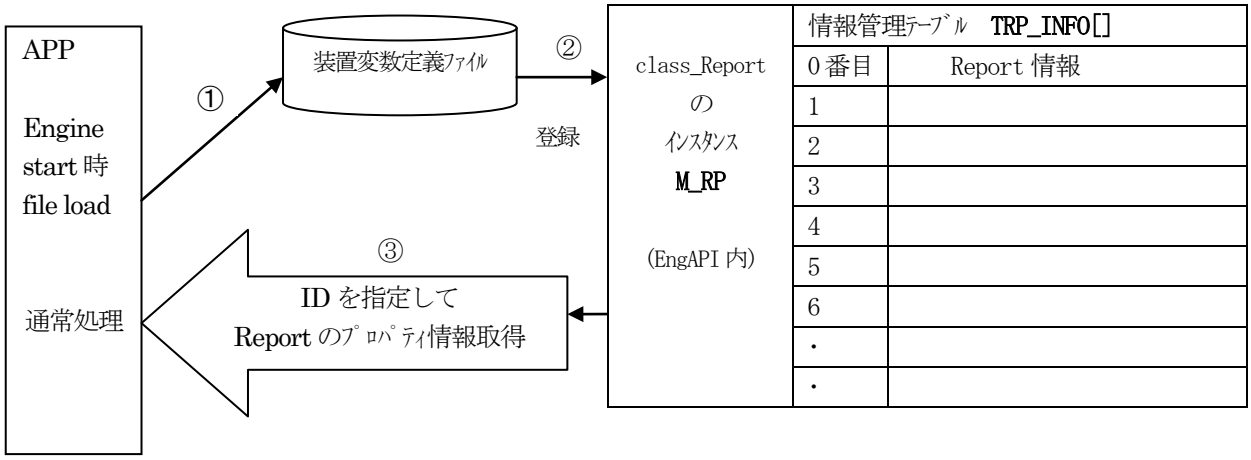
DSHEng5 開始時に、装置変数定義ファイル内に定義されているすべてのレポート ID と情報を本 class_Report クラスのインスタンスに登録します。その class_Report のインスタンスが EngAPI. **M_RP** です。

各 Report ID 情報の保存には TRP_INFO クラスを使用します。(後述します)

APP は本クラスのインスタンスに対しては、ID を指定して各インスタンスのプロパティを参照（取得）することができます。

プロパティの内容を変更設定できるのは、**S2F33** メッセージ（Define Report）による Report ID にリンクする変数 ID の変更による場合だけです。

Report 情報の構成と参照については概略以下の通りです。



５．１．１　コンストラクタ

	名前	説明
1	public class_Report(int max_id)	ｲﾝｽﾀﾝｽを生成します。 引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_Report クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)

引数 max_id は管理する ID の最大数を指定します。プロパティ rp_info_tab[] の配列サイズになります。

(本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_RP を生成します。)

５．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1	TRP_INFO[] rp_info_tab	レポートの登録テーブル	

TRP_INFO クラスについては、**５．１　TRP_INFO クラス**　の説明を参照ください。

５．１．３　メソッド

通信エンジン内に予め class_Report のインスタンスが準備されています。このインスタンスは、EngAPI クラスの中に **M_RP** の名前で準備されており、この中にすべてのレポート情報が登録され、管理されることになります。

APP は、このインスタンス、**M_RP** に対してメソッドを実行することによって特定 ID のレポート情報（プロパティ）にアクセスすることができます。

例えば、RP_ControlState　にリンクしている変数 ID を取得する場合、次のようなコーディングになります。

```
TRP_INFO info = new TRP_INFO();
UInt32[]  vid_list = new UInt32[0];
int result = EngAPI.M_RP.get_vid_list( RP_ControlState, ref vid_list);
```

APP が使用できる class_Report クラスのメソッドは下記一覧表のとおりです。

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int get_id_name_list()	登録されている ID,名前リストを取得します。
4	public int get_id_by_name()	名前からその ID を取得します。
5	public int get_name()	レポート ID の名前を取得します。
6	public int get()	指定 ID のレポート情報を取得します。 （TRP_INFO ： Report 情報用ｸﾗｽ）
7	public int get_v_list()	レポート ID にリンクされている変数 ID リストを取得します。
8	public int get_v_name_list()	レポート ID にリンクされている変数 ID の名前リストを取得します。
9	public int get_TRP_CONTENT()	レポート ID に含む全てのリンク変数をリスト構造で取得します。

５．１．３．１　get_id_count() - 登録されている ID 数の取得

DSHEng5 内に登録されているレポート ID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されているレポート ID 数を取得します。

５．１．３．２　get_id_list() - 登録されている ID リストの取得

DSHEng5 内に登録されているレポート ID の ID リストを取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大容量

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list リスト内に取得します。

戻り値は、取得した ID 数です。

５．１．３．３　get_id_name_list() - 登録されている ID,名前リストの取得

EC レポートに登録されているすべての ID とその名前をそれぞれのリストに取得します。

【構文】

public int get_id_name_list(UInt32[] id_list, string[] name_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

name_list

　　レポート名を保存するリスト

max_size

　　両方の list の最大サイズ

【戻り値】

返却値	意　味
ID 数	取得した ID 数

【説明】

登録されている ID とその名前をそれぞれ id_list, name_list リストに取得します。
戻り値は、取得した ID 数です。

５．１．３．４　get_id_by_name() - レポート名から ID を取得

登録されているレポート名からレポート ID を取得します。

【構文】

public int get_id_by_name(string vname, ref UInt32 id)

【引数】

vname

　　レポート名

id

　　ID 格納用領域

【戻り値】

返却値	意　味
0	取得できた。
-1	レポート名で指定されたレポートはなかった。

【説明】

レポート名からレポート ID を取得します。
取得できた場合は 0 を、レポート名が見つからなかった場合は (-1)を返却します。

５．１．３．５　get_name()－ID からレポート名を取得

登録されているレポート ID からレポート名を取得します。

【構文】

public string get_name(UInt32 id)

【引数】

id

　　レポート ID

【戻り値】

返却値	意　味
レポート名	取得できた。
“”	ID が見つからなかった。

【説明】

ID からレポート名を取得します。

５．１．３．６　get()－レポート情報の取得

DSHEng5 から、TRP_INFO クラスに設定されたレポート情報を指定された ID のレポート情報を取得します。

【構文】

public int get(UInt32 id, ref TRP_INFO info)

【引数】

id

　　取得したいレポート ID

info

　　取得レポート情報の参照

【戻り値】

返却値	意　味
0	取得できた。
-1	取得できなかった。

【説明】

id で指定されたレポート情報を info で指定された TRP_INFO クラスのインスタンスに取得します。

TRP_INFO クラスについては、５．２を参照して下さい。

５．１．３．７　get_v_list() - リンクされている変数 ID リストの取得

指定されたレポート ID にリンクされている変数 ID リストを取得します。

【構文】

public int get_v_list(UInt32[] id, ref UInt32[] vid_list)

【引数】

id

　　レポート ID です。

vid_list

　　変数 ID を保存するための配列リストです。

【戻り値】

返却値	意　味
>=0	id_list リストに取得した ID 数
(-1)	指定された ID のレポートが無かった。

【説明】

指定されたレポート ID にリンクされている変数 ID リストを取得します。

５．１．３．８　get_v_name_list() - リンクされている変数名リストの取得

指定された AL レポート ID にリンクされている変数 ID と名前リストを取得します。

【構文】

public int get_v_name_list(UInt32[] id, ref UInt32[] vid_list, ref string[] name_list)

【引数】

id

　　レポート ID です。

vid_list

　　変数 ID を保存するための配列リストです。

name_list

　　変数名を保存する配列リストです。

【戻り値】

返却値	意　味
>=0	id_list(name_list)リストに取得した ID 数
(-1)	指定された ID のレポートが無かった。

【説明】

id で指定されたレポートにリンクされている変数の ID と名前をそれぞれ vid_list, name_list に取得します。

５．１．３．９　get_TRP_CONTENT()- レポート詳細情報の取得

指定された ID のレポートについて詳細情報を取得します。

【構文】

public int get_TRP_CONTENT(UInt32 rpid, ref TRP_CONTENT rc_info)

【引数】

rpid

　　取得したいレポート ID

rc_info

　　レポート詳細情報を保存する TRP_CONTENT クラスのインスタンス

【戻り値】

返却値	意　味
0	取得できた。
-1	取得できなかった。

【説明】

id で指定されたレポート情報を rc_info で指定された TRP_INFO クラスのインスタンスに取得します。

TRP_CONTENT クラスについては、５．２を参照して下さい。

５．２　TRP_INFO- レポート情報保存クラス

TRP_INFO クラスは、１個のレポート情報の保存に使用されます。
５．１で説明した class_Report クラスの中で rp_info_tab 配列リストがありましたが、その配列要素になります。

５．２．１　コンストラクタ

TRP_INFO クラスのインスタンスを生成します。

例 ： RP_123 のレポート ID のインスタンスを生成します。

```
UInt32 RP_123 = 100;
TRP_INFO RP_123 = new TRP_INFO();
```

５．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public UInt32 vid	レポート ID
2	public string name	レポート ID の名前
3	public int v_count	リンクされている変数の数
4	public UInt32[] vid_list	リンクされている変数 ID 配列リスト
5	public string[] vname_list	リンクされている変数 ID の名前リスト

(注) 変数リンク情報の変更は、DSHEng5 が行います。

５．２．３　メソッド

レポート情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int copy() public static int copy()	インスタンスを別のインスタンスにコピーします。 （static メソッドの２種類あります。）

５．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．２．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）も開放します。）

５．２．３．３　copy() - TRP_INFO の情報コピー

TRP_INFO の情報を別の TRP_INFO クラスのインスタンスにコピーします。
２種類のものがあります。

【構文】

```
public int copy(TRP_INFO src_info)
public static int copy(ref TRP_INFO dst_info, TRP_INFO src_info)
```

【引数】

src_info
　　コピー元の TRP_INFO クラスのインスタンス
dst_info
　　コピー先の TRP_INFO クラスのインスタンス

[戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src_ info が null であった。

【説明】

１番目のメソッドは、当該インスタンスが他のインスタンスからのコピーに使用します。

２番目のメソッドは、当該インスタンスが、他の２つインスタンス間でのコピーになります。

５．３　TRP_LIST - レポートリンク情報リスト

TRP_LIST クラスは、TRP_LINK クラス配列リストを保存するために使用します。
レポートにリンクされる変数の数の配列になります。

５．３．１　コンストラクタ

（省略）

５．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	配列リストのサイズ
2	public TRP_LINK[] rp_list	1 個の RPID にリンクされる変数 ID 情報配列リスト

５．３．３　メソッド

TRP_LIST クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。

５．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスのプロパティをすべて消去、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

５．４　TRP_LINK - レポートの変数リンクリスト

TRP_LINK クラスは、1 個のレポートにリンクする変数 ID の配列リスト情報を保存するクラスです。

５．４．１　コンストラクタ

（省略）

５．４．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	配列リストのサイズ
2	public UInt32[] vid_list	リンク変数 ID の配列リスト

５．４．３　メソッド

TRP_LINK クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。

５．４．３．１　Dispose() - インスタンスの破棄

当該インスタンスのプロパティをすべて消去、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．４．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

５．５　TRP_CONTENT - レポートの変数リンクリスト詳細保存クラス

TRP_CONTENT クラスは、1 個のレポートにリンクする変数 ID の情報の詳細を保存するクラスです。

５．５．１　コンストラクタ

（省略）

５．５．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public uint rpid	
2	public int v_count	変数情報配列リストのサイズ
	public TV_CONTENT[] v_list	変数 ID 情報の詳細クラスの配列

５．５．３　メソッド

レポート情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_rpid()	プロパテイ　rpid に ID を設定います。
4	public int get_content()	TRP_CONTENT の当該インスタンスの内容をコピーします。
5	public static int copy()	TRP_CONTENT のインスタンスを別のインスタンスにコピーします。
6	public static int setup_V_Linklist()	指定変数 ID の値をプロパティ v_list[]に設定します。

５．５．３．１　Dispose() - インスタンスの破棄

当該インスタンスのプロパティをすべて消去、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．５．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

５．５．３．３　setup_V_Linklist()- リンク変数値情報の作成

指定された変数 VID の変数について、その値を TV_CONTENT クラスのインスタンスに設定します。
format = L の変数については、その変数の下にリンクされているすべての変数の値を展開し、保存します。

【構文】

public static int setup_V_Linklist(ref TV_CONTENT vc_info, uint vid)

【引数】

vc_info

指定された変数値を設定する対象クラスのインスタンスです。

vid

値を vc_info 内に設定する変数 ID です。

【戻り値】

返却値	意　味
0	設定できた。
(-1)	vid が登録されていなかった。

【説明】

vid で指定された変数値を vc_info の中に設定します。
変数 vid のフォーマットによって以下のように設定します。
（1）format が L 以外の場合は、値を vc_info の value に設定します。

（2）format が L の場合は、それにリンクされている変数 ID とその情報も TV_CONTENT クラスの vc_info に展開し設定します。
例えば、以下のように展開されます。

TRP_CONTENT	format	value	v_count	link_list
v_list[0] =vid-1	U2	100	0	null
v_list[1] =vid-2	L	2	2	TV_CONTENT link_list[2]

v_list[1]の link_list 内容が下の表

TV_CONTENT	format	value	v_count	link_list	
vid-2-0	U1	12	0	null	
vid-2-1	B	5	0	null	

上の表が当該 TRP_CONTENT のインスタンスの内容です。
v_list の 2 番目の vid-2 が formt=L で、size=2 であり、それにリンクしている変数が vid-2-0, vid-2-1 です。そして、vid-2-0 が U1 format で値=12、vid-2-1 が B format で値=5 の場合の TRP_CONTENT の内容です。

５．６　TV_CONTENT - 変数リンク詳細保存クラス

TV_CONTENT クラスは、1 個の変数の値、もしリンク情報があれば、その詳細も保存するクラスです。

５．６．１　コンストラクタ

（省略）

５．６．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public uint vid	変数 ID
2	public int format	変数 vid の format
3	public int size	変数値のサイズ（データ数）
4	public int count	format L の時に、ここにリンクされる変数の数
5	public TV_CONTENT[] link_list	format=L の時に、ここにリンクされる変数情報を保存する配列リスト（同じクラスに Nesting する）

５．６．３　メソッド

レポート情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。

５．６．３．１　Dispose() - インスタンスの破棄

当該インスタンスのプロパティをすべて消去、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．６．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．アラーム情報関連クラス

アラーム情報は装置からホストに S5F1 メッセージを使って送信されます。

アラーム関連クラスの一覧表を示します。

	クラス名	概要
1	class_Alarm	全アラーム情報管理クラス
2	TAL_INFO	アラーム情報保存クラス
3	TAL_S5F3_INFO	アラームの Enable / Disable (有効/無効)情報リスト
4	TAL_S5F5_INFO	アラーム ID リストクラス
5	TAL_S5F6_INFO	アラーム ID の情報 (ID, ALCD, ALTX)
6	TAL_S5F6_LIST	アラーム ID の情報配列リスト(TAL_S5F6_INFO[])

６．１　class_Alarm - 全アラーム情報管理クラス

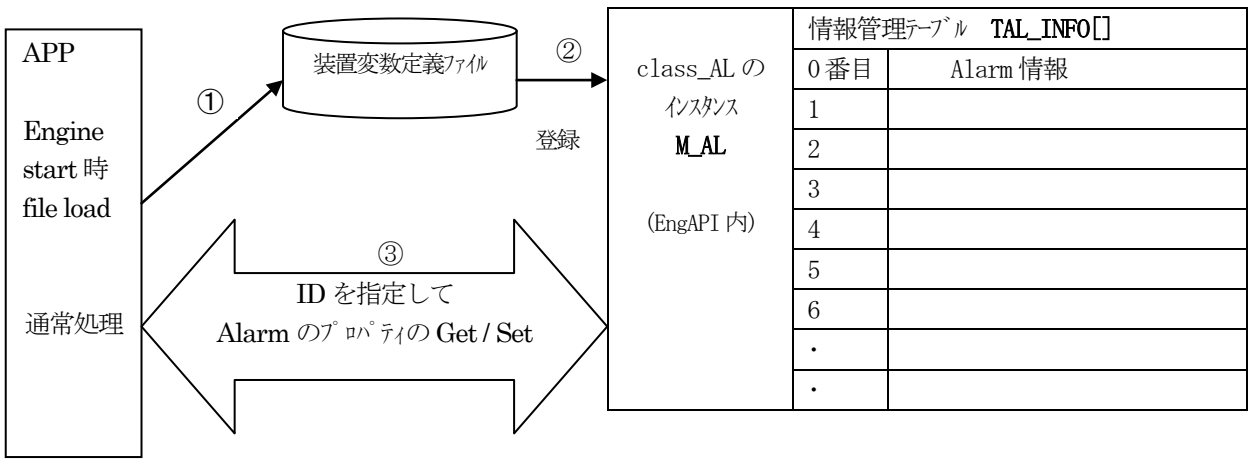
通信エンジンに登録されている全 Alarm 情報を一括して管理するためのクラスです。

DSHEng5 開始時に、装置変数定義ファイル内に定義されているすべてのアラーム ID と情報を本 class_AL クラスのインスタンスに登録します。その class_Alarm のインスタンスが EngAPI. **M_AL** です。

各アラーム ID 情報の保存には TAL_INFO クラスを使用します。(後述します)

APP は本クラスのインスタンスに対しては、ID を指定して各インスタンスのプロパティを参照 (取得) することができます。

アラーム情報の構成と参照については概略以下の通りです。



６．１．１　コンストラクタ

	名前	説明
1	public class_AL(int max_id)	ｲﾝｽﾀﾝｽを生成します。 引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_AL クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)

引数 max_id は管理する ID の最大数を指定します。プロパティ al_info_tab[] の配列サイズになります。

(本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス **M_AL** を生成します。)

６．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1	TAL_INFO[] al_info_tab	アラームの登録テーブル	

TAL_INFO クラスについては、**６．２　TAL_INFO クラス**　の説明を参照ください。

６．１．３　メソッド

通信エンジン内に予め class_AL のインスタンスが準備されています。このインスタンスは、EngAPI クラスの中に M_AL の名前で準備されており、この中にすべてのアラーム情報が登録され、管理されることになります。

APP は、このインスタンス、M_AL に対してメソッドを実行することによって特定 ID のアラーム情報（プロパティ）にアクセスすることができます。
例えば、AL_AlarmTempOver の情報を取得したい場合、次のようなコーディングになります。

```
TAL_INFO info = new TAL_INFO();
int result = EngAPI.M_AL.get (AL_AlarmTempOver, ref info);
```

APP が使用できる class_AL クラスのメソッドは下記一覧表のとおりです。

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int get_name_list()	登録されている ID 名リストを取得します。
4	public int get_id_name_list()	登録されている ID,名前リストを取得します。
5	public int get_id_by_name()	名前からその ID を取得します。
6	public int get_name()	アラーム ID の名前を取得します。
7	public int get()	指定 ID のアラーム情報を取得します。 （TAL_INFO ： アラーム情報保存用ｸﾗｽ）
8	public int get_alcd()	指定 ID の ALCD を取得します。
9	public int get_altx()	指定 ID の ALTX を取得します。
10	public int set_ceid_on()	指定 ID のアラーム発生時に発する CEID を設定します。
11	public int set_ceid_off()	指定 ID のアラーム復旧時に発する CEID を設定します。
12	public int get_ceid_on()	指定 ID のアラーム発生時に発する CEID を取得します。
13	public int get_ceid_off()	指定 ID のアラーム復旧時に発する CEID
14	public int set_enabled()	指定 ID のアラームを有効または無効にします。 （無効にすると S5F1 は往診できない）
15	public int get_enabled()	指定 ID のアラームの有効/無効状態を取得します。

６．１．３．１　get_id_count() - 登録されている ID 数の取得

DSHEng5 内に登録されているアラーム ID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されているアラーム ID 数を取得します。

６．１．３．２　get_id_list() - 登録されている ID リストの取得

DSHEng5 内に登録されているアラーム ID の ID リストを取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大容量

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list リスト内に取得します。

戻り値は、取得した ID 数です。

６．１．３．３　get_iname_list() - 登録されている名前リストの取得

登録されているすべてのアラームの名前をリストに取得します。

【構文】

public int get_name_list(string[] name_buff, int max_buff)

【引数】

name_list

　　アラーム名を保存するリスト

max_size

　　list の最大サイズ

【戻り値】

返却値	意　味
ID 数	取得した ID 数

【説明】

登録されている ID の名前を name_list リストに取得します。
戻り値は、取得した ID 数です。

６．１．３．４　get_id_name_list() - 登録されている ID、名前リストの取得

アラームに登録されているすべての ID とその名前をそれぞれのリストに取得します。

【構文】

public int get_id_name_list(UInt32[] id_list, string[] name_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

name_list

　　アラーム名を保存するリスト

max_size

　　両方の list の最大サイズ

【戻り値】

返却値	意　味
ID 数	取得した ID 数

【説明】

登録されている ID とその名前をそれぞれ id_list, name_list リストに取得します。
戻り値は、取得した ID 数です。

６．１．３．５　get_id_by_name() - アラーム名から ID を取得

登録されているアラーム名からアラーム ID を取得します。

【構文】

```
public int get_id_by_name(string vname, ref UInt32 id)
```

【引数】

vname

　　アラーム名

id

　　ID 格納用領域

【戻り値】

返却値	意　味
0	取得できた。
-1	アラーム名で指定されたアラームはなかった。

【説明】

アラーム名からアラーム ID を取得します。
取得できた場合は 0 を、アラーム名が見つからなかった場合は (-1)を返却します。

６．１．３．６　get_name() - ID からアラーム名を取得

登録されているアラーム ID からアラーム名を取得します。

【構文】

```
public string get_name(UInt32 id)
```

【引数】

id

　　アラーム ID

【戻り値】

返却値	意　味
アラーム名	取得できた。
“”	ID が見つからなかった。

【説明】

ID からアラーム名を取得します。

６．１．３．７　get()- アラーム情報の取得

指定された ID のアラーム情報を取得します。

【構文】

public int get(UInt32 id, ref TAL_INFO info)

【引数】

id

　　取得したいアラーム ID

info

　　取得アラーム情報の参照

【戻り値】

返却値	意　味
0	取得できた。
-1	取得できなかった。

【説明】

id で指定されたアラーム情報を info で指定された TAL_INFO クラスのインスタンスに取得します。

TAL_INFO クラスについては、６．２を参照して下さい。

６．１．３．８　get_alcd()-ALCD の取得

指定された ID の ALCD 値を取得します。

【構文】

public int get_alcd(UInt32 id)

【引数】

id

　　アラーム ID

【戻り値】

返却値	意　味
ALCD 値	ALCD

【説明】

登録されている ID の ALCD 値を取得します。

戻り値が ALCD 値です。

６．１．３．９　get_altx()-ALTX の取得

指定された ID の ALTX 値を取得します。

【構文】

public string get_altx(UInt32 id)

【引数】

id

　　アラーム ID

【戻り値】

返却値	意　味
ALTX 値	アラームテキスト

【説明】

登録されている ID の ALTX 値を取得します。
戻り値が ALTX 値です。

６．１．３．10　set_ceid_on()- アラーム発生時の送信 CEID の設定

指定 ID のアラーム発生時に送信する S6F11 メッセージの CEID を設定します。

【構文】

public int set_ceid_on(UInt32 id, uint ceid_on)

【引数】

id

　　アラーム ID

ceon

　　送信する S6F11 メッセージの CEID

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定された CEID の登録が無かった。

【説明】

指定されたアラーム ID のアラームが発生した際に S6F11 イベント通知に使用する CEID を設定します。

６．１．３．11　set_ceid_off() - アラーム復旧送信 CEID の設定

指定 ID のアラーム復旧時に送信する S6F11 メッセージの CEID を設定します。

【構文】

public int set_ceid_off(UInt32 id, uint ceid_off)

【引数】

id

　　アラーム ID

ceoff

　　送信する S6F11 メッセージの CEID

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定された CEID の登録が無かった。

【説明】

指定されたアラーム ID のアラームが復旧した際に S6F11 イベント通知に使用する CEID を設定します。

６．１．３．12　get_ceid_on() - アラーム発生時の送信 CEID の取得

指定 ID のアラーム発生時に送信する S6F11 メッセージの CEID を取得します。

【構文】

public uint get_ceid_on(UInt32 id)

【引数】

id

　　アラーム ID

【戻り値】

返却値	意　味
ALID	取得できた。
(-1)	設定された CEID が見つからなかった。

【説明】

指定されたアラーム ID のアラームが発生した際に S6F11 イベント通知に使用する CEID を取得します。

６．１．３．13　get_ceid_off()- アラーム復旧送信 CEID の取得

指定 ID のアラーム復旧時に送信する S6F11 メッセージの CEID を取得します。

【構文】

public uint get_ceid_off(UInt32 id)

【引数】

id

アラーム ID

【戻り値】

返却値	意　味
ALID	取得できた。
(-1)	設定された CEID が見つからなかった。

【説明】

指定されたアラーム ID のアラームが復旧した際に S6F11 イベント通知に使用する CEID を取得します。

６．１．３．14　set_enabled()- アラームの有効／無効の設定

指定されたアラーム ID の有効／無効の設定を行います。

【構文】

public int set_enabled(UInt32 id, bool f)

【引数】

id

設定したいアラーム ID です。

f

設定したい enabled の値です。

f = true ： 有効、　f= false ： 無効

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定された ID の CE が無かった。

【説明】

id で指定された CE のアラームを有効か、無効に設定します。

enabled の値は、TAL_INFO クラスの プロパティ名、enabled に設定されます。

有効であれば、その ID のアラームの事象が発生すれば、当該アラーム ID の S5F1 の送信が行われます。

無効に設定されると、当該 ID のアラームの事象が発生しても、S5F1 は送信されないことになります。

６．１．３．15　get_enabled() - アラームの有効／無効の取得

指定されたアラーム ID の有効／無効状態を取得します。

【構文】

public int get_enabled(UInt32 id, ref bool f)

【引数】

id

取得したいアラーム ID です。

f

取得した enabled を保存する領域

f = true ： 有効　f = false ： 無効

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定された ID の CE が無かった。

【説明】

id で指定されたアラームの有効/無効状態を取得します。

６．２　TAL_INFO - アラーム情報保存クラス

アラーム情報を保存するクラスです。

６．２．１　コンストラクタ

TAL_INFO クラスのインスタンスを生成します。

インスタンス al_info を生成する例です。

（1）空のインスタンスを生成

```
TAL_INFO al_info = new TAL_INFO();
```

（2）AL_123 のアラーム ID のインスタンスを生成します。

```
UInt32 AL_123 = 100;
TAL_INFO al_info = new TAL_INFO(AL_123);
```

（注）AL_123 のアラームが登録されていない場合は空のインスタンスを生成します。

６．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public uint alid	アラーム ID
2	public string name	アラーム名
3	public uint alcd	ALCD (Alarm Code)
4	public string altx	ALTX (Alarm Text)
5	public int enabled	アラームが有効か無効 1=有効、0=無効
6	public string ceon_name	CE 名 アラーム発生時に通知する S6F11 の CEID
7	public UInt32 ceid_on	CEID アラーム発生時に通知する S6F11 の CEID
8	public string ceoff_name	CE 名 アラーム復旧時に通知する S6F11 の CEID
9	public UInt32 ceid_on	CEID アラーム復旧時に通知する S6F11 の CEID

６．２．３　メソッド

レポート情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void copy(TAL_INFO src)	他のインスタンスのアラーム情報を当該インスタンスのアラーム情報をコピーします。

６．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．２．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）も開放します。）

６．２．３．３　copy() - TAL_INFO の情報コピー

TAL_INFO の情報を当該 TAL_INFO クラスのインスタンスにコピーします。

【構文】

public int copy(TAL_INFO src_info)

【引数】

src_info

コピー元の TAL_INFO クラスのインスタンス

[戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src_ info が null であった。

【説明】

当該インスタンスが他のインスタンスからのコピーに使用します。

６．３　TAL_S5F3_INFO - アラーム有効/無効設定情報保存クラス

アラームの有効／無効(Enabled)リスト情報を保存するクラスです。
S5F3 メッセージ情報のデコード結果を保存します。

６．３．１　コンストラクタ

TAL_S5F3_INFO クラスのインスタンスを生成します。

６．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int enabled	Enable 状態
2	public int count	アラーム ID 数
3	public UInt32[] alid_list	アラーム ID リスト

６．３．３　メソッド

レポート情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void add_id()	アラーム ID リストに１個アラーム ID を追加します。
4	public void copy(TAL_INFO src)	他のインスタンスのアラーム情報を当該インスタンスのアラーム情報をコピーします。

６．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）も開放します。）

６．４．３．３　add()－アラーム ID を追加

アラーム ID 配列リストにアラーム ID を１個追加します。

【構文】

public int add(UInt32 alid)

【引数】

alid

追加するアラーム ID

【戻り値】

追加した後のプロパティ count の値を返却します。

【説明】

アラーム ID 配列リスト、プロパティ alid_list に引数 alid を追加し、プロパティ count　+１します。
追加後、配列リストに保存されているアラーム ID 数を返却します。

６．３．３．４　copy()－TAL_S5F3_INFO の情報コピー

TAL_S5F3_INFO の情報を当該 TAL_S5F3_INFO クラスのインスタンスにコピーします。

【構文】

public int copy(TAL_S5F3_INFO src_info)
public static int copy(ref TAL_S5F3_INFO dst, TAL_S5F3_INFO src)

【引数】

dst_info

コピー先の TAL_S5F3_INFO クラスのインスタンス

src_info

コピー元の TAL_S5F3_INFO クラスのインスタンス

[戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src_ info が null であった。

【説明】

TAL_S5F3_INFO クラスのインスタンスを別の TAL_S5F3_INFO のインスタンスにコピーします。

引数として dst_info が指定されない場合は、当該インスタンスがコピー先になります。
引数 dst_info が指定された場合は、static のメソッドですので注意ください。

６．４　TAL_S5F6_INFO - アラーム情報保存クラス

アラームの ID, ALCD, ALTX を保存するクラスです。
S5F6 メッセージ情報のデコード結果を保存します。

６．４．１　コンストラクタ

TAL_S5F6_INFO クラスのインスタンスを生成します。

６．４．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public uint alcd	Alarm Code
2	public UInt32 alid	アラーム ID 数
3	public string altx	Alarm Text

６．４．３　メソッド

レポート情報クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。

６．４．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをclear()メソッドによってすべてクリアします。
（プロパティが使用している資源（非管理メモリ）もシステムに返却します。）
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．４．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）も開放します。）

６．５　TAL_S5F6_LIST - アラーム情報保存リストクラス

複数のアラームの ID, ALCD, ALTX を保存するクラスです。
S5F6 メッセージ情報のデコード結果を保存します。

６．５．１　コンストラクタ

TAL_S5F6_LIST クラスのインスタンスを生成します。

６．５．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	保存アラーム情報の数
2	public TAL_S5F6_INFO[] list	アラーム情報のリスト

６．５．３　メソッド

TFPP_PPARA のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。

６．５．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【引数】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
Dispose()の後、このインスタンスを使用することはできません。

６．５．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。
（プロパティが使用している資源（非管理メモリ）も開放します。