DSHGEM-CLASS　GEM 通信エンジン・クラスライブラリ

ソフトウェア・パッケージ

クラス・ライブラリ説明書

Vol-１　エンジン起動と管理情報クラス 編

Part-２　第１４章　～　２１章

２０１１年2月（改－６）

株式会社データマップ

文書番号　DSHGEM-07-30361-00-ClassLib-Info-2

[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.	2010 年 1 月	初版	
2.	2010/3/4	誤字など訂正	14.2.2 DshCjCmd のﾌﾟﾛﾊﾟﾃｨ表の 2 番目の行削除 16.3.2 DshCarAction ﾌﾟﾛﾊﾟﾃｨ表の番号を訂正 17 端末表示 Msg ｸﾗｽの表の 2 DshTermMultiMsg の説明を訂正
3	2011/02/07	ｸﾗｽに Dispose ﾒｿｯﾄﾞﾄｸｸﾗｽのﾄﾚｰｽﾓﾆﾀｰと表示機能を実装した。	ｸﾗｽに Dispose ﾒｿｯﾄﾞを追加した。 また、ﾃﾞﾊﾞｯｸﾞ機能強化のためﾕｰｻﾞがｸﾗｽの生成、消滅の回数を管理し、それらのﾀｲﾐﾝｸﾞをﾄﾚｰｽ表示できるようにした。
4.	2011/02/07	DshDebug ｸﾗｽに class trace 表示機能を追加	21. ｸﾗｽ･ﾗｲﾌﾞﾗﾘの DshDebug ｸﾗｽにｸﾗｽのﾄﾚｰｽ表示機能を追加した。。

目次

14. コントロール・ジョブ関連クラス

コントロールジョブ(CJ)関連情報を保存するクラスです。

関連クラスの一覧を次表に示します。

	クラス名	用途
1	DshCj	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ情報保存ｸﾗｽです。 S14F9
2	DshCjCmd	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞｺﾏﾝﾄﾞ情報保存用ｸﾗｽです。 S16F27
3	DshCjAttr	ｵﾌﾞｼﾞｪｸﾄの情報を保存するためのｸﾗｽです。 ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞｸﾗｽ内では配列で使用します。
4	DshCjAttrList	DshCjAttr ｸﾗｽの配列ﾘｽﾄｸﾗｽです。
5	DshStatusPara	ｽﾃｰﾀｽﾊﾟﾗﾒｰﾀ値を保存するためのｸﾗｽです
6	DshStatusParaList	DshStatusPara ｸﾗｽの配列ｸﾗｽです。
7	DshMaterialOutStatus	MtrlOutByStatus 属性情報を保存するｸﾗｽです。
8	DshMaterialOutSpec	MtrlOutSpec 属性情報を保存するためのｸﾗｽです。
9	DshCjCarSlotList	ｷｬﾘｱのｽﾛｯﾄ ID を配列で保存するためのｸﾗｽです。
10	DshProcControlSpec	ProccessingCtrlSpec 属性情報を保存するためのｸﾗｽです。
11	DshControlRule	ProccessingCtrlSpec 属性のｺﾝﾄﾛｰﾙﾙｰﾙ情報を保存するためのｸﾗｽです
12	DshOutputRule	ProccessingCtrlSpec 属性のｱｳﾄﾌﾟｯﾄﾙｰﾙ情報を保存するためのｸﾗｽです。
13	DshTextList	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞの属性で、ﾃｷｽﾄ配列情報を保存するためのｸﾗｽです。

CJ の通信関連クラスとして、以下のものがあります。 これらについては、Vol-2 で説明します。

DshS14F9Send
DshS14F10Response
DshS14F11Send
DshS14F12Response
DshS16F27Send
DshS16F28Response

14. 1 DshCj クラス

コントロールジョブ情報を保存するために使用されます。
通常、S14F9 メッセージを通して装置、ホスト間でやり取りされる情報です。

14. 1. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshCj()	装置 ID=0 のｲﾝｽﾀﾝｽを生成します。 （後で、set_id ﾒｿｯﾄﾞでｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ ID(cjid)を指定します）
2	public DshCj(string cjid)	装置 ID=0 のｲﾝｽﾀﾝｽをｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ ID を指定して生成します。
3	public DshCj(int eqid)	装置 ID を指定してｲﾝｽﾀﾝｽを生成します。 （後で、set_id ﾒｿｯﾄﾞでｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ ID を指定します）
4	public DshCj(int eqid, cjid)	装置 ID と cjid を指定してｲﾝｽﾀﾝｽを生成します。

14. 1. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int eqid	装置 ID、ﾃﾞﾌｫﾙﾄ値 = 0 です。
2	public string cjid	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ ID です。 ｴﾝｼﾞﾝはこの ID で管理します。
3	public string name	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ名です。cjid と同じ値です。
4	public string objspec	ｵﾌﾞｼﾞｪｸﾄ仕様文字列です。
5	public string objtype	ｵﾌﾞｼﾞｪｸﾄのｸﾞﾙｰﾌﾟまたはｸﾗｽの ID です。
6	public int attr_count	ｵﾌﾞｼﾞｪｸﾄ属性情報の数です。
7	public DshCjAttr[] attr_list	ｵﾌﾞｼﾞｪｸﾄ属性情報の配列ﾘｽﾄです。 DshObjClass ｸﾗｽの配列です。

14. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_id()	cjid ﾌﾟﾛﾊﾟﾃｨに ID を設定します。
2	public void init_set()	cjid, objspec, objtype を設定します。
3	public void add_attr()	DshCjAttr ｸﾗｽに保存されている属性情報を attr_list に追加します。
4	public int alloc_id()	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ ID をｴﾝｼﾞﾝに登録します。
5	public int set()	ｲﾝｽﾀﾝｽ内に設定されたｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ情報をｴﾝｼﾞﾝに設定します。
6	public int get()	ｴﾝｼﾞﾝからｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ情報を取得します。
7	public void delete()	cjid に指定されたｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ情報をｴﾝｼﾞﾝの管理情報から削除します。
8	public int decode()	S14F9 ﾒｯｾｰｼﾞを当該ｸﾗｽにﾃﾞｺｰﾄﾞします。
9	public void copy()	DshCj ｸﾗｽの当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
10	public int get_id_count()	ｴﾝｼﾞﾝに登録されているｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ ID 情報の合計数を取得します。
11	public int get_id_list()	ｴﾝｼﾞﾝに登録されている全ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ ID をﾘｽﾄに取得します。
12	public void clear()	attrl_list 配列の内容を消去します。 attr_count=0 にします。
13	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。**

14. 1. 3. 1 set_id()

コントロールジョブ ID を設定します。
コントロールジョブ ID は、エンジンに登録し管理するための ID です。S14F9 などメッセージ処理に使用されます。

【構文】
public void set_id(string cjid)

【引数】
cjid
　　設定するコントロールジョブ ID です。

【戻り値】
なし。

【説明】
引数に与えられた cjid の値をインスタンス内の cjid に設定します。

14. 1. 3. 2 init_set()

インスタンスにコントロールジョブの３つのプロパティ値を設定します。

【構文】
public void init_set(string cjid, string objspec, string objtype)

【引数】
cjid
　　コントロールジョブ ID です。
objspce
　　オブジェクト仕様文字列です。
objtype
　　オブジェクトのグループまたはクラスの ID です。

【戻り値】
なし。

【説明】
引数に与えられた値を、それぞれインスタンス内のプロパティの値として設定します。

14. 1. 3. 3 add_attr()

インスタンスの attr_list 配列に属性情報を 1 個追加します。

【構文】

public void add_attr(ref DshCjAttr attr)

【引数】

attr

属性情報が保存されている DshCjAttr クラスのインスタンスです。

【戻り値】

なし。

【説明】

attr_list 配列に 1 個の属性情報を追加します(attr の内容を attr_list[attr_count]にコピーします。)
その後、attr_count + 1 します。

14. 1. 3. 4 alloc_id()

コントロールジョブ ID をエンジンに新規登録します。

【構文】

public int alloc_id()
public int alloc_id(string cjid)

【引数】

cjid

登録するコントロールジョブ ID です。

【戻り値】

返却値	意　味
0 or 1	正常に登録できた。(1 は既に登録済みであったことを意味します)
(-1)	登録に失敗した。

【説明】

引数として cjid が指定された場合は、そのコントロールジョブ ID をエンジンに登録します。

登録できた場合は、引数の cjid がインスタンス内の cjid に設定されます。
引数が無い場合は、インスタンス内の cjid のコントロールジョブ ID をエンジンに登録します。
正常に登録できた場合は、0 または 1 を返却します。失敗した場合は、(-1)を返します。
既に ID が登録されていた場合、エンジンは管理情報の中から ID 以外の情報をクリアします。

14. 1. 3. 5 set()

コントロールジョブ情報をエンジンに設定します。

【構文】

```
public int  set()
```

【引数】

なし。

【戻り値】

返却値	意　味
0	正常に設定できた。
(-1)	設定に失敗した。

【説明】

インスタンス内に設定されたコントロールジョブ情報をエンジンの管理情報に設定します。

14. 1. 3. 6. get()

エンジンからコントロールジョブ情報を取得します。

【構文】

```
public int get()
public int get( string cjid)
```

【引数】

cjid

情報を取得したいコントロールジョブ ID です。

【戻り値】

返却値	意　味
0	正常に取得できた。
(-1)	取得に失敗した。

【説明】

エンジンの管理情報からコントロールジョブ情報を取得します。
引数で cjid が指定された場合は、そのコントロールジョブの情報を取得します。

引数が無い場合は、インスタンス内の cjid のコントロールジョブの情報を取得します。

正常に取得できた場合は、0 を、失敗した場合は(-1)を返却します。

14. 1. 3. 7 delete()

指定されたコントロールジョブの情報をエンジン管理情報の中から削除します。

【構文】

public int delete()
public int delete(string cjid)

【引数】

cjid

削除したいコントロールジョブの ID です。

【戻り値】

返却値	意　味
0	正常に削除できた。
1	ｴﾝｼﾞﾝに登録されていなかった。
(-1)	削除できなかった。(cjid が無効であった)

【説明】

1 個のコントロールジョブ情報をエンジンの管理情報から削除します。

削除対象は、eqid で指定された装置の情報であり、引数にコントロールジョブ ID の指定があるメソッドでは、引数で指定されたコントロールジョブを、そして、引数が無いメソッドの場合は、インスタンスの cjid プロパティのコントロールジョブを削除します。

一旦削除された情報は、他のメソッドでは復元できませんので注意してください。

14. 1. 3. 8 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshCj dst)

【引数】

dst

コピー先の DshCj インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

14．1．3．9　decode()

S14F9 に含まれる情報を DshCj クラス内のプロパティにデコードします。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

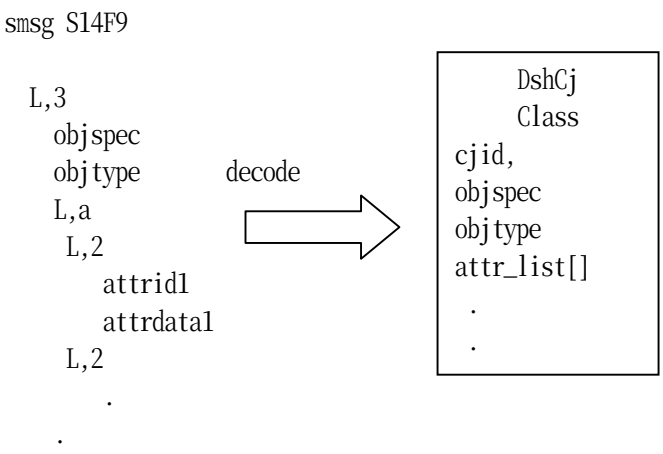
S14F9 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった（S14F9 のﾒｯｾｰｼﾞの形式が正しくなかった）

【説明】

smsg に含まれているコントロールジョブ情報を DshCj クラス内にデコードします。



メッセージに含まれるコントロールジョブ情報(cjid,属性情報等)は、クラスのプロパティに保存されます。

正常にデコードできた場合、0 を返却します。

もし、S14F9 のメッセージフォーマットが正しくないなどの理由でデコードできなかった場合、 (-1)を返却します。

14. 1. 3. 10 get_id_count()

エンジンの当該装置に登録されているコントロールジョブ ID の合計数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

登録されているコントロールジョブ ID 数が返却されます。

【説明】

登録されているコントロールジョブ ID の合計数を取得します。

14. 1. 3. 11 get_id_list()

エンジンの当該装置に登録されている全コントロールジョブの ID と名前のリストを取得します。

【構文】

public int get_id_list(string[] id_list, string[] n, name_list, int list_size)

【引数】

id_list

コントロールジョブ ID を格納する配列です。

n, name_list

コントロールジョブ名を格納する配列（name は id と同じになります）です。

list_size

準備されたリストの配列サイズです。

【戻り値】

取得できたコントロールジョブ ID の数が返却されます。

【説明】

エンジンに登録されている全コントロールジョブ ID とその名前をそれぞれ id_list[], name_list[]に取得します。

list_size は配列のサイズを指定します。

配列は、登録されている全コントロールジョブ ID を保存できる充分のサイズの配列を準備してください。

返却値は取得できたコントロールジョブ ID の合計数です。

（注）ID と名前格納用リストのサイズは。先に説明した get_id_count()メソッドで合計 ID 数を取得し、
そのサイズの配列を準備して本メソッドを使用すると便利です。

14. 1. 3. 12 clear()

インスタンス内の attr_list 配列に含まれる属性情報を全て消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

attr_list 配列の中の attr_count 分の情報を消去し、配列を空にします。そして attr_count=0 にします。
具体的には、各配列の DshCjAttr クラスのインスタンスの消去です。それらの中に使用されている非管理メモリを開放します。

14. 1. 3. 13 Dispose()

本クラスが使用済になった際、クラス内で使用していて、開放すべきメモリがあれば、それを開放し、また Dispose すべきオブジェクトがあれば、それらを Dispose します。

【構文】

public void Dispose()

【引数】

なし。

【戻り値】

なし。

【説明】

生成された後、当該クラスが使用済みになった時点で、ユーザプログラムが明示的に使用していた資源を解放するためのメソッドです。

本メソッドが実行されるとシステムから本クラスに対する Finalizer は呼び出されません。

14. 2 DshCjCmd クラス

コントロールジョブコマンド情報を保存するためのクラスです。本クラスは DshObjPara の派生クラスです。

本クラスは次のメッセージの送信または受信処理のために使用されます。

S16F27 メッセージ

14. 2. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshCjCmd()	空のｲﾝｽﾀﾝｽを生成します。 ﾌﾟﾛﾊﾟﾃｨ値は set_cmd_info()ﾒｿｯﾄﾞを使って設定します。
	public DshCjCmd(string cjid, int cmd)	ﾌﾟﾛｾｽｼﾞｮﾌﾞ ID とｺﾏﾝﾄﾞを指定してｲﾝｽﾀﾝｽを生成します。 他のﾌﾟﾛﾊﾟﾃｨ値は set_cmd_info()ﾒｿｯﾄﾞを使って設定します。
	public DshCjCmd(string cjid, int cmd, string cpname, int fmt, int size, IntPtr cpval)	cjid, cmd, ﾊﾟﾗﾒｰﾀ名、ﾌｫｰﾏｯﾄ、値格納ﾎﾟｲﾝﾀを指定して ｲﾝｽﾀﾝｽを生成します。 全ﾌﾟﾛﾊﾟﾃｨを指定します。
	public DshCjCmd(string cjid, int cmd, string cpname, string cpval)	cjid, cmd, ﾊﾟﾗﾒｰﾀ名、文字列ﾀｲﾌﾟのﾊﾟﾗﾒｰﾀ値を指定して ｲﾝｽﾀﾝｽを生成します。 全ﾌﾟﾛﾊﾟﾃｨを指定します。(HSMS.ICODE_A ﾌｫｰﾏｯﾄ)

14. 2. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string cjid	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ ID です。
2	public int cmd	ｺﾏﾝﾄﾞ文字列です。
3	public string name	DshObjPara の name ﾒﾝﾊﾞｰです。 ﾊﾟﾗﾒｰﾀ名です。
4	public int format	DshObjPara の format ﾒﾝﾊﾞｰです。 ﾊﾟﾗﾒｰﾀ値のﾌｫｰﾏｯﾄです。(HSMS.ICODE_A など)
5	public int size	DshObjPara の size ﾒﾝﾊﾞｰです。 ﾊﾟﾗﾒｰﾀ値の配列ｻｲｽﾞです。(HSMS.ICODE_A 以外は=1)
6	public IntPtr value	DshObjPara の value ﾒﾝﾊﾞｰです。 ﾊﾟﾗﾒｰﾀ値が保存されているﾎﾟｲﾝﾀです。
7	public int para_flag	ﾊﾟﾗﾒｰﾀの設定状態のﾌﾗｸﾞです。 ﾕｰｻﾞの処理には関係ありません。

14. 2. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_cmd_info()	cjid、cmd とｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ を設定します。
2	public int decode()	S16F27 ﾒｯｾｰｼﾞ を当該ｸﾗｽにﾃﾞｺｰﾄﾞ します。
3	public void free()	DshObjPara ｸﾗｽの value に使用した非管理ﾒﾓﾘを解放します。 （DshObjPara のﾒｿｯﾄﾞです。そちらを参照ください。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14．2．3．1　set_cmd_info()

コントロールジョブ ID とコマンドをインスタンス内に設定します。

【構文】

```
public void set_cmd_info( string cjid, int cmd)
public void set_cmd_info(string cjid, int cmd,
                         string cpname, int format, int size, IntPtr cpval)
public void set_cmd_info(string cjid, int cmd, string cpname, string cpval)
```

【引数】

cjid
　　コントロールジョブ ID です。
cmd
　　コントロールジョブに与えるコマンドです。
cpname
　　パラメータ名です。
format
　　パラメータ値のフォーマットです。(HSMS.ICODE_U1 など)
size
　　パラメータ値の配列サイズです。HSMS.ICODE_A のとき、文字列長になります。
　　その他のフォーマットでは =1 になります。
cpval
　　パラメータ値が格納されているポインタです。
　　string の場合は文字列です。

【戻り値】

なし。

【説明】

コントロールジョブ ID、cjid とコマンド、cmd をそれぞれのプロパティに設定します。
また、パラメータの引数が指定されているものについては、それぞれ、DshObjPara クラスのインスタンスに設定します。

14．2．3．2　decode()

S16F27 メッセージに含まれる情報を DshCjCmd クラス内のプロパティにデコードします。
本メソッドは S16F27 メッセージ受信処理時に使用することができます。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

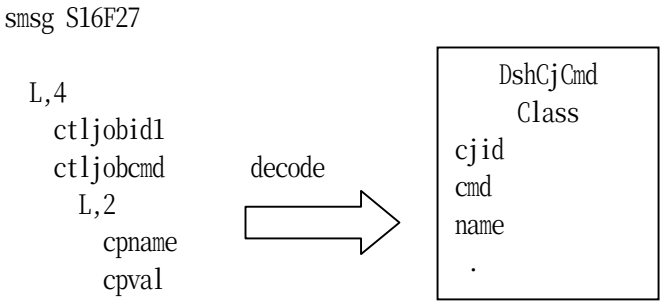
S16F27 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった（S16F27 のﾒｯｾｰｼﾞの形式が正しくなかった）

【説明】

smsg に含まれている情報を DshCjCmd クラス内にデコードします。



14. 3 DshCjAttr クラス

S14F9 コントロールジョブ送信メッセージに含まれる 1 個の属性情報を保存するためのクラスです。

14. 3. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshCjAttr()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshCjAttr(string attrid)	属性 ID を指定してｲﾝｽﾀﾝｽを生成します。 （属性ｲﾝﾃﾞｸｽの値を変換して設定します。）
3	public DshCjAttr(int attr_index)	属性 ID ｲﾝﾃﾞｸｽを指定してｲﾝｽﾀﾝｽを生成します。 attrid の値も取得し設定します。

14. 3. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string attrid	属性 ID を保存します。(下の表参照)
2	public int attr_index	属性 ID をｲﾝﾃﾞｸｽに変換したものです。 dsh_const ｸﾗｽに定義されています。
3	public int num_data	整数の属性値を保存します。
4	public string str_data	文字列の属性値を保存します。
5	public int status_count	DshMaterialOutStatus が保存している属性値の数です。
6	public DshMaterialOutStatus[] mtrl_out_status	DshMaterialOutStatus ｸﾗｽの配列です。 MtrlOutByStatus 属性の情報を保存します。
7	public int spec_count	DshMaterialOutSpec が保存している属性値の数です。
8	public DshMaterialOutSpec[] mtrl_out_spec	DshMaterialOutSpec ｸﾗｽの配列です。 MtrlOutSpec 属性の情報を保存します。
9	public int proc_count	DshProcControlSpec が保存している属性値の数です。
10	public DshProcControlSpec[] proc_control_spec	DshProcControlSpec ｸﾗｽの配列です。 ProcessingCtrlSpec 属性の情報を保存します。
11	public DshPrjStateList prj_state_list	PRJobStatusList 属性の情報を保存します。
12	public DshPauseEvent pause_event	PauseEvent 属性の情報を保存します。
13	public DshTextList text_list	ﾃｷｽﾄ配列の属性の情報を保存します。

コントロールジョブ属性インデクス値 / 属性 ID

index 値	定数名 – attr_index	属性 ID – attrid
0	EN_ObjID	“ObjID”
1	EN_CarrierInputSpec	“CarrierInputSpec”
2	EN_CurrentPRJob	“CurrentPRJob”
3	EN_DataCollectionPlan	“DataCollectionPlan”
4	EN_MtrlOutByStatus	“MtrlOutByStatus”
5	EN_MtrlOutSpec	“MtrlOutSpec”
6	EN_PauseEvent	“PauseEvent”
7	EN_ProcessingCtrlSpec	“ProcessingCtrlSpec”
8	EN_ProcessingOrderMgmt	“ProcessingOrderMgmt”
9	EN_PRJobStatusList	“PRJobStatusList”
10	EN_StartMethod	“StartMethod”
11	EN_State	“ State ”

14. 3. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	mtrl_out_status, mtrl_out_spec, proc_control_spec, prj_state_list, pause_event, text_list の内容を消去し、空にします。 そしてそれぞれの配列の count を 0 にします。
2	public void set_string()	文字列情報を設定します。
3	public void set_num()	整数情報を設定します。
4	public void add_MaterialOutStatus()	MtrlOutByStatus 属性の情報を追加します。 mtrl_out_status 配列に追加します。
5	public void add_slot()	DshMaterialOutStatus の car_slot ﾒﾝﾊﾞｰに slotid を１個追加します。
6	public void add_MaterialOutSpec()	MtrlOutSpec 属性の情報を追加します。 mtrl_out_spec 配列に追加します。
7	public void add_src_slot()	DshMaterialOutSpec の src_car_slot ﾒﾝﾊﾞｰに slotid を１個追加します。
8	public void add_dst_slot()	DshMaterialOutSpec の dst_car_slot ﾒﾝﾊﾞｰに slotid を１個追加します。
9	public void add_prj_proc_count()	ProcessingCtrlSpec 属性の情報として proc_control_spec 配列に prjid を指定して DshProcControlSpec ｲﾝｽﾀﾝｽを１個追加します。
10	public int init_prj_proc_count()	ProcessingCtrlSpec 属性の情報として proc_control_spec 配列の指定位置に prjid を指定して DshProcControlSpec ｲﾝｽﾀﾝｽを１個生成します。
11	public void init_PrjStateList()	prj_state_list に DshPrjStateList ｸﾗｽのｲﾝｽﾀﾝｽを生成します。
12	public void init_PauseEvent()	pause_event に DshPauseEvent ｸﾗｽのｲﾝｽﾀﾝｽを生成します。
13	public void init_Text()	text_list に DshTextList ｸﾗｽのｲﾝｽﾀﾝｽを生成します。
14	public int set_string_attr()	attr_index から属性 ID を求め、attrid に設定した上で、文字列ﾃﾞｰﾀを str_data に設定します。
15	public int set_num_attr()	attr_index から属性 ID を求め、attrid に設定した上で、文字列ﾃﾞｰﾀを str_data に設定します。
16	public static string get_attrid_from_index()	attr_index から属性 ID を求めます。
17	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
18	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 3. 3. 1 clear()

クラス内のプロパティの内容を消去します。

【構文】
public void clear()

【引数】
なし

【戻り値】
なし。

【説明】
mtrl_out_status, mtrl_out_spec, proc_control_spec, prj_state_list, pause_event, text_list の内容を消去し、空にします。
そしてそれぞれの配列の count を 0 にします。

14. 3. 3. 2 set_string()

文字列データを設定します。

【構文】
public void set_string(string data)

【引数】
data
　　文字列データです。

【戻り値】
なし。

【説明】
data に与えられた文字列を str_data プロパティに設定します。

14. 3. 3. 3 set_num()

整数データを設定します。

【構文】

public void set_num(int data)

【引数】

data

整数データです。

【戻り値】

なし。

【説明】

data に与えられた整数を num_data プロパティに設定します。

14. 3. 3. 4 add_MaterialOutStatus()

MtrlOutByStatus 属性のクラス DshMaterialOutStatus の配列 mtrl_out_status に１個の属性情報を追加します。
DshMaterialOutStatus クラスについては後述します。

【構文】

public void add_MaterialOutStatus(string carid, int status)

【引数】

carid

キャリア ID です。

status

キャリアの状態です。

【戻り値】

なし。

【説明】

mtrl_out_status 配列の status_count 位置に DshMaterialOutStatus クラスのインスタンスを生成し、そこに carid, status を設定します。その後、status_count + 1 します。

14. 3. 3. 5 add_slot()

mtrl_out_status の指定配列位置にスロット ID を追加します。

【構文】

public int add_slot(int index, int slotid)

【引数】

index

配列位置です。

slotid

設定するスロット ID です。

【戻り値】

返却値	意　味
0	正常に追加できた。
(-1)	追加できなかった。(index >=status_count であった。ｲﾝｽﾀﾝｽが未生成)

【説明】

mtrl_out_status の指定配列位置のインスタンスの car_slot メンバーにスロット ID を１個追加します。
car_slot については、後述する DshCjCarSlotList クラスの説明を参照してください。

14. 3. 3. 6 add_MaterialOutSpec()

MtrlOutSpec 属性のクラス DshMaterialOutSpec の配列 mtrl_out_spec に１個の属性情報を追加します。
DshMaterialOutSpec クラスについては後述します。

【構文】

public void add_MaterialOutSpec(string src_carid, string dst_carid)

【引数】

src_carid

ソースのキャリア ID です。

dst_carid

行き先のキャリア ID です。

【戻り値】

なし。

【説明】

mtrl_out_spec 配列の spec_count 位置に DshMaterialOutSpec クラスのインスタンスを生成し、そこに src_carid, dst_carid を設定します。その後、spec_count + 1 します。

14. 3. 3. 7 add_src_slot()

mtrl_out_spec の指定配列位置インスタンスの src_car_slot にスロット ID を追加します。

【構文】

public int add_src_slot(int index, int slotid)

【引数】

index

配列位置です。

slotid

設定するスロット ID です。

【戻り値】

返却値	意　味
0	正常に追加できた。
(-1)	追加できなかった。(index >=spec_count であった。ｲﾝｽﾀﾝｽが未生成)

【説明】

mtrl_out_spec の指定配列位置のインスタンスの src_car_slot メンバーにスロット ID を１個追加します。
src_car_slot については､後述する DshMaterialOutSpec とDshCjCarSlotList クラスの説明を参照してください。

14. 3. 3. 8 add_dst_slot()

mtrl_out_spec の指定配列位置インスタンスの dst_car_slot にスロット ID を追加します。

【構文】

public int add_dst_slot(int index, int slotid)

【引数】

index

配列位置です。

slotid

設定するスロット ID です。

【戻り値】

返却値	意　味
0	正常に追加できた。
(-1)	追加できなかった。(index >=spec_count であった。ｲﾝｽﾀﾝｽが未生成)

【説明】

mtrl_out_spec の指定配列位置のインスタンスの dst_car_slot メンバーにスロット ID を１個追加します。
dst_car_slot については､後述する DshMaterialOutSpec とDshCjCarSlotList クラスの説明を参照してください。

14. 3. 3. 9 add_prj_proc()

ProcessingCtrlSpec 属性のクラス DshProcControlSpec の配列 proc_control_spec に１個の属性情報を追加します。
DshProcControlSpec クラスについては後述します。

【構文】

public void add_prj_proc(string prjid)

【引数】

prjid

プロセスジョブ ID です。

【戻り値】

なし。

【説明】

proc_control_spec 配列の proc_count 位置に DshProcControlSpec クラスのインスタンスを生成し prjobid プロパティに prjid を設定します。その後、proc_count + 1 します。

14. 3. 3. 10 init_prj_proc()

proc_control_spec の指定配列位置インスタンスにプロセスジョブ ID を設定します。

【構文】

public int init_prj_proc(int index, string prjid)

【引数】

index

配列位置です。

prjid

設定するプロセスジョブ ID です。

【戻り値】

返却値	意　味
0	正常に追加できた。
(-1)	追加できなかった。(index >=proc_count であった。ｲﾝｽﾀﾝｽが未生成)

【説明】

proc_control_spec の指定配列位置のインスタンスの prjobid プロパティに 引数で与えられた prjid を設定します。

後述する DshProcControlSpec クラスの説明を参照してください。

14. 3. 3. 11 init_PrjStateList()

PRJobStatusList 属性のクラス DshPrjStateList のインスタンスをに prj_state_list に生成します。
DshPrjStateList クラスについては後述します。

【構文】
public void init_PrjStateList()

【引数】
なし。

【戻り値】
なし。

【説明】
prj_state_list に DshPrjStateList クラスのインスタンスを生成します。

14. 3. 3. 12 init_PauseEvent()

PauseEvent 属性のクラス DshPauseEven のインスタンスをに pause_event に生成します。
DshPauseEven クラスについては後述します。

【構文】
public void init_PauseEvent()

【引数】
なし。

【戻り値】
なし。

【説明】
pause_event に DshPauseEven クラスのインスタンスを生成します。

14. 3. 3. 13 init_Text()

複数のテキスト（文字列）を保存する属性のクラス DshTextList のインスタンスをに text_list に生成します。
DshTextList クラスについては後述します。

【構文】
public void init_Text()

【引数】
なし。

【戻り値】
なし。

【説明】
text_list に DshTextList クラスのインスタンスを生成します。

14. 3. 3. 14 set_string_attr()

文字列属性に対して文字列を設定します。

【構文】
public void set_string_attr(int attr_index, string value)
public void set_string_attr(string value)

【引数】
attr_index
　　属性 ID のインデクスです。
value
　　文字列データです。

【戻り値】
なし。

【説明】
str_data に value の文字列データを設定します。
attr_index の引数が指定された場合は、attr_index と attrid も設定します。

14. 3. 3. 15 set_num_attr()

数値属性に対して整数値を設定します。

【構文】

public void set_num_attr(int attr_index, int value)
public void set_num_attr(int value)

【引数】

attr_index
　　属性 ID のインデクスです。
value
　　整数データです。

【戻り値】

なし。

【説明】

num_data に value の数値データを設定します。
attr_index の引数が指定された場合は、attr_index と attrid も設定します。

14. 3. 3. 16 get_attrid_from_index()

属性インデクスから属性 ID（文字列）を取得します。

【構文】

public static string get_attrid_from_index(int index)

【引数】

attr_index
　　属性 ID のインデクスです。

【戻り値】

属性 ID が返却されます。
もし、attr_index が未定義の場合は空の文字列が返却されます。

【説明】

属性インデクスから属性 ID（文字列）を取得します。

14. 3. 3. 17 copy()

当該インスタンスの内容を別の DshCjAttr クラスのインスタンスにコピーします。

【構文】

public void copy(ref DshCjAttr dst)

【引数】

dst

コピー先インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンス内に保存されているプロパティ情報を dst で指定されたインスタンスにコピーします。

14. 4 DshCjAttrList クラス

複数のコントロールジョブの属性を保存するための配列クラスです。

14. 4. 1 コンストラクタ

	名前	説明
1	public DshCjAttrList()	空のｲﾝｽﾀﾝｽを生成します。

14. 4. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	list に保存されている属性数です。
2	public DshCjAttr[] list	属性情報保存配列です。 DshCjAttr ｸﾗｽの配列です。

14. 4. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	value に使用されているﾒﾓﾘを開放します。
2	public void add_attr()	パラメータ情報を設定します。
3	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14．4．3．1　clear()

list に保存されている属性情報を消去します。

【構文】
public void clear()

【引数】
なし

【戻り値】
なし。

【説明】
list に保存されている DshCjAttr 属性情報を消去し、配列を空にします。
そして、count=0 にします。

14．4．3．2　add_attr()

list 配列に属性情報を１個追加します。

【構文】
public void add_attr(ref DshCjAttr attr)

【引数】
attr
　　追加したい DshCjAttr クラスのインスタンスです。

【戻り値】
なし。

【説明】
list 配列の count 位置に DshCjAttr のインスタンスを生成し、引数 attr のインスタンスの内容をコピーします。
そして、 count+1 します。

14. 5 DshStatusPara クラス

１個のステータスパラメータ値を保存するためのクラスです。

ステータスパラメータのデータタイプには、HSMS クラスで定義されている SECS-II メッセージに使用される様々なものがありますが、それらを非管理メモリ領域に保存します。

14. 5. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshStatusPara()	空のｲﾝｽﾀﾝｽを生成します。 値はﾒｿｯﾄﾞで設定します。
2	public DshStatusPara(int status, int format, int size, IntPtr value)	引数として status, ﾌｫｰﾏｯﾄ、size, 値が格納されているﾎﾟｲﾝﾀ(IntPtr)が指定されます。

14. 5. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int status	ｽﾃｰﾀｽﾊﾟﾗﾒｰﾀ ID です。
2	public int format	ｽﾃｰﾀｽﾊﾟﾗﾒｰﾀのﾌｫｰﾏｯﾄです。 (HSMS.ICODE_U1　など)
3	public int size	ｽﾃｰﾀｽﾊﾟﾗﾒｰﾀの配列ｻｲｽﾞ（ﾃﾞｰﾀ数、基本的に ICODE_A 以外は size=1 です。
4	public IntPtr value	ｽﾃｰﾀｽﾊﾟﾗﾒｰﾀ値を格納する非管理ﾒﾓﾘのﾎﾟｲﾝﾀです。

14. 5. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void free()	value に使用されている非管理ﾒﾓﾘを開放します。
2	public void set_para()	ｽﾃｰﾀｽｽﾃｰﾀｽﾊﾟﾗﾒｰﾀ値を設定します。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 5. 3. 1 free()

インスタンスの value に使用されている非管理メモリを開放します。

【構文】

public void free( )

【引数】

なし

【戻り値】

なし。

【説明】

ステータスパラメータ値 value に使用されている非管理メモリを開放します。

14. 5. 3. 2 set_para()

ステータスパラメータを設定します。

【構文】

public void set_para(int status, int format, int size, IntPtr value)

【引数】

status

状態値です。

format

ステータスパラメータのデータフォーマットです。(HSMS.ICODE_U1 など)

size

ステータスパラメータ値の配列サイズ

value

設定したいステータスパラメータ値が格納されているポインタです。

【戻り値】

なし。

【説明】

各プロパティ値に引数の値を設定します。
value については別の IntPtr メモリを確保します。

14. 5. 3. 3 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshStatusPara dst)

【引数】

dst

コピー先の DshStatusPara インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

14. 6 DshStatusParaList クラス

複数個のステータスパラメータを保存するための DshStatusPara クラスの配列です。

14. 6. 1 コンストラクタ

	名前	説明
1	public DshStatusParaList()	空のｲﾝｽﾀﾝｽを生成します。 配列ｻｲｽﾞは=0 になります。

14. 6. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	設定されているｽﾃｰﾀｽﾊﾟﾗﾒｰﾀの数です。
2	public DshStatusPara[] para_list	ｽﾃｰﾀｽﾊﾟﾗﾒｰﾀ保存ｸﾗｽ DshStatusPara の配列です。

14. 6. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	para_list 配列を空にします。
2	public void add_para()	para_list 配列に 1 個のｽﾃｰﾀｽﾊﾟﾗﾒｰﾀを加えます。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 6. 3. 1 clear()

インスタンスの para_list 配列の中に含まれるステータスパラメータの value に使用されている全非管理メモリを開放し、para_list リストを空にします。

【構文】

public void clear( )

【引数】

なし

【戻り値】

なし。

【説明】

para_list 配列のなかのステータスパラメータ値 value に使用されている非管理メモリを開放し、リストを空にします。そして、count = 0 にします。

14. 6. 3. 2 add_para()

インスタンスの para_list 配列にステータスパラメータ情報を１個追加します。

【構文】

```
public void add_para(DshStatusPara para)
public void add_para(int status, int format, int size, IntPtr value)
```

【引数】

para
　　DshStatusPara クラスのインスタンス
status
　　追加するステータスパラメータ ID です。
format
　　ステータスパラメータ値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　ステータスパラメータ値の配列サイズ
value
　　設定したいステータスパラメータ値が格納されているポインタです。

【戻り値】

なし。

【説明】

para_list 配列に１個のステータスパラメータ情報を追加します。
引数が、DshStatusPara クラスのインスタンスの場合は、それをコピーして設定します。
２番目のオーバーロードメソッドでは、DshStatusPara クラスの set_para()メソッドを使ってインスタンスを生成し追加します。追加後、count + 1 します。

14. 6. 3. 3 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshStatusParaList dst )
```

【引数】

dst
　　コピー先の DshStatusParaList インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

14. 7 DshMaterialOutStatus クラス

MtrlOutByStatus 属性情報を保存するためのクラスです。

14. 7. 1 コンストラクタ

	名前	説明
1	public DshMaterialOutStatus()	空のｲﾝｽﾀﾝｽを生成します。

14. 7. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int status	ｽﾃｰﾀｽです。(material out status)
2	public int slot_count	設定されているｽﾛｯﾄ ID の数です。
3	public DshCjCarSlotList car_slot	ｷｬﾘｱ ID とｽﾛｯﾄ ID を保存する DshCjCarSlotList ｸﾗｽです。(14.9 参照) 情報設定は DshCjCarSLotList ｸﾗｽのﾒｿｯﾄﾞを使用します。

14. 7. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_status()	ｽﾃｰﾀｽを設定します。
2	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
3	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 7. 3. 1 set_status()

ステータス(material out status)を設定します。

【構文】

public void set_status(int mtrl_status)

【引数】

mtrl_status

ステータス値です。

【戻り値】

なし。

【説明】

ステータスを設定します。

14. 7. 3. 2 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshMaterialOutStatus dst)

【引数】

dst

コピー先の DshMaterialOutStatus インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

14. 8 DshMaterialOutSpec クラス

MtrlOutSpec 属性情報を保存するためのクラスです。

14. 8. 1 コンストラクタ

	名前	説明
1	public DshMaterialOutSpec()	空のｲﾝｽﾀﾝｽを生成します。

14. 8. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public DshCjCarSlotList src_car_slot	ｿｰｽのｷｬﾘｱ ID とｽﾛｯﾄ ID を保存する DshCjCarSlotList ｸﾗｽです。(14.9 参照) 情報設定は DshCjCarSLotList ｸﾗｽのﾒｿｯﾄﾞを使用します。
2	public DshCjCarSlotList dst_car_slot	行き先ｷｬﾘｱ ID とｽﾛｯﾄ ID を保存する DshCjCarSlotList ｸﾗｽです。(14.9 参照) 情報設定は DshCjCarSLotList ｸﾗｽのﾒｿｯﾄﾞを使用します。

14. 8. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
2	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 8. 3. 1 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshMaterialOutSpec dst)

【引数】

dst

コピー先の DshMaterialOutSpec インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

14. 9 DshCjCarSlotList クラス

キャリアのスロット ID を配列で保存するためのクラスです。

14. 9. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshCjCarSlotList()	空のｲﾝｽﾀﾝｽを生成します。 配列ｻｲｽﾞは=0 になります。
2	public DshCjCarSlotList(string carid)	ｷｬﾘｱ ID を指定してｲﾝｽﾀﾝｽを生成します。

14. 9. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string carid	ｷｬﾘｱ ID です。
2	public int slot_count	設定されているｽﾛｯﾄ ID の数です。
3	public int[] slot_list	ｽﾛｯﾄ ID を保存する配列です。

14. 9. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	slot_list 配列を空にします。
2	public void set_carid()	ｷｬﾘｱ ID を設定します。 (slot_list を空にします。)
3	public void add_slotid()	slot_list 配列に 1 個のｽﾛｯﾄ ID を加えます。
4	public void copy_slotid()	配列に保存されているｽﾛｯﾄ ID を当該ｲﾝｽﾀﾝｽにｺﾋﾟｰします。
5	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
6	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 9. 3. 1 clear()

インスタンスの slot_list 配列リストを空にします。

【構文】
public void clear( )

【引数】
なし

【戻り値】
なし。

【説明】
slot_list 配列リストを空にします。そして、count = 0 にします。

14. 9. 3. 2 set_carid()

キャリア ID を設定し、slot_list を空にします。

【構文】
public void set_carid(string carid)

【引数】
carid
設定したいキャリア ID です。

【戻り値】
なし。

【説明】
キャリア ID を設定します。そして、スロット ID リストを空にします。

14. 9. 3. 3 add_slotid()

インスタンスの slot_list 配列にスロット ID を 1 個追加します。

【構文】
public void add_slotid(int id)

【引数】
id
追加するスロット ID です。

【戻り値】
なし。

【説明】
slot_list 配列に 1 個のスロット ID を追加します。
追加後、count + 1 します。

14. 9. 3. 4 copy_slotid()

複数個のスロット ID を当該インスタンスの slot_list にコピーします。

【構文】

public void copy_slotid(IntPtr srcptr, int count)

【引数】

srcptr

複数のスロット ID が保存されているコピー元メモリのポインタです。

count

コピーしたいスロット ID 数です。(srcptr に保存されているスロット ID 数)

【戻り値】

なし。

【説明】

srcptr が指すメモリに保存されている count 分のスロット ID を当該インスタンスにコピーします。

14. 9. 3. 5 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshCjCarSlotList dst)

【引数】

dst

コピー先の DshCjCarSlotList インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

14. 10 DshProcControlSpec クラス

ProccessingCtrlSpec 属性情報を保存するためのクラスです。

14. 10. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshProcControlSpec()	空のｲﾝｽﾀﾝｽを生成します。
	public DshProcControlSpec(string prjid)	ﾌﾟﾛｾｽｼﾞｮﾌﾞ ID を指定してｲﾝｽﾀﾝｽを生成します。

14. 10. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string prjobid	ﾌﾟﾛｾｽｼﾞｮﾌﾞ ID です。
2	public DshControlRule control_rule_list	control rule 情報ﾘｽﾄです。 DshControlRule ｸﾗｽは 14.11 で説明します。
3	public DshOutputRule output_rule_list	output rulu 情報ﾘｽﾄです。 DshOutputlRule ｸﾗｽは 14.12 で説明します。

14. 10. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	control_rule_list, output_rule_list を消去します。
2	public void set_prjid()	ﾌﾟﾛｾｽｼﾞｮﾌﾞ ID を設定します。
3	public void add_control_rule()	1 個の control_rule 情報を追加します。
4	public void add_output_rule()	1 個の output rule 情報を追加します。
5	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
6	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 10. 3. 1 clear()

control_rule_list, output_rule_list の内容を消去します。

【構文】

public void clear( )

【引数】

なし。

【戻り値】

なし。

【説明】

control_rule_list, output_rule_list の内容を消去します。
消去には、それぞれ、DshControlRule クラスの clear()メソッド、DshOutputRule クラスの clear()メソッドを使用します。

14. 10. 3. 2 set_prjobid()

プロセスジョブ ID を設定します。

【構文】

public void set_prjid(string prjid)

【引数】

prjid

プロセスジョブ ID です。

【戻り値】

なし。

【説明】

プロセスジョブ ID を設定します。
同時に、control_rule_list, output_rule_list の内容を消去します。

14. 10. 3. 3 add_control_rule()

インスタンスの control_rule_list にコントロールルール情報を 1 個追加します。

【構文】

public void add_control_rule(string name, int format, int size, IntPtr value)

【引数】

name

コントロールルール名です。

format

ルール値のデータフォーマットです。(HSMS.ICODE_U1 など)

size

ルール値の配列サイズです。

value

設定したいルール値が格納されているポインタです。

【戻り値】

なし。

【説明】

control_rule_list に 1 個のコントロールルール情報を追加します。
DshControlRule クラスの add_control_rule メソッドを使って追加します。

14. 10. 3. 4 add_output_rule()

インスタンスの output_rule_list にアウトプットルール情報を１個追加します。

【構文】

public void add_output_rule(int status, int format, int size, IntPtr value)

【引数】

status

材料(material)の状態です。

format

ルール値のデータフォーマットです。(HSMS.ICODE_U1 など)

size

ルール値の配列サイズです。

value

設定したいルール値が格納されているポインタです。

【戻り値】

なし。

【説明】

control_rule_list に１個のアウトプットルール情報を追加します。
DshOutPutRule クラスの add_output_rule メソッドを使って追加します

14. 10. 3. 5 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshProcControlSpec dst)

【引数】

dst

コピー先の DshProcControlSpec インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

14. 11 DshControlRule クラス

ProccessingCtrlSpec 属性のコントロールルール情報を保存するためのクラスです。

14. 11. 1 コンストラクタ

	名前	説明
1	public DshControlRule()	空のｲﾝｽﾀﾝｽを生成します。

14. 11. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	2 の control_list に保存されている control rule 情報の数です。
2	public DshObjPara[] control_list	control rule 情報ﾘｽﾄです。 DshObjPara ｸﾗｽの配列です。 （18.1 DshObjPara ｸﾗｽを参照してください）

14. 11. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	control_list 配列の内容を空にします。
2	public void add_control_rule()	1 個の control_rule 情報を追加します。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 11. 3．1 clear()

control_list 配列の内容を空にします。

【構文】

public void clear( )

【引数】

なし

【戻り値】

なし。

【説明】

control_list 配列の内容を空にします。
DshObjPara クラス内に使用されている非管理メモリを開放します。
count = 0 にします。

14. 11. 3. 2 add_control_rule()

control_list にコントロールルール情報を１個追加します。

【構文】

public void add_control_rule(string name, int format, int size, IntPtr value)

【引数】

name

コントロールルール名です。

format

ルール値のデータフォーマットです。(HSMS.ICODE_U1 など)

size

ルール値の配列サイズです。

value

設定したいルール値が格納されているポインタです。

【戻り値】

なし。

【説明】

control_list 配列に１個のコントロールルール情報を追加します。
追加した後、count + 1 します。

14. 11. 3. 3 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshControlRule dst )
```

【引数】

dst

コピー先の DshControlRule クラスのインスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

14. 12 DshOutputRule クラス

ProccessingCtrlSpec 属性のアウトプットルール情報を保存するためのクラスです。

14. 12. 1 コンストラクタ

	名前	説明
1	public DshOutputRule()	空のｲﾝｽﾀﾝｽを生成します。

14. 12. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	2 の output_list に保存されている output rule 情報の数です。
2	public DshObjPara[] output_list	output rule 情報ﾘｽﾄです。 DshObjPara ｸﾗｽの配列です。 （18.1 DshObjPara ｸﾗｽを参照してください）

14. 12. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	output_list 配列の内容を空にします。
2	public void add_output_rule()	1 個の output_rule 情報を追加します。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 12. 3．1 clear()

output_list 配列の内容を空にします。

【構文】

public void clear( )

【引数】

なし

【戻り値】

なし。

【説明】

output_list 配列の内容を空にします。
DshObjPara クラス内に使用されている非管理メモリを開放します。
count = 0 にします。

14. 12. 3. 2 add_output_rule()

インスタンスの output_list にアウトプットルール情報を 1 個追加します。

【構文】

public void add_output_rule(int status, int format, int size, IntPtr value)

【引数】

status

材料（material）の状態です。

format

ルール値のデータフォーマットです。(HSMS.ICODE_U1 など)

size

ルール値の配列サイズです。

value

設定したいルール値が格納されているポインタです。

【戻り値】

なし。

【説明】

output_list 配列に 1 個のアウトプットルール情報を追加します。
追加した後、count + 1 します。

14. 12. 3. 3 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshOutputRule dst)

【引数】

dst

コピー先の DshOutputRule インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

14. 13 DshTextList クラス

コントロールジョブの属性で、テキスト配列情報を保存するためのクラスです。

14. 13. 1 コンストラクタ

	名前	説明
1	public DshTextList()	空のｲﾝｽﾀﾝｽを生成します。

14. 13. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	2 の text_list に保存されているﾃｷｽﾄ情報の数です。
2	public string[] text_list	ﾃｷｽﾄ情報を保存するための配列です。

14. 13. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	output_list 配列の内容を空にします。
2	public void add_text()	1 個の text 情報を追加します。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 13. 3．1 clear()

text_list 配列の内容を空にします。

【構文】

public void clear( )

【引数】

なし

【戻り値】

なし。

【説明】

text_list 配列の内容を空にします。
count = 0 にします。

14. 13. 3. 2 add_text()

インスタンスの text_list にテキスト情報を１個追加します。

【構文】

public void add_text(string text)

【引数】

text

追加するテキストです。

【戻り値】

なし。

【説明】

text_list 配列に１個のテキスト情報を追加します。
追加した後、count + 1 します。

14. 13. 3. 3 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshTextList dst)

【引数】

dst

コピー先の DshTextList インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

14. 14 DshS14Rsp クラス

プロセスジョブ関連 SECS-Ⅱメッセージの応答情報を保存するためのクラスです。
S14F10, S14F12 メッセージが対象メッセージです。

14. 14. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS14Rsp()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshS14Rsp(DshCj cj_class, int objack)	Cj 情報と objack を指定してｲﾝｽﾀﾝｽを生成します。 DshCj ｸﾗｽについては 14.1 参照
3	public DshS14Rsp(string objspec, int objack)	objspec と objack を指定してｲﾝｽﾀﾝｽを生成します。 S14F10 のためのｺﾝｽﾄﾗｸﾀです。

14. 14. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public objspec	S14F10 に設定する objspec です。
2	public int objack	S14F10,F12 に設定する objack です。
3	public DshCj cj_class	付属するｺﾝﾄﾛｰﾙ情報 DshCj ｸﾗｽのｲﾝｽﾀﾝｽです。
4	public int count	付属するｴﾗｰ情報の数です。 使用 S14 ﾒｯｾｰｼﾞ 共通です。
5	public DshObjError[] list	ｴﾗｰ情報を保存する配列です。 使用 S14 ﾒｯｾｰｼﾞ 共通です。 Vol-1 19.1 参照

14. 14. 3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	cj_class と list の配列に保存されている情報を消去します。
2	public void init_set()	cj_calss, objack, objspec を設定します。
	public void set_count()	ｴﾗｰﾘｽﾄ list 配列のｻｲｽﾞを設定します。
3	public void add_error()	ｴﾗｰ情報(DshObjError)を 1 個 list に追加します。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 14. 3．1　clear()

cj_class に含む属性情報と list に設定されているエラー情報（DshObjError の内容）を全て消去します。

【構文】
public void clear ()

【引数】
なし

【戻り値】
なし。

【説明】
cj_class に含む属性情報と list に設定されているエラー情報（DshObjError の内容）を全て消去します。
cj_class については、使用されている非管理メモリの開放も行います。
そして、count = 0 にします。

14. 14. 3. 2 void init_set()

cj_class に含む属性情報と list に設定されているエラー情報（DshObjError の内容）を全て消去します。

【構文】

```
public void init_set(DshCj cj_class )
public void init_set(DshCj cj_class, int objack)
public void init_set(DshCj cj_class, string objspec, int objack)
public void init_set(string objspec, int objack)
```

【引数】

cj_class

DshCj クラスのインスタンスです。CJ 情報が保存されます。

objack

S14F10, S14F12 に含まれる ojack です。

objspec

S14F10 に設定される objspec です。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティに情報を設定します。
必要に応じて、オーバーロードされたメソッドを使用できます。

14. 14. 3. 3 set_count()

list 配列に、指定されたエラー情報分のサイズの配列を作成します。

【構文】

public void set_count(int count)

【引数】

count

配列サイズです。

【戻り値】

なし。

【説明】

list の配列を引数に与えられたサイズにします。またプロパティの count に値を設定します。
本メソッドは、IntPtr で指された配列リストに格納されている複数個のエラー情報をまとめて list 配列に設定する際に使用します。エラー情報設定に使用する関数は、DshObjError.copy_err_list()です。

list にエラー情報を設定したい場合は、通常、次に説明する add_err()メソッドを使ってください。

14. 14. 3. 4 add_err()

インスタンスの list 配列にエラー情報を 1 個追加します。

【構文】

public void add_error(int code, string text)

【引数】

code

エラーコードです。

text

エラーテキストです。

【戻り値】

なし。

【説明】

list 配列に、1 個のエラー情報を DshObjError のクラスに設定し、追加します。
追加した後、count を +1 します。

14. 15 DshS16F27Rsp クラス

プロセスジョブ関連 SECS-Ⅱメッセージの応答情報を保存するためのクラスです。
S16F27 コントロールジョブコマンドに対する S16F28 応答メッセージが対象メッセージです。

14. 15. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS16F27Rsp()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshS16F27Rsp(int ack)	ack (acka)を指定してｲﾝｽﾀﾝｽを生成します。 err_flag=false にします。
3	public DshS16F27Rsp(int ack, int errcode, string errtext)	ack, errcode, errtext 情報を指定してｲﾝｽﾀﾝｽを生成します。 err_flag=true にします。

14. 15. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int acka	応答ﾒｯｾｰｼﾞの設定する acka です。 使用 S16 ﾒｯｾｰｼﾞ共通です。
2	public bool err_flag	errcode, errtext が有効化どうかを示します。 false=無効(ｴﾗｰ情報なし)、true=有効(ｴﾗｰ情報あり)
3	public int errcode	ｴﾗｰｺｰﾄﾞです。
4	public string errtext	ｴﾗｰﾃｷｽﾄです。

14. 15. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_obj_err()	value に使用されているﾒﾓﾘを開放します。
2	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

14. 15. 3．1 set_obj_err ()

エラーコードとｴﾗｰテキストを設定します。

【構文】

public void set_obj_err(int code, string text)
public void set_obj_err(int code, IntPtr text)

【引数】

code

エラーコードです。

text

エラーテキストです。

【戻り値】

なし。

【説明】

エラーコードとエラーテキストを設定します。
IntPtr で与えられた text は string に変換して設定します。

15．スプール関連クラス

スプール(SPOOL)関連情報を保存するクラスです。

関連クラスの一覧を次表に示します。

	クラス名	用途
1	DshSpool	ｽﾌﾟｰﾙ情報保存ｸﾗｽです。
2	DshSpoolList	ｽﾌﾟｰﾙ情報保存用ｸﾗｽです。 ただし、S16F27 が含む情報を保存します。
3	DshSpoolUnit	１個の stream のｽﾌﾟｰﾙ情報を保存します。
4	DshSpoolRsp	S2F43 に対する S2F44 応答ﾒｯｾｰｼﾞ情報を保存するｸﾗｽです。
5	DshSpoolRspUnit	S2F44 DshSpoolRsp に含まれる stream1 個分に対する応答情報です。

スプールの通信関連クラスとして、以下のものがあります。 これらについては、Vol-2 で説明します。

DshS2F43Send
DshS2F44Response
DshS6F23Send

15. 1 DshSpool クラス

スプール情報を保存するために使用されます。

本クラスはエンジンに登録するスプール情報保存用に使用されます。
スプール対象になる可能性がある全てのストリームに対するスプール情報を保存することができます。

15. 1. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshSpool()	装置 ID=0 のｲﾝｽﾀﾝｽを生成します。
2	public DshSpool(int eqid)	装置 ID を指定してｲﾝｽﾀﾝｽを生成します。

15. 1. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int eqid	装置 ID、ﾃﾞﾌｫﾙﾄ値 = 0 です。
2	public DshSpoolUnit[] list	stream=1 から 127 までの情報を保存するための配列です。 DshSpoolUnit ｸﾗｽは 1 個のｽﾄﾘｰﾑのｽﾌﾟｰﾙ情報を保存するためのｸﾗｽです。

15. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear_all()	ｸﾗｽ内の全ｽﾌﾟｰﾙ情報を消去し、空にします。 ｴﾝｼﾞﾝの管理情報には影響しません。
2	public void clear_stream()	ｸﾗｽ内の指定ｽﾄﾘｰﾑのｽﾌﾟｰﾙ情報を消去します。ｴﾝｼﾞﾝの管理情報には影響しません。
3	public void reset()	ｴﾝｼﾞﾝに管理されているｽﾌﾟｰﾙ情報を消去します。 ｸﾗｽ内の情報は消去しません。 全ｽﾄﾘｰﾑの消去またはｽﾄﾘｰﾑを指定して、その情報だけを消去します。
4	public void add_func()	ｸﾗｽ内のｽﾄﾘｰﾑにﾌｧﾝｸｼｮﾝを１個追加します。
5	public int set()	ｸﾗｽ内に設定された指定ｽﾄﾘｰﾑのｽﾌﾟｰﾙ情報をｴﾝｼﾞﾝに設定します。
6	public int get()	指定されたｽﾄﾘｰﾑにｽﾌﾟｰﾙ対象として設定されているﾌｧﾝｸｼｮﾝを配列に取得します。
7	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

15. 1. 3. 1 clear_all()

クラス内に設定されているスプール情報を全て消去します。

【構文】

public void clear_all()

【引数】

なし

【戻り値】

なし。

【説明】

クラス内の全てのストリームのスプール情報を消去します。
各ストリームに与えられた DshSpoolUnit のファンクションリストを空にし、count = 0 にします。
エンジン管理情報のスプール情報は、消去しません。エンジン管理情報上のスプール情報を消去するためには、reset()メソッドを使用してください。

15. 1. 3. 2 clear ()

指定されたストリームに対して設定されているスプール情報を消去し、空にします。

【構文】

public void clear(int s)

【引数】

s

スプール情報を消去したいストリームコードです。

【戻り値】

なし。

【説明】

クラス内の s で指定されたストリームのスプール情報を全て消去します。
s ストリームに与えられた DshSpoolUnit のファンクションリストを空にし、count = 0 にします。

15. 1. 3. 3 reset()

エンジンの管理情報の中に設定されているスプール情報を全て消去します。

【構文】

```
public void reset()
public void reset( int s )
```

【引数】

s

スプール情報を消去したいストリームコードです。

【戻り値】

なし。

【説明】

引数がない（ストリームの指定なし）の場合、エンジン管理情報の全ストリームのスプール情報を全て消去します。
引数 s が指定されていた場合は、エンジン管理情報の中から s で指定されたスプール情報だけを消去します。
クラス内のスプール情報は、消去しません。クラス内の消去は、clear_all() または clear()メソッドを使用してください。

15. 1. 3. 4 add_func ()

ストリームに対してファンクションを１個追加します。
クラス内のストリームが対象です。

【構文】

```
public void add_func(int s, int f)
```

【引数】

s

ストリームコードです。

f

追加するファンクションコードです。

【戻り値】

なし。

【説明】

クラス内の s で指定されたストリームのスプール情報に f で指定されたファンクションを１個追加します。
list[s]の DshSpoolUnit の func_list に１個追加します。

15. 1. 3. 5 set ()

クラス内に設定されている s に指定されたストリームのスプール情報をエンジンの管理情報に設定します。

【構文】

public int set(int s)

【引数】

s

ストリームコードです。

戻り値】

返却値	意　味
0	正常に設定できた。
(-1)	設定に失敗した。

【説明】

クラス内の s で指定されたストリームのスプール情報に f で指定されたファンクションを 1 個追加します。
list[s]の DshSpoolUnit の func_list に 1 個追加します。

15. 1. 3. 6 get()

エンジン管理情報から、s に指定されたストリームに設定されているスプール情報を取得します。

【構文】

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

【引数】

s

ストリームコードです。

flist

スプール情報に設定されているファンクションを格納するための配列です。

戻り値】

返却値	意　味
>= 0	取得できたﾌｧﾝｸｼｮﾝの数です。 0 はﾌｧﾝｸｼｮﾝがまったく設置されていなかったことを意味します。
(-1)	取得に失敗した。

【説明】

エンジン管理情報から、s で指定されたストリームに設定されているファンクションを flist の配列領域に取得します。

戻り値は、設定されていたファンクションの数です。(-1)の値の場合は、s の値が正しくなかったことを意味します。

15. 2 DshSpoolList クラス

S2F43 メッセージから得られたスプール情報を保存するために使用されます。

１５．１の DshSpool クラスは、エンジンに設定される全てのストリームスプール情報を保存できますが、本クラスは、S2F43 に含まれているストリーム分だけのスプール情報を保存します。

15. 2. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshSpoolList()	空のｲﾝｽﾀﾝｽを生成します。

15. 2. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	ｽﾌﾟｰﾙ情報を保存しているｽﾄﾘｰﾑの数です。
2	public DshSpoolUnit[] list	S2F43 に含まれているｽﾌﾟｰﾙ情報を順次保存するための DshSpoolUnit ｸﾗｽの配列です。 DshSpoolUnit ｸﾗｽは１個のｽﾄﾘｰﾑのｽﾌﾟｰﾙ情報を保存するためのｸﾗｽです。

15. 2. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear ()	ｸﾗｽ内の全ｽﾌﾟｰﾙ情報を消去し、空にします。
2	public void add_stream()	１個のｽﾄﾘｰﾑを list 配列に追加します。
3	public int get_func()	func_list からﾌｧﾝｸｼｮﾝを１個取得します。 配列位置を指定します。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

15. 2. 3. 1 clear()

クラス内に設定されているスプール情報を全て消去します。

【構文】

public void clear ()

【引数】

なし

【戻り値】

なし。

【説明】

クラス内に含む全てのファンクションを消去します。
各ストリームに与えられた DshSpoolListUnit のファンクションリストを空にし、count = 0 にします。

14．2．3．2　decode()

S2F43 メッセージに含まれるスプール情報を DshSpoolList クラス内のプロパティにデコードします。
本メソッドは S2F43 メッセージ受信処理時に使用します。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

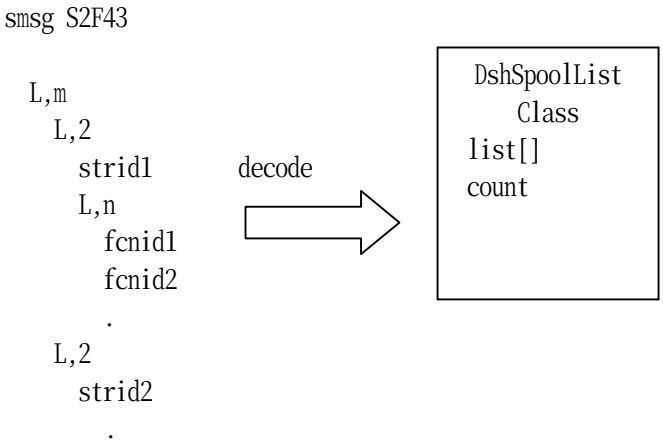
S2F43 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった（S2F43 のﾒｯｾｰｼﾞの形式が正しくなかった）

【説明】

smsg に含まれている情報を DshSpoolList クラス内にデコードします。



15. 3 DshSpoolUnit クラス

１個のストリームのスプール情報を保存するために使用されます。

15. 3. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshSpoolUnit()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshSpoolUnit(int s)	ｽﾄﾘｰﾑを指定してｲﾝｽﾀﾝｽを生成します。

15. 3. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int stream	ｽﾄﾘｰﾑです。
2	public int count	保存しているﾌｧﾝｸｼｮﾝの数です。
3	public int[] func_list	ｽﾄﾘｰﾑに含むﾌｧﾝｸｼｮﾝの配列です。

15. 3. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	ｸﾗｽ内の全ｽﾌﾟｰﾙ情報を消去し、空にします。
2	public void add_func()	１個のﾌｧﾝｸｼｮﾝを func_list 配列に追加します。
3	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

15. 3. 3. 1 clear()

クラス内に設定されているファンクション情報を全て消去します。

【構文】
public void clear ()

【引数】
なし

【戻り値】
なし。

【説明】
クラス内の func_list に含むファンクション情報を消去します。count = 0 にします。

15. 3. 3. 2 add_func ()

ファンクションを１個追加します。

【構文】
public void add_func(int f)

【引数】
f
追加するファンクションコードです。

【戻り値】
なし。

【説明】

クラスの func_list に f で指定されたファンクションを 1 個追加します。count+1 します。

15. 4 DshSpoolSend クラス

S2F43 メッセージを使って送信するスプール情報を保存するために使用されます。

15. 4. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshSpoolSend()	装置 ID=0 のｲﾝｽﾀﾝｽを生成します。

15. 4. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	ｸﾗｽに含まれる DshSpoolUnit ｸﾗｽの数です。
2	public DshSpoolUnit[] list	１個のｽﾄﾘｰﾑの対するｽﾌﾟｰﾙ情報を保存するための配列です。

15．４．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	ｸﾗｽ内の全ｽﾌﾟｰﾙ情報を消去し、空にします。
2	public void add_list()	ｽﾄﾘｰﾑ１個分のｽﾌﾟｰﾙ情報を list に追加します。
3	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

15．４．３．１　clear()

クラス内に設定されているスプール情報を全て消去します。

【構文】
public void clear()

【引数】
なし

【戻り値】
なし。

【説明】
クラス内の全てのストリームのスプール情報を消去します。

15．4．3．2　add_list()

１個のストリームのスプール情報を list に追加します。
ストリームに含まれるファンクションのリストを提供します。

【構文】

public void add_list(int s, int f_count, int[] f_list)

【引数】

s
　　ストリームコードです。
f_count
　　追加するファンクションコードの数です。
f_list
　　ファンクションコードが保存されている配列です。

【戻り値】

なし。

【説明】

クラス内のｓで指定されたストリームの f_list の配列に含まれるファンクションを f_count 分のスプール情報を list に追加します。
追加後、count + 1 します。

15. 5 DshSpoolRsp クラス

S2F43 の応答メッセージ S2F44 に対する応答情報を保存するために使用されます。

15. 5. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshSpoolRsp()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshSpoolRsp(int rsack)	応答 ack を指定してｲﾝｽﾀﾝｽを生成します。

15. 5. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int rsack	S2F44 の ack である rsack の値です。
2	public int count	ｸﾗｽに含まれる DshSpoolRspUnit ｸﾗｽの数です。
3	public DshSpoolRspUnit[] list	S2F43 ﾒｯｾｰｼﾞでうけいれることができなかったｽﾌﾟｰﾙ応答情報が保存される配列です。 DshSpoolRspUnit については 15.6 で説明します。

15. 5. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	ｸﾗｽ内の全ｽﾌﾟｰﾙ応答情報を消去し、空にします。
2	public void add ()	ｽﾄﾘｰﾑ 1 個分のｽﾌﾟｰﾙ応答情報を list に追加します。
3	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

15. 5. 3. 1 clear()

クラス内に設定されているスプール応答情報を全て消去します。

【構文】
public void clear()

【引数】
なし

【戻り値】
なし。

【説明】
クラス内の list に含まれる全てのストリームのスプール応答情報を消去します。
count=0 にします。

15．5．3．2　add()

１個のストリームのスプール応答情報を list に追加します。
ストリームに対する ack である strack、ストリーム、ファンクション情報が含まれています。

【構文】

```
public void add(DshSpoolRspUnit uclass)
```

【引数】

uclass

１個のストリームに対する応答情報です。

【戻り値】

なし。

【説明】

uclass の含まれているスプール応答情報を list に追加します。
追加後、count + 1 します。

15. 6 DshSpoolRspUnit クラス

S2F43 の応答メッセージ S2F44 に対する 1 個のストリームに対する応答情報を保存するために使用されます。
本クラスは、DshSpoolUnit クラスから派生しています。

15. 6. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshSpoolRspUnit()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshSpoolRspUnit(int strack)	ｽﾄﾘｰﾑに対する応答 ack を指定してｲﾝｽﾀﾝｽを生成します。

15. 6. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int strack	S2F44 の 1 個のｽﾄﾘｰﾑに対する ack である strack の値です。
2	public int count	派生元 DshSpoolUnit 内のﾌﾟﾛﾊﾟﾃｨです。 ｸﾗｽに含まれる DshSpoolUnit ｸﾗｽの数です。
3	public DshSpoolUnit[] list	派生元 DshSpoolUnit 内のﾌﾟﾛﾊﾟﾃｨです。 ｸﾗｽに含まれる DshSpoolUnit の配列です。

15. 6. 3 メソッド

	名前	説明
1	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。

16. ホストコマンド、キャリアアクションメッセージ関連クラス

S2F41,S2F49, S3F17, S3F23, S3F25 と S3F27 メッセージで使用するクラスについて説明します。

関連クラスの一覧を次表に示します。

	クラス名	用途
1	DshRCmd	S2F41 ﾎｽﾄｺﾏﾝﾄﾞ ﾒｯｾｰｼﾞ 情報を保存するｸﾗｽです。
2	DshERCmd	S2F49 拡張ﾘﾓｰﾄｺﾏﾝﾄﾞ ﾒｯｾｰｼﾞ 情報を保存するｸﾗｽです。
3	DshCarAction	S3F17 ｷｬﾘｱｱｸｼｮﾝ要求ﾒｯｾｰｼﾞ 情報を保存するｸﾗｽです。
4	DshCarActionPara	S3F17 ｷｬﾘｱｱｸｼｮﾝ要求ﾒｯｾｰｼﾞ に含まれるﾊﾟﾗﾒｰﾀ情報を保存するｸﾗｽです。
5	DshContentMap	S3F17 ｷｬﾘｱｱｸｼｮﾝ要求ﾒｯｾｰｼﾞ に含まれる contentmap 情報を保存するｸﾗｽです。
6	DshSlotMap	S3F17 ｷｬﾘｱｱｸｼｮﾝ要求ﾒｯｾｰｼﾞ に含まれるｽﾛｯﾄﾏｯﾌﾟ情報を保存するｸﾗｽです。
7	DshPortGroupAction	S3F23 ﾎﾟｰﾄｸﾞﾙｰﾌﾟｱｸｼｮﾝ要求ﾒｯｾｰｼﾞ 情報を保存するｸﾗｽです。
8	DshPortAction	S3F25 ﾎﾟｰﾄｱｸｼｮﾝ要求ﾒｯｾｰｼﾞ 情報を保存するｸﾗｽです。
9	DshPortAccess	S3F27 ﾎﾟｰﾄｱｸｾｽ変更ﾒｯｾｰｼﾞ 情報を保存するｸﾗｽです。

通信関連クラスとして、以下のものがあります。 これらについては、Vol-2 で説明します。

DshS2F41Send
DshS2F42Response
DshS2F49Send
DshS2F50Response
DshS3F17Send
DshS3F18Response
DshS3F23Send
DshS3F24Response
DshS3F25Send
DshS3F24Response
DshS3F27Send
DshS3F28Response

16. 1 DshRCmd クラス

S2F41 ホストリモートコマンド送信メッセージに含まれるコマンド情報を保存するためのクラスです。

decode メソッドによって S2F41 メッセージに含まれる情報を本クラス内にデコードすることができます。

16. 1．1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshRCmd()	装置 ID=0 の空のｲﾝｽﾀﾝｽを生成します。
2	public DshRCmd(string cmd)	装置 ID=0 のｲﾝｽﾀﾝｽをｺﾏﾝﾄﾞ名を指定して生成します。
3	public DshRCmd(int eqid)	装置 ID を指定して空のｲﾝｽﾀﾝｽを生成します。
4	public DshRCmd(int eqid, string cmd)	装置 ID、ｺﾏﾝﾄﾞ名を指定してｲﾝｽﾀﾝｽを生成します。

16. 1．2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int eqid	装置 ID です。 ﾃﾞﾌｫﾙﾄ値は 0 です。
2	public string cmd	ﾎｽﾄ(ﾘﾓｰﾄ)ｺﾏﾝﾄﾞです。
3	public int count	含まれるｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀの数です。
4	public DshObjPara[] list	ｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ情報を保存するための DshObjPara ｸﾗｽの配列です。 ﾊﾟﾗﾒｰﾀ値が LIST 構造(ICODE_L)の場合は、その LIST に含まれるｻﾌﾞﾊﾟﾗﾒｰﾀは次の sub_list に保存されます。
5	public DshObjSubPara[] sub_list	上の list のﾊﾟﾗﾒｰﾀ値が LIST 構造の場合、その LIST に含まれるﾊﾟﾗﾒｰﾀ情報を保存します。 LIST 構造ではない位置の配列は null になります。

16. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	list と sub_list 配列内の情報を消去します。
2	public void set_cmd()	ﾎｽﾄｺﾏﾝﾄﾞを設定します。
3	public void add_para()	list 配列に 1 個のｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀを追加します。
4	public int add_sub_para()	sub_list 配列に 1 個のｻﾌﾞﾊﾟﾗﾒｰﾀを追加します。
5	public int decode()	S2F41 ﾒｯｾｰｼﾞを当該ｸﾗｽにﾃﾞｺｰﾄﾞします。
6	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 1. 3. 1 clear()

クラス内に設定されているコマンドパラメータを全て消去します。

【構文】

public void clear ()

【引数】

なし

【戻り値】

なし。

【説明】

クラスの list, sub_list 配列内に含む全てのコマンドパラメータを消去します。
非管理メモリを使用しているパラメータ値についてはメモリを開放します。
そして、count = 0 にします。

16. 1. 3. 2 set_cmd()

ホストコマンドを設定します。

【構文】
public void set_cmd(string cmd)

【引数】
cmd
　　ホストコマンド名です。

【戻り値】
なし。

【説明】
引数に与えられたコマンドを cmd プロパティに設定します。

16. 1. 3. 3 add_para()

インスタンスの list 配列にコマンドパラメータ情報を 1 個追加します。

【構文】
public void add_para(string pname, int format, int size, IntPtr value)
public void add_para(string pname, string value)

【引数】
pname
　　パラメータ名です。
format
　　パラメータ値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　パラメータ値の配列サイズです。
value
　　設定したいパラメータ値が格納されているポインタです。

【戻り値】
なし。

【説明】
list 配列に 1 個のパラメータ情報を DshObjPara のクラスに設定し、追加します。
string で与えられた value は HSMS.ICODE_A のフォーマットとして保存されます。

format が HSMS.ICODE_L、リストの場合は、sub_list に DshObjSubPara クラスのインスタンスを生成します。
(sublist[count]の配列に)
追加した後、count を +1 します。

16．1．3．4　add_sub_para()

インスタンスの sub_list 配列にコマンドパラメータ情報を 1 個追加します。

【構文】

```
public int add_sub_para(int pos, int format, int size, IntPtr value)
public int add_sub_para(int pos, string value)
```

【引数】

pos
　　sub_list の配列位置を指定します。
format
　　パラメータ値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　パラメータ値の配列サイズ
value
　　設定したいパラメータ値が格納されているポインタです。

【戻り値】

なし。

【説明】

sub_list の pos で指定された配列位置に 1 個のパラメータ情報を追加します。
string で与えられた value は HSMS.ICODE_A のフォーマットとして保存されます。

16．1．3．5　copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshRCmd dst )
```

【引数】

dst
　　コピー先の DshRCmd インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

16．1．3．6　decode()

S2F41 に含まれるホストコマンド情報を DshRCmd クラス内のプロパティにデコードします。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

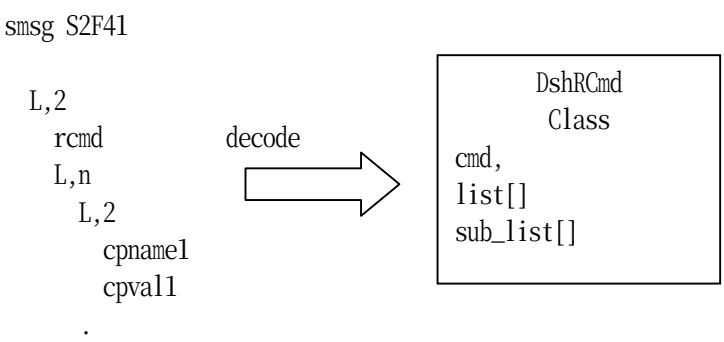
S2F41 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった（S2F41 のﾒｯｾｰｼﾞの形式が正しくなかった。）

【説明】

smsg に含まれている情報を DshRCmd クラス内にデコードします。



メッセージに含まれるコマンドとパラメータがクラスのプロパティに保存されます。
正常にデコードできた場合、0 を返却します。
もし、S2F41 のメッセージフォーマットが正しくないなどの理由でデコードできなかった場合、 (-1)を返却します。

16. 2 DshERCmd クラス

S2F49 拡張リモートコマンド送信メッセージに含まれるコマンド情報を保存するためのクラスです。

decode メソッドによって S2F49 メッセージに含まれる情報を本クラス内にデコードすることができます。

16. 2. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshERCmd()	装置 ID=0 の空のｲﾝｽﾀﾝｽを生成します。
2	public DshERCmd(string objspec, string cmd)	装置 ID=0 のｲﾝｽﾀﾝｽを objspec とｺﾏﾝﾄﾞ名を指定して生成します。
3	public DshERCmd(int eqid)	装置 ID を指定して空のｲﾝｽﾀﾝｽを生成します。
4	public DshERCmd(int eqid, string objspec, string cmd)	装置 ID、objspec, ｺﾏﾝﾄﾞ名を指定してｲﾝｽﾀﾝｽを生成します。

16. 2. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int eqid	装置 ID です。 ﾃﾞﾌｫﾙﾄ値は 0 です。
2	public string cmd	ﾎｽﾄ(ﾘﾓｰﾄ)ｺﾏﾝﾄﾞです。 (下の表参照)
3	public int count	含まれるｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀの数です。
4	public DshObjParaList[] list	ｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ情報を保存するための DshObjParaList ｸﾗｽの配列です。 ﾊﾟﾗﾒｰﾀのﾈｽﾃｨﾝｸﾞ情報も保存されます。
5	public DshDataItemList[] sub_list	上の list のﾊﾟﾗﾒｰﾀ値が LIST 構造の場合で、複数のﾊﾟﾗﾒｰﾀを有するものについてﾃﾞｰﾀ情報を保存します。 使用されていない場合は、null がｾｯﾄされます。

拡張リモートコマンド名

コマンド名
“abort”
“cancel”
“infoupdate”
“pause”
“resume”
“stage”
“stagedelete”
“transfer”
“pp_select”
“start”
“stop”

16. 2. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	list と sub_list 配列内の情報を消去します。
2	public void set_cmd()	objspec と拡張ﾘﾓｰﾄｺﾏﾝﾄﾞを設定します。
3	public void add_para()	list 配列に 1 個のｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀを追加します。
4	public int add_sub_para()	sub_list 配列に 1 個のｻﾌﾞﾊﾟﾗﾒｰﾀを追加します。
5	public int set_multi()	ﾊﾟﾗﾒｰﾀが、cpname, cpval のﾘｽﾄ構造で多重にﾈｽﾃｨﾝｸﾞした場合に、指定された list 位置に対して multi ﾌﾗｸﾞをｾｯﾄします。
6	public void add_multi_para()	cpname, cpval のﾘｽﾄ構造が多重にﾈｽﾃｨﾝｸﾞしたﾊﾟﾗﾒｰﾀに対して 1 個のｻﾌﾞﾊﾟﾗﾒｰﾀを追加します。
7	public int decode()	S2F49 ﾒｯｾｰｼﾞを当該ｸﾗｽにﾃﾞｺｰﾄﾞします。
8	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 2. 3. 1 clear()

クラス内に設定されているコマンドパラメータを全て消去します。

【構文】

public void clear ()

【引数】

なし

【戻り値】

なし。

【説明】

クラスの list, sub_list 配列内に含む全てのコマンドパラメータを消去します。
非管理メモリを使用しているパラメータ値についてはメモリを開放します。
そして、count = 0 にします。

16. 2. 3. 2 set_cmd()

objspec と拡張リモートコマンドを設定します。

【構文】

public void set_cmd(string objspec, string cmd)

【引数】

objsepc

objspec(オブジェクト仕様)です。

cmd

拡張リモートコマンド名です。

【戻り値】

なし。

【説明】

引数に与えられた objspec とコマンドを objspec, cmd プロパティに設定します。

16. 2. 3. 3 add_para()

インスタンスの list 配列にコマンドパラメータ情報を 1 個追加します。

【構文】

public void add_para(string pname, int format, int size, IntPtr value)
public void add_para(string pname, string value)

【引数】

pname
　　パラメータ名です。
format
　　パラメータ値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　パラメータ値の配列サイズです。
value
　　設定したいパラメータ値が格納されているポインタです。

【戻り値】

なし。

【説明】

list 配列に、1 個のパラメータ情報を DshObjParaList のクラスに設定し、追加します。
string で与えられた value は HSMS.ICODE_A のフォーマットとして保存されます。

format が HSMS.ICODE_L、リストの場合は、sub_list に DshDataItemList クラスのインスタンスを生成します。
(sublist[count]の配列に)

追加した後、count を +1 します。

16．2．3．4　add_sub_para()

インスタンスの sub_list 配列にコマンドパラメータ情報を１個追加します。

【構文】

public int add_sub_para(int pos, int format, int size, IntPtr value)
public int add_sub_para(int pos, string value)

【引数】

pos
　　sub_list の配列位置を指定します。
format
　　パラメータ値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　パラメータ値の配列サイズです。
value
　　設定したいパラメータ値が格納されているポインタです。

【戻り値】

なし。

【説明】

sub_list の pos で指定された配列位置に１個のパラメータ情報を追加します。
string で与えられた value は HSMS.ICODE_A のフォーマットとして保存されます。

16. 2. 3. 5 set_multi()

list 配列の指定された配列位置のﾊﾟﾗﾒｰﾀが cpname, cpval の構造でネスティングする旨のフラグを設定します。

【構文】

public int set_multi(int index)

【引数】

index

list の配列位置を指定します。

戻り値】

返却値	意　味
0	正常に設定できた。
(-1)	設定に失敗した。index で指定された配列位置は未設定であった。

【説明】

index で指定された list の配列位置のパラメータ情報に対し、同じ構造の情報がネスティングすることを示すフラグ multi_flag に１を設定します。(multi_flag については DshERCParaList クラスの説明を参照下さい。

もし、list[index]のパラメータが未設定の場合は (-1) を返却します。

16．2．3．6　add_multi_para()

インスタンスの list 配列にコマンドパラメータ情報を 1 個追加します。

【構文】

```
public int add_multi_para(int index, string pname, int format, int size, IntPtr value)
public int add_multi_para(int index, string pname, string value)
```

【引数】

index
　　ネスティングしたパラメータを加えたい list 配列の位置を指定します。
pname
　　パラメータ名です。
format
　　パラメータ値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　パラメータ値の配列サイズです。
value
　　設定したいパラメータ値が格納されているポインタです。

【戻り値】

返却値	意　味
0	正常に追加できた。
(-1)	追加に失敗した。index で指定された配列位置は未設定であった。

【説明】

list の index で指定された位置の配列に多重パラメータとして 1 個のサブパラメータ情報を追加します。
string で与えられた value は HSMS.ICODE_A のフォーマットとして保存されます。

追加した後、count を +1 します。

もし、list の index で指定された配列位置が未設定であったり、multi_flag が =1 でなかった場合には、(-1) を返却します。

16. 2. 3. 7 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshERCmd dst )
```

【引数】

dst

コピー先の DshERCmd インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

16．2．3．8　decode()

S2F49 に含まれる拡張リモートコマンド情報を DshERCmd クラス内のプロパティにデコードします。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

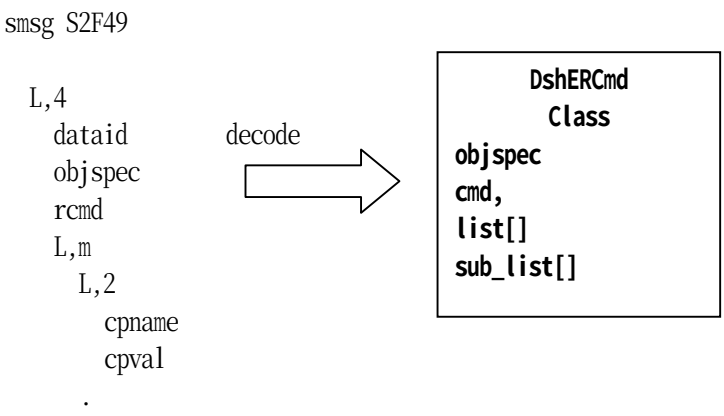
S2F49 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった（S2F49 のﾒｯｾｰｼﾞの形式が正しくなかった。）

【説明】

smsg に含まれている情報を DshERCmd クラス内にデコードします。



メッセージに含まれるコマンドとパラメータがクラスのプロパティに保存されます。
正常にデコードできた場合、0 を返却します。
もし、S2F49 のメッセージフォーマットが正しくないなどの理由でデコードできなかった場合、 (-1)を返却します。

16. 3 DshCarAction クラス

S3F17 キャリアアクション送信メッセージに含まれるキャリアアクション情報を保存するためのクラスです。

decode メソッドによって S3F17 メッセージに含まれる情報を本クラス内にデコードすることができます。

16. 3. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshCarAction()	装置 ID=0 の空のｲﾝｽﾀﾝｽを生成します。
2	public DshCarAction(string caction, string carid, int port)	装置 ID=0 のｲﾝｽﾀﾝｽを caction, ｷｬﾘｱ名, ﾎﾟｰﾄを指定して生成します。
3	public DshCarAction(int caction_index, string carid, int port)	装置 ID=0 のｲﾝｽﾀﾝｽを caction のｲﾝﾃﾞｸｽ, ｷｬﾘｱ名, ﾎﾟｰﾄを指定して生成します。
4	public DshCarAction(int eqid)	装置 ID を指定して空のｲﾝｽﾀﾝｽを生成します。
5	public DshCarAction(int eqid, string caction, string carid. int port)	装置 ID、caction, ｷｬﾘｱ名、ﾎﾟｰﾄを指定してｲﾝｽﾀﾝｽを生成します。
6	public DshCarAction(int eqid, int caction_index, string carid, int port)	装置 ID、caction ｲﾝﾃﾞｸｽ,ｷｬﾘｱ名,ﾎﾟｰﾄを指定してｲﾝｽﾀﾝｽを生成します。

16. 3. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int eqid	装置 ID です。 ﾃﾞﾌｫﾙﾄ値は 0 です。
2	public string carid	ｷｬﾘｱ ID です。
3	public int default_capacity	容量のﾃﾞﾌｫﾙﾄ値です。 装置起動ﾌｧｲﾙの capacity で設定される値です。
4	public string action	ｱｸｼｮﾝ名です。(下の表参照)
5	public caction_index	ｱｸｼｮﾝ名に対するｲﾝﾃﾞｸｽ値です。 (action から変換します)
6	public int port	ﾛｰﾄﾞﾎﾟｰﾄです。
7	public int count	ｱｸｼｮﾝ属性情報の数です。
8	public DshCarActionPara[] list	ｱｸｼｮﾝ属性(ﾊﾟﾗﾒｰﾀ)情報を保存する DshCarActionPara の配列です。

caction_index - キャリアアクションインデクス値 / アクション名

index 値	定数名 - index	アクション名 - action
0	CA_Bind	**"Bind"**
1	CA_CancelBind	**"CancelBind"**
2	CA_CancelCarrier	**"CancelCarrier"**
3	CA_CancelCarrierAtPort	**"CancelCarrierAtPort"**
4	CA_CancelCarrierNotification	**"CancelCarrierNotification"**
5	CA_CancelCarrierOut	**"CancelCarrierOut"**
6	CA_CarrierIn	**"CarrierIn"**
7	CA_CarrierNotification	**"CarrierNotification"**
8	CA_CarrierOut	**"CarrierOut"**
9	CA_CarrierReCreate	**"CarrierReCreate"**
10	CA_CarrierRelease	**"CarrierRelease"**
11	CA_ProceedWithCarrier	**"ProceedWithCarrier"**

16. 3. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	list 配列内の属性情報を消去します。
2	public void init_set()	ｱｸｼｮﾝ、ｷｬﾘｱ ID, ﾎﾟｰﾄを設定します。
3	public void add_para()	list 配列に 1 個の属性を追加します。
4	public int get_attr_pos()	指定属性(ﾊﾟﾗﾒｰﾀ)が保存されている配列位置を取得する。
5	public int decode()	S3F17 ﾒｯｾｰｼﾞを当該ｸﾗｽにﾃﾞｺｰﾄﾞします。
6	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 3. 3. 1 clear()

クラス内に設定されている属性情報を全て消去します。

【構文】

public void clear()

【引数】

なし

【戻り値】

なし。

【説明】

クラスの list, sub_list 配列内に含む全てのキャリア属性を消去します。
非管理メモリを使用している属性値についてはメモリを開放します。
そして、count = 0 にします。

16．3．3．2　init_set()

アクション名、キャリア ID とポートを設定します。

【構文】

public void init_set(string caction, string carid, int port)
public void init_set(int caction_index, string carid, int port)

【引数】

caction
　　キャリアアクション名です。(文字列)
caction_index
　　キャリアアクションのインデクス値です。
　　（アクションインデクスは dsh_const クラス内に定義されています。）
carid
　　キャリア ID です。
port
　　ポート番号です。

【戻り値】

なし。

【説明】

引数に与えられた caction 名、carid, port をそれぞれプロパティに設定します。
設定時に 属性リスト list 配列を空にします。

16. 3. 3. 3 add_para()

インスタンスの list 配列にキャリア属性情報を 1 個追加します。

【構文】

public void add_para(string cattrid, int attrdata)
public void add_para(string cattrid, string attrdata)
public void add_para(int cattr_index, string attrdata)
public void add_para(int cattr_index, int attrdata)

【引数】

cattrid
　　属性 ID です。
cattrid_index
　　属性 ID インデクスです。　(dsh_const クラスに定義されているインデクス値です。)
attrdata
　　属性値です。文字列型(string)と整数型(int)のものがあります。

【戻り値】

なし。

【説明】

list 配列に、1 個の属性情報を DshObjParaList のクラスに設定し、追加します。
cattrid attr_index によって属性値の型が決まっていますので、正しく設定してください。
追加した後、count を +1 します。

16. 3. 3. 4 get_attr_pos ()

特定の属性が保存されている配列位置を検索します。

【構文】

public int get_attr_pos(int attr_index)

【引数】

index

属性のインデクス値です。(dsh_const クラスで定義されています。)

戻り値】

返却値	意　味
>= 0	保存されている list の配列位置です。
(-1)	見つけることができなかった。

【説明】

list の属性情報配列リスト上に attr_index で指定された属性が存在するかどうかを検索します。
見つかったら、list の配列位置を返却します。
見つからなかったら、(-1)を返却します。

16. 3. 3. 5 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshCarAction dst)

【引数】

dst

コピー先の DshCarAction インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

16．3．3．6　decode()

S3F17 に含まれるキャリアアクション情報を DshCarAction クラス内のプロパティにデコードします。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

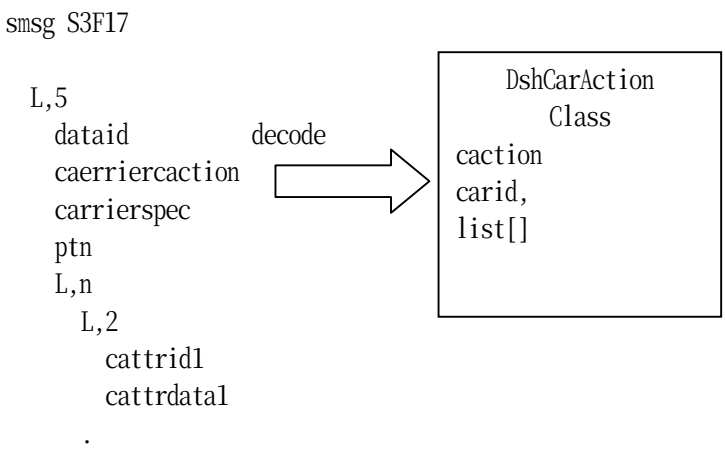
S3F17 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった（S3F17 のﾒｯｾｰｼﾞの形式が正しくなかった。）

【説明】

smsg に含まれている情報を DshCarAction クラス内にデコードします。



メッセージに含まれるアクション名、キャリア ID と属性がクラスのプロパティに保存されます。
正常にデコードできた場合、0 を返却します。
もし、S3F17 のメッセージフォーマットが正しくないなどの理由でデコードできなかった場合、 (-1)を返却します。

16. 4 DshCarActionPara クラス

S3F17 キャリアアクション送信メッセージに含まれる 1 個のアクション属性情報を保存するためのクラスです。

16. 4. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshCarActionPara()	空のｲﾝｽﾀﾝｽを生成します。

16. 4. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string attrid	ｱｸｼｮﾝ属性 ID です。(下の表参照)
2	public int attr_index	属性 ID をｲﾝﾃﾞｸｽに変換したものです。 dsh_const ｸﾗｽに定義されています。
3	public DshContentMap content_map	ContentMap 情報を保存します。 DshContentMap ｸﾗｽに保存します。 16.5 参照
4	public DshSlotMap slot_map	SlotMap 情報を保存します。 DshSlotMap ｸﾗｽに保存します。 16.6 参照
5	public string objtype	ｵﾌﾞｼﾞｪｸﾄﾀｲﾌﾟ です。
6	public string objid	ｵﾌﾞｼﾞｪｸﾄ ID をです。
7	public int capacity	ｷｬﾘｱ容量(slot 数)です。
8	public int access_status	ｱｸｾｽ状態です。
9	public int id_status	ｷｬﾘｱ ID 読取り状態です。
10	public string location	ｷｬﾘｱの所在位置です。
11	public int map_status	ｽﾛｯﾄﾏｯﾌﾟ の読取り状態です。
12	public int subst_count	基板数です。
13	public string usage	usage(用途)です。

キャリアアクション属性インデクス値 / 属性 ID

index 値	定数名 – attr_index	属性 ID - attrid
0	CA_ObjType	**"ObjType"**
1	CA_ObjId	**"ObjId"**
2	CA_Capacity	**"Capacity"**
3	CA_CarrierAccessingStatus	**"CarrierAccessingStatus"**
4	CA_CarrierIDStatus	**"CarrierIDStatus"**
5	CA_ContentMap	**"ContentMap"**
6	CA_LocationID	**"LocationID"**
7	CA_SlotMap	**"SlotMap"**
8	CA_SlotMapStatus	**"SlotMapStatus"**
9	CA_SubStrateCount	**"SlotMapStatus"**
10	CA_Usage	**"Usage"**

16. 4. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	content_map, slot_map の内容を消去します。
2	public void set_info()	1 個の属性情報を設定します。
3	public void add_content_map()	content_map の要素情報を追加します。 lotid, substid
4	public void add_slot_map()	slot_map に slotid を追加します。
5	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 4. 3. 1 clear()

クラス内の content_map, slot_map プロパティの内容を消去します。

【構文】
public void clear()

【引数】
なし

【戻り値】
なし。

【説明】
content_map, slot_map プロパティの内容を消去し、空にします。

16．4．3．2　set_info()

属性 ID、属性インデクスと属性値を設定します。

【構文】

public void set_info(string attrid, int attr_index, string attrdata)
public void set_info(string attrid, int attr_index, int attrdata)
public void set_info(int attr_index, int int attr_data)
public void set_info(int attr_index, string attr_data)

【引数】

attrid
　　属性 ID です。
attr_index
　　属性 ID のインデクス値です。
　　（属性インデクスは dsh_const クラス内に定義されています。）
attr_data
　　属性値です。文字列型と整数型の１つの種類があります。

【戻り値】

なし。

【説明】

引数に与えられた属性 ID または属性インデクスに従って、それに対応したプロパティに値を設定します。
attr_index が指定された場合、インデクスから属性 ID を求めてそれを attrid に設定します。
また、attrid が指定された場合、属性 ID からインデクスを求めて attr_index に設定します。

属性(ID or Index)が決まると値のデータ型がきまりますので、メソッドを使い分けして下さい。

属性が CONTENTMAP, SLOTMAP につきましては、本メソッドで属性を設定した後、それぞれ、
add_content_map()、add_slot_map()メソッドを使ってそれぞれの詳細情報を設定してください。

16. 4. 3. 3 add_content_map()

インスタンスの content_map 属性情報にロット ID と基板 ID を追加します。

【構文】

public void set_content_map(string lotid, string substid)

【引数】

lotid

ロット ID です。

substid

基板 ID です。

【戻り値】

なし。

【説明】

content_map プロパティに 1 個の情報 lotid, substid を追加します。
DshContentMap クラスの lotid_list、subst_list の配列に追加されます。

16. 4. 3. 4 add_slot_map()

インスタンスの slot_map 属性情報をスロット ID を 1 個追加します。

【構文】

public void add_slot_map(int slotid)

【引数】

slotid

スロット ID です。

【戻り値】

なし。

【説明】

slot_map プロパティに slotid を 1 個だけ追加します。
DshSlotMap クラスの slotid_list 配列に追加されます。

16．４．３．５　copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshCarActionPara dst )
```

【引数】

dst

コピー先の DshCarActionPara インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

16. 5 DshContentMap クラス

S3F17 キャリアアクション送信メッセージに含まれる CONTENTMAP 属性情報を保存するためのクラスです。

16. 5. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshContentMap()	空のｲﾝｽﾀﾝｽを生成します。

16. 5. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	lotid_list, subst_list の配列に保存されている情報の数です。
2	public string[] lotid_list	ﾛｯﾄ ID を保存する配列です。
3	public string[] subst_list	基板 ID を保存する配列です。

16. 5. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	lotid_list, subst_list の内容を消去します。
2	public void add_info()	1 個のｺﾝﾃﾝﾄﾏｯﾌﾟ情報を配列に追加します。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を他の DshContentMap ｸﾗｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 5. 3. 1 clear()

クラス内の lotid_list, subst_list 配列の内容を消去します。

【構文】

public void clear()

【引数】

なし

【戻り値】

なし。

【説明】

lotid_list, subst_list 配列の内容を消去し、空にします。
count は =0 にします。

16. 5. 3. 2 add_info()

ロット ID、基板 ID を lotid_list, subst_list 配列に追加します。

【構文】

public void add_info(string lotid, string substid)

【引数】

lotid

追加するロット ID です。

substid

追加する基板 ID です。

【戻り値】

なし。

【説明】

lotid を lotid_list 配列に、substid を subst_list の配列にそれぞれ追加します。
追加した後、count + 1 します。

16. 5. 3. 3 copy()

当該インスタンスの内容を別の DshContentMap クラスのインスタンスにコピーします。

【構文】

public void copy(ref DshContentMap dst)

【引数】

dst

コピー先インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンス内に保存されている count 分の lotid_list, subst_list 配列の情報を dst で指定されたインスタンスにコピーします。

16. 6 DshSlotMap クラス

S3F17 キャリアアクション送信メッセージに含まれる SLOTMAP 属性情報を保存するためのクラスです。

16. 6. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshSlotMap()	空のｲﾝｽﾀﾝｽを生成します。

16. 6. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	slotid_list の配列に保存されている情報の数です。
2	public int[] slotid_list	ｽﾛｯﾄ ID を保存する配列です。

16. 6. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	slotid_list の内容を消去します。
2	public void add_info()	1 個のｽﾛｯﾄ ID を配列に追加します。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を他の DshSlotMap ｸﾗｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 6. 3. 1 clear()

クラス内の slotid_list 配列の内容を消去します。

【構文】

public void clear()

【引数】

なし

【戻り値】

なし。

【説明】

slotid_list 配列の内容を消去し、空にします。
count は =0 にします。

16. 6. 3. 2 add_info()

スロット ID を slotid_list 配列に追加します。

【構文】

public void add_info(int slotid)

【引数】

slotid

追加するスロット ID です。

【戻り値】

なし。

【説明】

slotid を slotid_list 配列に追加します。
追加した後、count + 1 します。

16. 6. 3. 3 copy()

当該インスタンスの内容を別の DshSlotMap クラスのインスタンスにコピーします。

【構文】

public void copy(ref DshSlotMap dst)

【引数】

dst

コピー先インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンス内に保存されている count 分の slotid_list 配列の情報を dst で指定されたインスタンスにコピーします。

16. 7 DshPortGroupAction クラス

S3F23 ポートグループアクション要求メッセージに含まれるアクション情報を保存するためのクラスです。

decode メソッドによって S3F23 メッセージに含まれる情報を本クラス内にデコードすることができます。

16. 7. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshPortGroupAction()	装置 ID=0 の空のｲﾝｽﾀﾝｽを生成します。
2	public DshPortGroupAction(string action, string groupname)	装置 ID=0 のｲﾝｽﾀﾝｽをｱｸｼｮﾝ名、ｸﾞﾙｰﾌﾟ名を指定して生成します。
3	public DshPortGroupAction(int eqid)	装置 ID を指定して空のｲﾝｽﾀﾝｽを生成します。
4	public DshPortGroupAction(int eqid, string action, string groupname)	装置 ID、ｱｸｼｮﾝ名、ｸﾞﾙｰﾌﾟ名を指定してｲﾝｽﾀﾝｽを生成します。

16. 7. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int eqid	装置 ID です。 ﾃﾞﾌｫﾙﾄ値は 0 です。
2	public string action	ｱｸｼｮﾝ名です。
3	public int action_index	ｱｸｼｮﾝ名に対するｲﾝﾃﾞｸｽです。
4	public string group_name	ﾎﾟｰﾄｸﾞﾙｰﾌﾟ名です。
5	public int count	含まれるｱｸｼｮﾝﾊﾟﾗﾒｰﾀの数です。
6	public DshObjPara[] list	ｱｸｼｮﾝﾊﾟﾗﾒｰﾀ情報を保存するための DshObjPara ｸﾗｽの配列です。

16. 7. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	list と sub_list 配列内の情報を消去します。
2	public void set_info()	ｱｸｼｮﾝ名とｸﾞﾙｰﾌﾟ名を設定します。
3	public void add_para()	list 配列に 1 個のｱｸｼｮﾝﾊﾟﾗﾒｰﾀを追加します。
4	public int decode()	S3F23 ﾒｯｾｰｼﾞを当該ｸﾗｽにﾃﾞｺｰﾄﾞします。
5	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 7. 3. 1 clear()

クラス内に設定されているアクションパラメータを全て消去します。

【構文】

public void clear ()

【引数】

なし

【戻り値】

なし。

【説明】

クラスの list 配列内に含む全てのアクションパラメータを消去します。
非管理メモリを使用しているパラメータ値についてはメモリを開放します。
そして、count = 0 にします。

16．7．3．2　set_info()

グループ名とアクション名を設定します。

【構文】

public void set_info(string action , string group_name)

【引数】

action

　　アクション名です。

group_name

　　ポートグループ名です。

【戻り値】

なし。

【説明】

引数に与えられたアクション名とグループ名をそれぞれ action と group_name プロパティに設定します。

16．7．3．3　add_para()

インスタンスの list 配列にアクションパラメータ情報を１個追加します。

【構文】

```
public void add_para(string pname, int format, int size, IntPtr value)
public void add_para(string pname, string value)
```

【引数】

pname
　　パラメータ名です。
format
　　パラメータ値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　パラメータ値の配列サイズです。
value
　　設定したいパラメータ値が格納されているポインタです。

【戻り値】

なし。

【説明】

list 配列に１個のパラメータ情報を DshObjPara のクラスに設定し、追加します。
string で与えられた value は HSMS.ICODE_A のフォーマットとして保存されます。
追加した後、count を +1 します。

16．7．3．4　copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshCarPortGroupAction dst )
```

【引数】

dst
　　コピー先の DshCarPortGroupAction インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

16．7．3．5　decode()

S3F23 に含まれるポートグループアクション情報を DshPortGroupAction クラス内のプロパティにデコードします。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

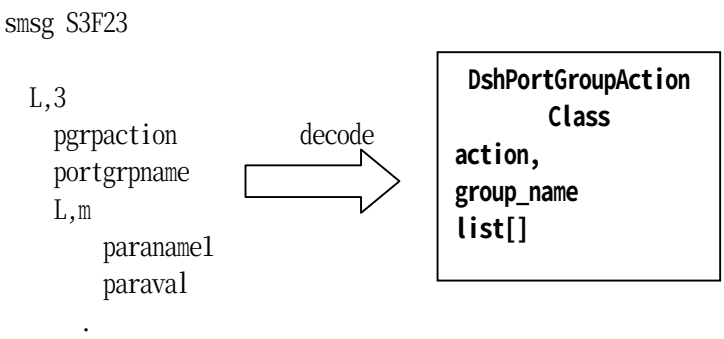
S3F23 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった（S3F23 のﾒｯｾｰｼﾞの形式が正しくなかった。）

【説明】

smsg に含まれている情報を DshPortGroupAction クラス内にデコードします。



メッセージに含まれるアクション名、グループ名とパラメータ情報がクラスのプロパティに保存されます。
正常にデコードできた場合、0 を返却します。
もし、S3F23 のメッセージフォーマットが正しくないなどの理由でデコードできなかった場合、 (-1)を返却します。

16. 8 DshPortAction クラス

S3F25 ポートアクション要求メッセージに含まれるアクション情報を保存するためのクラスです。

decode メソッドによって S3F25 メッセージに含まれる情報を本クラス内にデコードすることができます。

16. 8. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshPortAction()	装置 ID=0 の空のｲﾝｽﾀﾝｽを生成します。
2	public DshPortAction(string action, int port)	装置 ID=0 のｲﾝｽﾀﾝｽをｱｸｼｮﾝ名､ﾎﾟｰﾄを指定して生成します。
3	public DshPortAction(int eqid)	装置 ID を指定して空のｲﾝｽﾀﾝｽを生成します。
4	public DshPortAction(int eqid, string action, int port)	装置 ID、ｱｸｼｮﾝ名、ﾎﾟｰﾄを指定してｲﾝｽﾀﾝｽを生成します。

16. 8. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int eqid	装置 ID です。 ﾃﾞﾌｫﾙﾄ値は 0 です。
2	public string action	ｱｸｼｮﾝ名です。
3	public int action_index	ｱｸｼｮﾝ名に対するｲﾝﾃﾞｸｽです。
4	public int port	ﾎﾟｰﾄです。
5	public int count	含まれるｱｸｼｮﾝﾊﾟﾗﾒｰﾀの数です。
6	public DshObjPara[] list	ｱｸｼｮﾝﾊﾟﾗﾒｰﾀ情報を保存するための DshObjPara ｸﾗｽの配列です。

16. 8. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	list 配列内の情報を消去します。
2	public void set_info()	ｱｸｼｮﾝ名とﾎﾟｰﾄを設定します。
3	public void add_para()	list 配列に 1 個のｱｸｼｮﾝﾊﾟﾗﾒｰﾀを追加します。
4	public int decode()	S3F25 ﾒｯｾｰｼﾞを当該ｸﾗｽにﾃﾞｺｰﾄﾞします。
5	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 8. 3. 1 clear()

クラス内に設定されているアクションパラメータを全て消去します。

【構文】

public void clear ()

【引数】

なし

【戻り値】

なし。

【説明】

クラスの list 配列内に含む全てのアクションパラメータを消去します。
非管理メモリを使用しているパラメータ値についてはメモリを開放します。
そして、count = 0 にします。

16. 8. 3. 2 set_info()

アクション名とポートを設定します。

【構文】

public void set_info(string action, int port)

【引数】

action

アクション名です。

port

アクション対象ポート番号です。

【戻り値】

なし。

【説明】

引数に与えられたアクション名、port をそれぞれ action と port プロパティに設定します。

16．8．3．3　add_para()

インスタンスの list 配列にアクションパラメータ情報を１個追加します。

【構文】

```
public void add_para(string pname, int format, int size, IntPtr value)
public void add_para(string pname, string value)
```

【引数】

pname
　　パラメータ名です。
format
　　パラメータ値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　パラメータ値の配列サイズ
value
　　設定したいパラメータ値が格納されているポインタです。

【戻り値】

なし。

【説明】

list 配列に１個のパラメータ情報を DshObjPara のクラスに設定し、追加します。
string で与えられた value は HSMS.ICODE_A のフォーマットとして保存されます。
追加した後、count を +1 します。

16．8．3．4　copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshCarPortAction dst )
```

【引数】

dst
　　コピー先の DshCarPortAction インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

16．8．3．5　decode()

S3F25 に含まれるポートアクション情報を DshPortAction クラス内のプロパティにデコードします。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

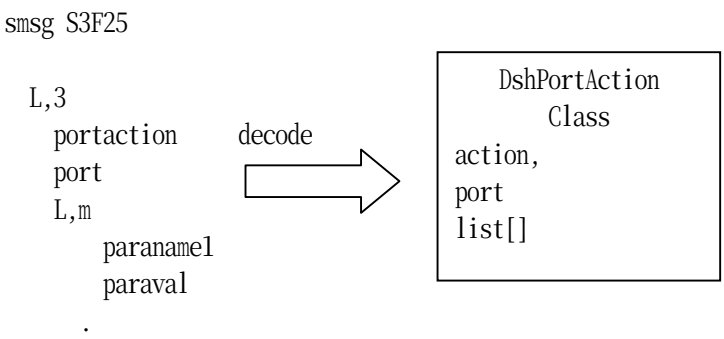
S3F25 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった（S3F25 のﾒｯｾｰｼﾞの形式が正しくなかった。）

【説明】

smsg に含まれている情報を DshPortAction クラス内にデコードします。



メッセージに含まれるアクション名、ポート番号とパラメータ情報がクラスのプロパティに保存されます。
正常にデコードできた場合、0 を返却します。
もし、S3F25 のメッセージフォーマットが正しくないなどの理由でデコードできなかった場合、 (-1)を返却します。

16. 9 DshPortAccess クラス

S3F27 ポートアクセス変更要求メッセージに含まれるアクセスモード変更情報を保存するためのクラスです。

decode メソッドによって S3F27 メッセージに含まれる情報を本クラス内にデコードすることができます。

16. 9. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshPortAccess()	装置 ID=0 の空のｲﾝｽﾀﾝｽを生成します。
2	public DshPortAccess(int mode)	装置 ID=0 のｲﾝｽﾀﾝｽをﾓｰﾄﾞ指定して生成します。
3	public DshPortAccess(int eqid, int mode)	

16. 9. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int eqid	装置 ID です。 ﾃﾞﾌｫﾙﾄ値は 0 です。
2	public int mode	ｱｸｾｽﾓｰﾄﾞです。
3	public int count	対象となるﾎﾟｰﾄの数です。
4	public int[] list	ﾎﾟｰﾄ番号を保存するための配列です。

16. 9. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	port_list 配列内の情報を消去します。
2	public void set_info()	ﾓｰﾄﾞを設定します。
3	public void add_port()	list 配列に 1 個のﾎﾟｰﾄ番号を追加します。
4	public int decode()	S3F27 ﾒｯｾｰｼﾞを当該ｸﾗｽにﾃﾞｺｰﾄﾞします。
5	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 9. 3. 1 clear()

port_list に設定されているポートを全て消去します。

【構文】

public void clear ()

【引数】

なし

【戻り値】

なし。

【説明】

クラスの list 配列内に含む全てのポートを消去します。
そして、count = 0 にします。

16. 9. 3. 2 set_info()

アクセスモードを設定します。

【構文】

public void set_info(int mode)

【引数】

mode

アクセスモードです。

【戻り値】

なし。

【説明】

引数に与えられたアクセスモードをプロパティに設定します。

16. 9. 3. 3 add_port()

インスタンスの port_list 配列にポート番号を 1 個追加します。

【構文】

public void add_port(int port)

【引数】

port

port_list 配列に追加するポート番号です。

【戻り値】

なし。

【説明】

list 配列に 1 個のポートを追加します。
追加した後、count を +1 します。

16. 9. 3. 4 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshCarPortAccsess dst )
```

【引数】

dst

コピー先の DshCarPortAccsess インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

16．9．3．5　decode()

S3F27 に含まれるポートアクセス変更情報を DshPortAccess クラス内のプロパティにデコードします。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

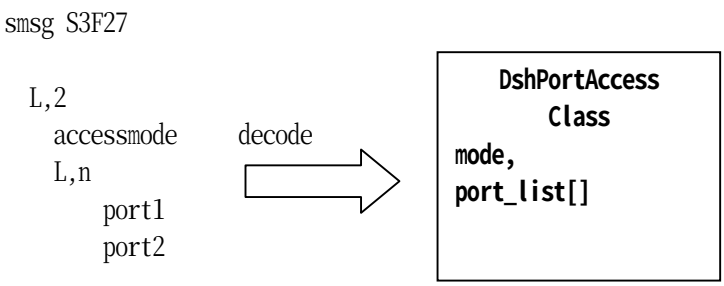
S3F27 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった（S3F27 のﾒｯｾｰｼﾞの形式が正しくなかった。）

【説明】

smsg に含まれている情報を DshPortAccess クラス内にデコードします。



メッセージに含まれるアクセスモード、ポートがクラスのプロパティに保存されます。
正常にデコードできた場合、0 を返却します。
もし、S3F27 のメッセージフォーマットが正しくないなどの理由でデコードできなかった場合、 (-1)を返却します。

16. 10 DshRCmdRsp クラス

S2F41，S2F49 の応答メッセージ S2F42，S2F50 の情報を保存するためのクラスです。

16. 10. 1 コンストラクタ

	名前	説明
1	public DshRCmdRsp()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshRCmdRsp(int hcack)	hcack を指定してｲﾝｽﾀﾝｽを生成します。

16. 10. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int hcack	S2F42，S2F50 の hcack です。
2	public int count	3 の list 配列に保存されている DshCpAck ｸﾗｽのｲﾝｽﾀﾝｽの数です。
3	public DshCpAck[] list	ｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ単位の応答情報を保存する DshCpAck ｸﾗｽの配列ﾘｽﾄです。 DshCpAck ｸﾗｽについては 16.11 で説明します。

16. 10. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	list 配列の内容を空にします。
2	public void add_ack()	1 個のｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ応答情報を追加します。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 10. 3．1 clear()

control_list 配列の内容を空にします。

【構文】

public void clear( )

【引数】

なし

【戻り値】

なし。

【説明】

list 配列の内容を空にして、count = 0 にします。

16. 10. 3. 2 add_ack()

コマンドパラメータ応答情報を１個追加します。

【構文】

public void add_ack(string cp_name, int cp_ack)

【引数】

cp_name

　　コマンドパラメータ名です。

cp_ack

　　コマンドパラメータに対する ack です。

【戻り値】

なし。

【説明】

list 配列に１個のコマンドパラメータ応答情報を追加します。
追加した後、count + 1 します。

16. 10. 3. 3 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshRCmdRsp dst)

【引数】

dst

　　コピー先の DshRCmdRsp クラスのインスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

16. 11 DshCpAck クラス

S2F41，S2F49 の応答メッセージ S2F42，S2F50 の情報を保存するためのクラスです。

ホストコマンドのコマンドパラメータ単位に対する応答情報を保存します。

16. 11. 1 コンストラクタ

	名前	説明
1	public DshCpAck(string name, int cp_ack)	ｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ名と ACK を指定してｲﾝｽﾀﾝｽを生成します。

16. 11. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string cp_name	ﾊﾟﾗﾒｰﾀ名です。(cpname)
2	public int cp_ack	cpname に対する ack です。(cpack)

16. 11. 3 メソッド

	名前	説明
1	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose() も行います。 14.1.3.13 と同様です。そちらを参照ください。

16. 12 DshCarActionRsp クラス

キャリアアクション関連 SECS-Ⅱメッセージの応答情報を保存するためのクラスです。
S3F18 メッセージが対象メッセージです。

16. 12. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshCarActionRsp()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshCarActionRsp(int ack)	ack (caack)を指定してｲﾝｽﾀﾝｽを生成します。

16. 12. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int caack	応答ﾒｯｾｰｼﾞの設定する caack です。
2	public int count	付属するｴﾗｰ情報の数です。
3	public DshObjError[] list	ｴﾗｰ情報を保存する配列です。 Vol-1 19.1 参照

16. 12. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	ｴﾗｰﾘｽﾄ list の配列を空にします。
2	public void set_count()	ｴﾗｰﾘｽﾄ list 配列のｻｲｽﾞを設定します。
3	public void add_error()	ｴﾗｰ情報(DshObjError)を１個 list に追加します。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 12. 3．1 clear()

list に設定されているエラー情報（DshObjError の内容）を全て消去します。

【構文】

public void clear ()

【引数】

なし

【戻り値】

なし。

【説明】

list に設定されているエラー情報（DshObjError の内容）を全て消去します。
そして、count = 0 にします。

16. 12. 3. 2 set_count()

list 配列に、指定されたエラー情報分のサイズの配列を作成します。

【構文】

public void set_count(int count)

【引数】

count

配列サイズです。

【戻り値】

なし。

【説明】

list の配列を引数に与えられたサイズにします。またプロパティの count に値を設定します。
本メソッドは、IntPtr で指された配列リストに格納されている複数個のエラー情報をまとめて list 配列に設定する際に使用します。エラー情報設定に使用する関数は、DshObjError.copy_err_list()です。

list にエラー情報を設定したい場合は、通常、次に説明する add_error()メソッドを使ってください。

16. 12. 3. 3 add_err()

インスタンスの list 配列にエラー情報を 1 個追加します。

【構文】

public void add_error(int code, string text)

【引数】

code

エラーコードです。

text

エラーテキストです。

【戻り値】

なし。

【説明】

list 配列に、1 個のエラー情報を DshObjError のクラスに設定し、追加します。
追加した後、count を +1 します。

16. 13 DshPortActionRsp クラス

ポートアクション関連 SECS-Ⅱメッセージの応答情報を保存するためのクラスです。
S3F18 メッセージが対象メッセージです。

16. 13. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshPortActionRsp()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshPortActionRsp(int ack)	ack (caack)を指定してｲﾝｽﾀﾝｽを生成します。

16. 13. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int caack	応答ﾒｯｾｰｼﾞの設定する caack です。
2	public int count	付属するｴﾗｰ情報の数です。
3	public DshObjError[] list	ｴﾗｰ情報を保存する配列です。 Vol-1 19.1 参照

16. 13. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	ｴﾗｰﾘｽﾄ list の配列を空にします。
2	public void set_count()	ｴﾗｰﾘｽﾄ list 配列のｻｲｽﾞを設定します。
3	public void add_error()	ｴﾗｰ情報(DshObjError)を１個 list に追加します。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 13. 3．1 clear()

list に設定されているエラー情報（DshObjError の内容）を全て消去します。

【構文】

public void clear ()

【引数】

なし

【戻り値】

なし。

【説明】

list に設定されているエラー情報（DshObjError の内容）を全て消去します。
そして、count = 0 にします。

16. 13. 3. 2 set_count()

list 配列に、指定されたエラー情報分のサイズの配列を作成します。

【構文】

public void set_count(int count)

【引数】

count

配列サイズです。

【戻り値】

なし。

【説明】

list の配列を引数に与えられたサイズにします。またプロパティの count に値を設定します。
本メソッドは、IntPtr で指された配列リストに格納されている複数個のエラー情報をまとめて list 配列に設定する際に使用します。エラー情報設定に使用する関数は、DshObjError.copy_err_list()です。

list にエラー情報を設定したい場合は、通常、次に説明する add_err()メソッドを使ってください。

16. 13. 3. 3 add_err()

インスタンスの list 配列にエラー情報を 1 個追加します。

【構文】

public void add_error(int code, string text)

【引数】

code

エラーコードです。

text

エラーテキストです。

【戻り値】

なし。

【説明】

list 配列に、1 個のエラー情報を DshObjError のクラスに設定し、追加します。
追加した後、count を +1 します。

16. 14 DshAccessError クラス

エラー情報を保存するためのクラスです。

16. 14. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshAccessError()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshAccessError(int port, int code, string text)	ﾎﾟｰﾄ番号、ｴﾗｰｺｰﾄﾞとｴﾗｰﾃｷｽﾄを指定して生成します。

16. 14. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int port	ﾎﾟｰﾄ番号です。
2	public int errcode	ｴﾗｰｺｰﾄﾞです。
3	public string errtext	ｴﾗｰﾃｷｽﾄです。

16. 14. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_err()	value に使用されているﾒﾓﾘを開放します。
2	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 14. 3．1 set_err ()

ポート番号、エラーコードとｴﾗｰテキストを設定します。

【構文】

public void set_obj_err(int port, int code, string text)

【引数】

code

エラーコードです。

text

エラーテキストです。

【戻り値】

なし。

【説明】

ポート番号、エラーコードとエラーテキストを設定します。

16. 15 DshPortAccessRsp クラス

ポートアクセス変更関連 SECS-Ⅱメッセージの応答情報を保存するためのクラスです。
S3F18 メッセージが対象メッセージです。

16. 15. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshPortAccessRsp()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshPortAccessRsp(int ack)	ack (caack)を指定してｲﾝｽﾀﾝｽを生成します。

16. 15. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int caack	応答ﾒｯｾｰｼﾞの設定する caack です。
2	public int count	付属するｴﾗｰ情報の数です。
3	public DshAccessError[] list	ｴﾗｰ情報を保存する配列です。 Vol-1 16.13 参照

16. 15. 3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	ｴﾗｰﾘｽﾄ list の配列を空にします。
2	public void add_error()	ｴﾗｰ情報(DshAccessError)を 1 個 list に追加します。
3	public void add_error()	ｴﾗｰ情報(DshAccessError)を 1 個 list に追加します。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

16. 15. 3．1　clear()

list に設定されているエラー情報（DshAccessError の内容）を全て消去します。

【構文】

public void clear ()

【引数】

なし

【戻り値】

なし。

【説明】

list に設定されているエラー情報（DshAccessError の内容）を全て消去します。
そして、count = 0 にします。

16. 15. 3. 2 add_err()

インスタンスの list 配列にエラー情報を 1 個追加します。

【構文】

public void add_error(int port, int code, string text)

【引数】

port

ポート番号です。

code

エラーコードです。

text

エラーテキストです。

【戻り値】

なし。

【説明】

list 配列に、1 個のエラー情報を DshAccessError のクラスに設定し、追加します。
追加した後、count を +1 します。

17. 端末表示メッセージ関連クラス

端末表示メッセージ関連情報を保存するクラスです。

関連クラスの一覧を次表に示します。

	クラス名	用途
1	DshTermMsg	端末ﾒｯｾｰｼﾞ保存用ｸﾗｽです。 S10F1，S10F3 に使用されます。
2	DshTermMultiMsg	端末複数ﾒｯｾｰｼﾞ保存用ｸﾗｽです。 S10F5 に使用されます

通信関連クラスとして、以下のものがあります。 これらについては、Vol-2 で説明します。

DshS10F1Send
DshS10F2Response
DshS10F3Send
DshS10F4Response
DshS10F5Send
DshS10F6Response

17. 1 DshTermMsg クラス

端末メッセージ情報のためのクラスであり、S10F1, S10F3 メッセージの送受信時に使用されます。

17. 1. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshTermMsg()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshTermMsg(int tid, string text)	端末 ID と表示ﾃｷｽﾄを指定してｲﾝｽﾀﾝｽを生成します。

17. 1. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int tid	端末 ID です。
2	public string_text	端末に表示する文字列です。

17．1．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_info()	端末 ID と表示文字列を設定します。
2	public int decode()	S10F1 または S10F3 ﾒｯｾｰｼﾞの情報を当該ｸﾗｽに保存します。
3	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

17．1．3．1　set_info()

端末 ID と表示文字列（1 行）をプロパティ tid，text に設定します。

【構文】

public void set_info(int tid, string text)

【引数】

tid

端末 ID です。

text

表示文字列です。

【戻り値】

なし。

【説明】

端末 ID と表示文字列（1 行）をプロパティ tid，text に設定します。

17．1．3．2　decode()

S10F1 または S10F3 に含まれる情報を DshTermMsg クラス内のプロパティにデコードします。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

S10F1 または S10F3 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

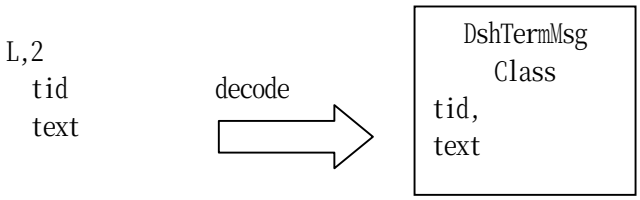
【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった （S10F1 または S10F3 のﾒｯｾｰｼﾞの形式が正しくなかった。

【説明】

smsg に含まれているレポート情報を DshTermMsg クラス内にデコードします。

smsg S10F1 または S10F3



メッセージに含まれる端末表示情報は、クラスのプロパティに保存されます。

正常にデコードできた場合、0 を返却します。

もし、S10F1 または S10F3 のメッセージフォーマットが正しくないなどの理由でデコードできなかった場合、(-1) を返却します。

17. 2 DshTermMultiMsg クラス

複数行の表示文字列を含む端末メッセージ情報のためのクラスであり、S10F5 メッセージの送受信時に使用されます。

17. 2. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshTermMultiMsg()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshTermMultiMsg(int tid)	端末 ID を指定してｲﾝｽﾀﾝｽを生成します。

17. 2. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int tid	端末 ID です。
1	public int count	設定されている文字列の数です。
2	public string[] list	string ｸﾗｽの配列であり、表示文字列を保存する配列です。

17. 2. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_tid()	端末 ID を設定します。
2	public void clear()	list 配列を空にします。
3	public void add()	list 配列に 1 個の文字列を加えます。
4	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
5	public int decode()	S10F5 ﾒｯｾｰｼﾞ の情報を当該ｸﾗｽに保存します。
6	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

17. 2. 3. 1 set_tid()

端末 ID をプロパティ tid に設定します。

【構文】

public void set_tid(int tid)

【引数】

tid

端末 ID です。

【戻り値】

なし。

【説明】

端末 ID をプロパティ tid に設定します。

17. 2. 3. 2 clear()

list 配列を空にします。

【構文】
public void clear( )

【引数】
なし

【戻り値】
なし。

【説明】
list 配列リストを空にします。そして count = 0 にします。

17. 2. 3. 3 add()

インスタンスの list 配列に文字列を 1 個追加します。

【構文】
public void add(string data)

【引数】
data
追加したい文字列です。

【戻り値】
なし。

【説明】
list 配列に 1 個のデータアイテムを追加します。
そして、count + 1 します。

17．2．3．4　copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshStrList dst )
```

【引数】

dst

コピー先の DshStrList インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

17．2．3．5　decode()

S10F5 に含まれる情報を DshTermMultiMsg クラス内のプロパティにデコードします。

【構文】

public int decode(ref DSHMSG smsg)

【引数】

smsg

S10F5 のメッセージ情報（生情報）が格納されている DSHMSG 構造体領域になります。
DSHMSG は、１次メッセージをポーリングした際にエンジンから与えられる情報です。
ユーザは DSHMSG 構造体については、特に意識しないで、ポーリングした後、本メソッドに渡すだけです。

【戻り値】

返却値	意　味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾃﾞｺｰﾄﾞできなかった （S10F5 のﾒｯｾｰｼﾞの形式が正しくなかった。

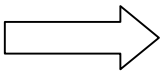
【説明】

smsg に含まれているレポート情報を DshTermMultiMsg クラス内にデコードします。

smsg S10F5

```
L,2
  tid
  L,n
    text1
    text2
      .
```

decode



```
DshTermMultiMsg
      Class
tid,
list[]
```

メッセージに含まれる端末表示情報は、クラスのプロパティに保存されます。

正常にデコードできた場合、0 を返却します。

もし、S10F5 のメッセージフォーマットが正しくないなどの理由でデコードできなかった場合、 (-1)を返却します。

18．オブジェクト属性、パラメータ情報関連クラス

本章では、メッセージに付属するオブジェクトの属性(attribute)、パラメータ情報を保存するためのクラスについて説明します。

関連クラスの一覧を次表に示します。

	クラス名	用途
1	DshObjPara	ｵﾌﾞｼﾞｪｸﾄ関連ｸﾗｽのﾊﾟﾗﾒｰﾀ（属性）情報を保存するためのｸﾗｽです。 S2F41，S2F49，S3F17，S3F23，S3F25，S14F9,11，S15F13 などのﾒｯｾｰｼﾞが含んでいる情報を保存します。
2	DshObjSubPara	DshObjPara の情報が中で多重化していた際に、そのｻﾌﾞﾊﾟﾗﾒｰﾀを保存するためのｸﾗｽです。 S2F41 のﾒｯｾｰｼﾞ情報保存に使用されます。
3	DshObjParaList	DshObjPara の配列ﾘｽﾄのｸﾗｽです。
4	DshDataItem	SECS-Ⅱのﾃﾞｰﾀｱｲﾃﾑの情報を１個保存するためのｸﾗｽです。
5	DshDataItemList	DshDataItem の配列ｸﾗｽです。
6	DshStrList	文字列(string)の配列ｸﾗｽです。

18. 1 DshObjPara クラス

オブジェクトのパラメータ（属性）情報を保存するためのクラスです。

(注) コントロールジョブ属性情報の保存には、DshCjAttr クラスを使います。

18. 1. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshObjPara()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshObjPara(string name, int format, int size, IntPtr value)	引数としてパラメータ ID，ﾌｫｰﾏｯﾄ、size，値が格納されているﾎﾟｲﾝﾀ(IntPtr)が指定されます。
3	public DshObjPara(string name, string value)	引数としてパラメータ ID，と文字列値が指定されます。

18. 1. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string name	パラメータ名です。
2	public int format	パラメータ値のﾌｫｰﾏｯﾄです。 (HSMS.ICODE_U1 など)
3	public int size	パラメータ値の配列ｻｲｽﾞ（ﾃﾞｰﾀ数、基本的に ICODE_A 以外は size=1 です。
4	public IntPtr value	パラメータ値を格納する非管理ﾒﾓﾘのﾎﾟｲﾝﾀです。

18．1．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void free()	value に使用されているﾒﾓﾘを開放します。
2	public void set_para()	パラメータ情報を設定します。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

18．1．3．1　free()

インスタンスの value に使用されている非管理メモリを開放します。

【構文】

public void free( )

【引数】

なし

【戻り値】

なし。

【説明】

パラメータ値 value に使用されている非管理メモリを開放します。

18. 1. 3. 2 set_para ()

パラメータ名とパラメータ値を設定します。

【構文】

```
public void set_para(string name, int format, int size, IntPtr value)
public void set_para(string name, string value)
```

【引数】

name
　　パラメータ名です。
format
　　パラメータ値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　パラメータ値の配列サイズです。
value
　　設定したいパラメータ値が格納されているポインタです。

【戻り値】

なし。

【説明】

パラメータ情報を DshObjPara クラスに設定します。
value が string で与えられるメソッドの場合、HSMS.ICODE_A のフォーマットで設定します。

18. 1. 3. 3 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshObjPara dst )
```

【引数】

dst
　　コピー先の DshObjPara インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

18. 2 DshObjSubPara クラス

オブジェクトのサブパラメータ（属性）情報を保存するためのクラスです。
DshObjPara クラスのパラメータ値が ICODE_L(リスト構造)のとき、ネスティングした複数のパラメータ値を保存するためのクラスです。

18. 2. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshObjSubPara()	空のｲﾝｽﾀﾝｽを生成します。

18. 2. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	ﾊﾟﾗﾒｰﾀ値の数です。
2	public DshDataItem[] list	ﾊﾟﾗﾒｰﾀ値を保存する DshDataItem ｸﾗｽの配列です。

18. 2. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void free()	value に使用されているﾒﾓﾘを開放します。
2	public void add()	list 配列にﾃﾞｰﾀを 1 個追加します。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

18. 2. 3. 1 clear()

クラス内の list 配列に設定されているデータを全て消去します。

【構文】

public void clear ()

【引数】

なし

【戻り値】

なし。

【説明】

クラスの list 配列内に含む全てのデータを消去します。
非管理メモリを使用しているパラメータ値についてはメモリを開放します。
そして、count = 0 にします。

18．2．3．2　add()

パラメータ値を配列に1個追加します。

【構文】

```
public void add(int format, int size, IntPtr value)
public void add(string value)
public void add(DshDataItem data)
```

【引数】

format
　　パラメータ値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　パラメータ値の配列サイズです。
value
　　設定したいパラメータ値が格納されているポインタです。
data
　　パラメータ値が保存されている DshDataItem クラスのインスタンスです。

【戻り値】

なし。

【説明】

パラメータ値を DshObjSubPara クラスの list 配列に追加します。
value が string で与えられるメソッドの場合、HSMS.ICODE_A のフォーマットで設定します。

18．2．3．3　copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshObjSubPara dst )
```

【引数】

dst
　　コピー先の DshObjSubPara インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

18. 3 DshDataItem クラス

1 個の SECS データアイテムのデータを保存するためのクラスです。

18. 3. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshDataItem()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshDataItem(int format, int size, IntPtr value)	ﾃﾞｰﾀのﾌｫｰﾏｯﾄ、ｻｲｽﾞ、値を指定してｲﾝｽﾀﾝｽを生成します。

18. 3. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int format	ﾃﾞｰﾀｱｲﾃﾑのﾌｫｰﾏｯﾄです。 HSMS.ICODE_U1 など
2	public int size	ﾃﾞｰﾀｱｲﾃﾑの配列ｻｲｽﾞです。 HSMS.ICODE_A,J では、文字列長になります。 HSMS.ICODE_L では、ﾘｽﾄｻｲｽﾞになります。 その他のﾌｫｰﾏｯﾄでは、１です。
3	public IntPtr value	ﾃﾞｰﾀ値が保存されているﾒﾓﾘのﾎﾟｲﾝﾀです。 非管理ﾒﾓﾘです。

18．3．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void free()	value に使用されているﾒﾓﾘを開放します。
2	public void set_data()	ﾃﾞｰﾀを設定します。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

18．3．3．1　free()

クラス内の value に使用されている非管理メモリを開放します。

【構文】

public void free()

【引数】

なし

【戻り値】

なし。

【説明】

value に使用されている非管理メモリを開放します。
value を IntPtr.Zero にします。

18. 3. 3. 2 set_data()

パラメータ値を配列に1個追加します。

【構文】

```
public void set_data(int format, int size, IntPtr value)
public void set_data( string value)
```

【引数】

format
　　データのフォーマットです。(HSMS.ICODE_U1 など)
size
　　データの配列サイズ
value
　　設定したいメータが格納されているポインタです。
　　string の場合は文字列です。

【戻り値】

なし。

【説明】

データを設定します。
value が string で与えられるメソッドの場合、HSMS.ICODE_A のフォーマットで設定します。

18. 3. 3. 3 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

```
public void copy( ref DshDataItem dst )
```

【引数】

dst
　　コピー先の DshDataItem インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

18. 4 DshDataItemList クラス

複数個のデータアイテムをリストで保存するための DshDataItem クラスの配列です。

18. 4. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshDataItemList()	空のｲﾝｽﾀﾝｽを生成します。 配列ｻｲｽﾞは=0 になります。

18. 4. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	設定されているﾃﾞｰﾀｱｲﾃﾑ値の数です。
2	public DshDataItem[] data_list	DshDataItem ｸﾗｽの配列です。

18. 4. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	data_list 配列を空にします。
2	public void add_data()	data_list 配列に 1 個のデータアイテム値を加えます。
3	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

18. 4. 3. 1 clear()

インスタンスの data_list 配列の中に含まれるデータアイテムの value に使用されている非管理メモリを開放し、data_list リストを空にします。

【構文】

public void clear( )

【引数】

なし

【戻り値】

なし。

【説明】

data_list 配列のなかのデータアイテム値value に使用されている非管理メモリを開放し、リストを空にします。
そして count = 0 にします。

18．4．3．2　add_data()

インスタンスの data_list 配列にデータアイテム値情報を１個追加します。

【構文】

public void add_data(int format, int size, IntPtr value)
public void add_data(ref DshDataItem data)

【引数】

format
　　データアイテム値のデータフォーマットです。(HSMS.ICODE_U1 など)
size
　　データアイテム値の配列サイズです。
value
　　設定したいデータアイテム値が格納されているポインタです。
data
　　データアイテムが保存されている DshDataItem クラスのインスタンスです。

【戻り値】

なし。

【説明】

data_list 配列に１個のデータアイテムを追加します。
value, data に使用されている IntPtr のメモリは新規に非管理メモリを取得して保存します。
そして、count + 1 します。

18. 5 DshStrList クラス

複数個の文字列を保存するための string クラスの配列です。

18. 5. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshStrList()	空のｲﾝｽﾀﾝｽを生成します。 配列ｻｲｽﾞは=0 になります。

18. 5. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int count	設定されている文字列の数です。
2	public string[] list	string ｸﾗｽの配列です。

18. 5. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void clear()	list 配列を空にします。
2	public void add()	list 配列に 1 個の文字列を加えます。
3	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

18. 5. 3. 1 clear()

list 配列を空にします。

【構文】

public void clear( )

【引数】

なし

【戻り値】

なし。

【説明】

list 配列リストを空にします。そして count = 0 にします。

18. 5. 3. 2 add()

インスタンスの list 配列に文字列を１個追加します。

【構文】

public void add(string data)

【引数】

data

追加したい文字列です。

【戻り値】

なし。

【説明】

list 配列に１個のデータアイテムを追加します。
そして、count + 1 します。

18. 5. 3. 3 copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshStrList dst)

【引数】

dst

コピー先の DshStrList インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

19. エラー情報関連クラス

本章では、応答メッセージに使用されるエラー情報を保存するクラスについて説明します。

関連クラスの一覧を次表に示します。

	クラス名	用途
1	DshObjError	ｵﾌﾞｼﾞｪｸﾄ関連ﾒｯｾｰｼﾞに対する応答ﾒｯｾｰｼﾞに含まれるｴﾗｰ情報を保存するためのｸﾗｽです。 S3F18，S14F10，S15F14､S16F12 などの応答ﾒｯｾｰｼﾞが含んでいる情報を保存します。

19. 1 DshObjError クラス

エラー情報を保存するためのクラスです。

19. 1. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshObjError()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshObjError(int code, string text)	ｴﾗｰｺｰﾄﾞとｴﾗｰﾃｷｽﾄを指定して生成します。
3	public DshObjError(int code, IntPtr text)	ｴﾗｰｺｰﾄﾞと IntPtr で与えられるﾃｷｽﾄを指定っして生成します。

19. 1. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int errcode	ｴﾗｰｺｰﾄﾞです。
2	public string errtext	ｴﾗｰﾃｷｽﾄです。

19．1．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_obj_err()	ｴﾗｰ情報を設定します。
2	public void copy()	当該ｲﾝｽﾀﾝｽの内容を別のｲﾝｽﾀﾝｽにｺﾋﾟｰします。
3	public static void copy_err_list()	IntPtr で示されるｵﾌﾞｼﾞｪｸﾄｴﾗｰ情報をﾌﾞﾛｯｸｺﾋﾟｰします。
4	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

19．1．3．1　set_obj_err ()

エラーコードとｴﾗｰテキストを設定します。

【構文】

public void set_obj_err(int code, string text)
public void set_obj_err(int code, IntPtr text)

【引数】

code
　　エラーコードです。
text
　　エラーテキストです。

【戻り値】

なし。

【説明】

エラーコードとエラーテキストを設定します。
IntPtr で与えられた text は string に変換して設定します。

19．1．3．2　copy()

当該クラスのインスタンスの内容を他のインスタンスにコピーします。

【構文】

public void copy(ref DshObjError dst)

【引数】

dst

コピー先の DshObjError インスタンスです。

【戻り値】

なし。

【説明】

当該インスタンスの内容（プロパティ値）を dst に指定されるインスタンスにコピーします。

19．1．3．3　copy_err_list()

IntPtr のポインタで渡される 1 個以上のオブジェクトエラー情報を DshObjError クラスの配列にブロックでコピーします。

【構文】

public static void copy_err_list(IntPtr err_ptr_list, ref DshObjError[] list, int count)

【引数】

err_ptr_list

1 個以上のオブジェクトエラー情報が格納されている構造体のポインタ配列です。
TERR_INFO 構造体のポインタが格納されている配列のポインタです。

【戻り値】

なし。

【説明】

IntPtr のポインタで渡される 1 個以上のオブジェクトエラー情報を DshObjError クラスの配列にブロックでコピーします。
本クラスライブラリのユーザは、ほとんど使用することがないメソッドです。
本メソッドは、static 宣言されていますので、DshObjError クラスのインスタンスを生成しないで使用することができます。(DshObjError.copy_err_list(...) のように)

20．データアイテム関連、エラー情報コピークラス

SECS-Ⅱメッセージに使用するデータアイテムの形式変換などを行う以下のクラスについて説明します。

	クラス名	用途
1	DshLib	ﾃﾞｰﾀｱｲﾃﾑの形式変換などを行うためのｸﾗｽです。 ﾃﾞｰﾀｱｲﾃﾑ値 から文字列への変換 文字列からﾃﾞｰﾀｱｲﾃﾑ値への変換

20．１ DshLib クラス

ここで述べる本クラスに属しているメソッドは全て static 関数になっています。

文字列からデータアイテムへの変換、データアイテム値から文字列への変換するためのメソッド（関数）が準備されています。本クラスに属する関数は、全て static で宣言されていますので、特に DshLib のインスタンスを生成しないで使用することができます。

20．１．１ コンストラクタ

ありません。

20．１．２ プロパティ

ありません。

20. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public static string DshPtrToFmtDataString()	ﾃﾞｰﾀｱｲﾃﾑのﾃﾞｰﾀを文字列に変換します。
2	public static IntPtr StringToFormatData()	文字列ﾃﾞｰﾀを指定ﾌｫｰﾏｯﾄのﾃﾞｰﾀに変換し、非管理ﾒﾓﾘに格納してその IntPtr のﾎﾟｲﾝﾀを返却します。
3	public static int DshPtrFmtDataInt()	ﾃﾞｰﾀﾌｫｰﾏｯﾄが 32 ﾋﾞｯﾄ整数で表現できるﾌｫｰﾏｯﾄの場合、int 型の値に変換します。
4	public static IntPtr PtrToFormatData()	IntPtr で与えられるﾃﾞｰﾀｱｲﾃﾑの値を、別の非管理ﾒﾓﾘに格納し、そのﾎﾟｲﾝﾀを返します。
5	public static IntPtr DataToPtr()	数値ﾃﾞｰﾀをﾃﾞｰﾀﾌｫｰﾏｯﾄにあわせて IntPtr ﾒﾓﾘに格納し、それを返却します。
6	public static void StartPerformanceTimer()	ﾊﾟﾌｫｰﾏﾝｽﾀｲﾏｰを開始します。 処理時間計測の開始です。
7	public static float StopPerformanceTimer()	ﾊﾟﾌｫｰﾏﾝｽﾀｲﾏｰを停止します。 処理時間を取得できます。
8	public static void Beep()	ﾋﾞｰﾌﾟ音を鳴らします。
9	public static string DshPtrToString()	HSMS.D_GetItem()で得られた ICODE_A の情報を string 型に変換するための関数です。
10	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)ｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 14.1.3.13 と同様です。そちらを参照ください。**

20．1．3．1　DshPtrToFmtDataString()

IntPtr で与えられるデータアイテムの値を文字列に変換します。

【構文】

public static string DshPtrToFmtDataString(IntPtr ptr, int fmt, int size)

【引数】

ptr

fmt で指定されたデータが格納されているポインタです。

fmt

ptr に格納されている値のフォーマットです。(HSMS.ITEM_U1 など)

size

ptr に格納されているデータの数です。(HSMS.ICODE_L のときだけ使用します。)

【戻り値】

変換された文字列を返却します。

【説明】

全データアイテムコードに対する変換を行います。

以下のように変換処理を行います。

（1）fmt が ICODE_L の場合は、size の値を文字列に変換します。

（2）その他のフォーマットの場合、もし、ptr が =IntPtr.Zero の場合、文字列 ”(null)” を返却します。

（3）ICODE_A の場合は、そのまま、ansi コードの文字列に変換します。

（4）ICODE_A 以外のものについて size=0 の場合、0 長文字列 ”” を返却します。
size > 0 であれば、先頭のデータ 1 個だけをフォーマットにあわせて文字列に変換します。

20．1．3．2　StringToFormatData()

文字列で表現されているデータを指定フォーマットの値に変換し、それを確保した非管理メモリに格納し、そのIntPtr のポインタを返却します。

【構文】

public static IntPtr StringToFormatData(string str, int fmt)

【引数】

str

データが表現されている文字列です。

fmt

変換したいデータのフォーマットです。(HSMS.ITEM_U1 など)

【戻り値】

文字列から変換された非管理メモリのポインタを返却します。

【説明】

全データアイテムコードに対する変換を行います。

以下のように変換処理を行います。

（1）str が=""(null)で、ICODE_A, ICODE_J 以外のフォーマットであれば、str="0"にして変換します。

（2）以下、指定フォーマットに必要なサイズのメモリを非管理メモリから確保し、変換した値をそのメモリに格納します。

（3）(2)で確保したメモリの IntPtr によるポインタを返却します。

20．１．３．３　DshPtrFmtDataInt ()

IntPtr で与えられるデータアイテムの値を３２ビット整数に変換します。

【構文】

public static int DshPtrFmtDataInt(int fmt, IntPtr ptr)

【引数】

fmt

ptr に格納されている値のフォーマットです。(HSMS.ICODE_U1 など)
適用フォーマットは、ICODE_L, ICODE_B, ICODE_BOOL, ICODE_U1, ICODE_U2, ICODE_U4, ICODE_I1, ICODE_I2
ICODE_I4 です。(それ以外のフォーマットでは、返却値が=0 になります。)

ptr

fmt で指定されたデータが格納されているポインタです。

【戻り値】

変換結果を整数(int)で返却します。
ICODE_A,J,F4,F8 の場合は、=0 を返却します。

【説明】

IntPtr が指すメモリに格納されている数値データを int 型に変換して返却します。

適用フォーマットは、ICODE_L, ICODE_B, ICODE_BOOL, ICODE_U1, ICODE_U2, ICODE_U4, ICODE_I1, ICODE_I2
ICODE_I4 です。

それ以外のフォーマットでは、0 を返却します。

使用する際は、予め適用するフォーマットであることを前提に使用してください。

20．１．３．４　PtrToFormatData()

IntPtr で与えられるデータアイテムの値を、別の非管理メモリに格納し、そのポインタを返します。

【構文】

public static IntPtr PtrToFormatData(IntPtr ptr, int fmt, int size)

【引数】

ptr

fmt で指定されたデータが格納されているポインタです。

fmt

ptr に格納されている値のフォーマットです。(HSMS.ICODE_U1 など)

size

データの配列サイズです。

【戻り値】

新しく確保した非管理メモリのポインタを返却します。
引数の ptr が = IntPtr.Zero の場合は、IntPtr.Zero を返却します。

【説明】

IntPtr が指すメモリに格納されているデータと同じサイズの非管理メモリを確保し、引数で ptr に与えられたデータをコピーし、新たに確保した非管理メモリのポインタを返却します。

もし、ptr が IntPtr.Zero であった場合は、IntPtr.Zero を返却します。

20．1．3．5 DataToPtr()

数値データを非管理メモリに格納し、そのポインタを返します。

【構文】

```
public static IntPtr DataToPtr(byte val)
public static IntPtr DataToPtr(byte val)
public static IntPtr DataToPtr(short val)
public static IntPtr DataToPtr(int val)
public static IntPtr DataToPtr(long val)
public static IntPtr DataToPtr(float val)
public static IntPtr DataToPtr(double val)
```

【引数】

val

データ数値です。

【戻り値】

val が格納された非管理メモリのポインタが返却されます。

【説明】

引数の val にあわせて、値を格納するための非管理メモリを確保し、そこに値を格納し、非管理メモリのポインタを返却します。

20．1．3．6　StartPerformanceTimer()

プログラムの処理時間を計測するタイマーを開始します。
計測する時間の単位は、秒で、float 型の値を得ることができます。

【構文】
public static void StartPerformanceTimer(int index)

【引数】
index
　　計測タイマーのインデクスです。0～7 の値の範囲です。

【戻り値】
なし。

【説明】
index で指定されたタイマーを開始します。
停止は、次に説明するメソッド StopPerformanceTimer()を使用します。計測時間が取得できます。

20．1．3．7　StopPerformanceTimer()

StartPerformanceTimer()で開始したタイマーを停止し、開始してから停止するまでの経過時間を秒単位の、float 型の値で取得します。

【構文】
public static float StopPerformanceTimer(int index)

【引数】
index
　　計測タイマーのインデクスです。0～7 の値の範囲です。

【戻り値】
開始から停止までの経過時間を float 型の秒単位の値をが返却します。
時間の分解能は、float の小数点部になります。

【説明】
StartPerformanceTimer()で開始したタイマーを停止し、開始してから停止するまでの経過時間を秒単位の、float 型の値で取得します。

20．1．3．8　beep()

ビープ音を鳴らします。

【構文】

public static void Beep(int a, int b)

【引数】

a

　　鳴らす回数です。

b

　　鳴らすインターバルです。(ms 単位)

【戻り値】

なし。

【説明】

周波数 2000HZ で、継続時間 600ms の音を b で指定された間隔で、a 回鳴らします。
その後、戻ります。

20．1．3．9 DshPtrToString ()

IntPtr メモリ内の文字列データを string 型の文字列に変換します。
指定の文字列位置に NULL 文字を設定し、string 型の文字列を返します。

【構文】

public static string DshPtrToString(IntPtr ptr, int max_size, int fixed_len)

【引数】

ptr

文字列情報が格納されているメモリのポインタです。

max_size

ptr のメモリサイズです。

fixed_len

NULL 文字をセットする配列位置です。

【戻り値】

NULL 文字がセットされた後の文字列が返却されます。

【説明】

ptr メモリの fixed_len 位置に NULL 文字をセットします。
もし、fixed_len >= max_size であれば、NULL 文字をセットしません。

NULL 文字セットを行った後、string 型の文字列データに変換して、それを返却します。

本関数は、SECS-Ⅱメッセージから 文字列（HSMS.ICODE_A）のデータアイテムを IntPtr メモリに HSMS.D_GetItem()関数によって取得した後､得られたデータアイテム長にあわせて string 型文字列に変換するときに使用することができます。

21. DshDebug - クラス・トレースのためのクラス

クラスライ・ブラリに含まれるクラスの生成、資源開放、消滅３つの事象の発生回数をカウントするとともに、事象発生時にそれらの情報をリアルタイムで、ユーザが提供する Windows Form に表示させる処理を行うためのクラスです。

説明の中で、クラスの生成、資源開放、消滅の３つの事象のことを**トレース事象**と呼ぶことにします。

21. 1 DshDebug クラス

本クラスは、他のクラスのトレース事象(Construct, Dispose, Finalize)の発生回数を管理し、トレースが有効になっているクラスについてはそのトレース事象発生を、ユーザの Windows フォームに通知する役割を果たします。

本クラスが有するメソッドは全て static 関数になっています。したがって、ユーザが本クラスのメソッドを呼び出す場合、 **DshDebug.**<method>のように、DshDebug クラスを直接呼び出すことになります。

なお、クラスのトレース表示機能については、次のドキュメントを参照ください。

「文書番号 DSHGEM-07-30360-00 クラス生成・消滅トレースと表示機能について」

本トレース機能を使用するためには、以下の処理を行う必要があります。

（1）init_DshDebug()で初期設定する。
(DshEngine が生成されるときに自動的に呼び出されます。)

（2）ユーザが用意するクラス・トレース表示用の Windows Form の情報を設定する。
Form ハンドルと、その Form にトレース情報を送信するための Windows Message ID です。

（3）DshDebug クラスのトレース表示有効設定（クラス全体）

（4）トレースしたいクラスの有効設定（クラス個別）

なお、トレース表示とは別に、トレース事象のカウンター値は（2）～（3）の設定に関係なく、いつでも取得することができます。

21. 1. 1 コンストラクタ

使用しません。

21. 1. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public constant int DBG_xxxx	ﾕｰｻﾞが参照できるｸﾗｽのｲﾝﾃﾞｸｽ名です。 xxxx の部分がｸﾗｽ名です。 ｲﾝﾃﾞｸｽを必要とするﾒｿｯﾄﾞに引数として説明します。 例 DshAlarm ｸﾗｽのｲﾝﾃﾞｸｽ名は DBG_DshAlarm になります。
2	private な情報	これらは、ﾒｿｯﾄﾞを通して設定または取得できます。 formHandle, form への Windows Message の設定 ﾄﾚｰｽ有効/無効ﾌﾗｸﾞ ﾄﾚｰｽｶｳﾝﾀ, ｸﾗｽ名 ﾄﾚｰｽｶｳﾝﾀｰには、Construct, Dispose, Finalize の ３つのｶｳﾝﾀｰがあります。

21. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public static void init_DshDebug()	DshDebug ｸﾗｽ内部の情報をｾｯﾄｱｯﾌﾟします。 本ﾒｿｯﾄﾞは、DshEngine ｸﾗｽが生成されたときに自動的に実行されます。
2	public static void set_trace_form_handle ()	ﾕｰｻﾞがﾄﾚｰｽ表示出力したい Windows Form のﾊﾝﾄﾞﾙを指定します。
3	public static void set_trace_WM()	上の2.で指定した Form に通知して貰う Windows Message ID を指定します。
4	public static void enable_trace_log()	DshDebug のﾄﾚｰｽ表示通知の有効/無効設定をします。(全体)
5	public static string get_class_name()	ｲﾝﾃﾞｸｽで指定したｸﾗｽの名前を取得します。
6	public static bool set_trace_flag()	ｸﾗｽを指定してﾄﾚｰｽ表示通知の有効/無効を設定します。
7	public static bool get_trace_flag()	ｸﾗｽを指定してﾄﾚｰｽ表示通知の有効/無効設定状態を取得します。
8	public static void reset_trace_count()	ｸﾗｽを指定して、そのｸﾗｽのﾄﾚｰｽ事象ｶｳﾝﾀｰを 0 にﾘｾｯﾄします。
9	public static int get_trace_count()	ｸﾗｽを指定して、ﾄﾚｰｽ事象ｶｳﾝﾀｰ値を取得します。 （ﾄﾚｰｽ事象の３つのｶｳﾝﾀｰ値を取得します。）

21．1．3．1　init_DshDebug()

クラスのトレース機能を開始するための準備をします。
DshEngine クラスが生成された際、本メソッドが未実行状態である場合、自動的に実行してくれます。

クラス名の設定、カウンターのリセットなどを行います。

【構文】

public static void init_DshDebug()

【引数】

なし。

【戻り値】

なし。

【説明】

本メソッドはクラスのトレース処理のために必ず実行する必要があります。

DshEngine クラスが生成されたときに、本メソッドが未実行状態であれば自動的に本メソッドを実行します。

ユーザプログラムからは、DshEngine クラスの生成（new キーワードによる）の前に実行することもできます。

本メソッドが実行する処理は、トレース対象となる全クラスの名前情報の設定、トレース事象カウンター(Construct, Dispose, Finalize カウンタ)の初期化などです。

21．1．3．2　set_trace_form_handle ()

トレースしたクラスの内容を表示通知するための宛先Windows フォームのハンドルを設定します。

ユーザに対するトレース表示情報は、ここで与えられたハンドルのフォームへ通知されます。

【構文】

public static void set_trace_form_handle(IntPtr handle)

【引数】

handle

Windows フォームのハンドルを指定します。

【戻り値】

なし。

【説明】

クラスのトレース事象の発生通知する相手の Windows Form のハンドルを設定します。

21．1．3．3　set_trace_WM()

先に説明した set_trace_form_handle()で指定されたフォームにトレース情報を通知するときに使用する Windows Message の ID を設定します。

【構文】

public static void set_WM(int wm)

【引数】

wm

Windows Message の ID です。

【戻り値】

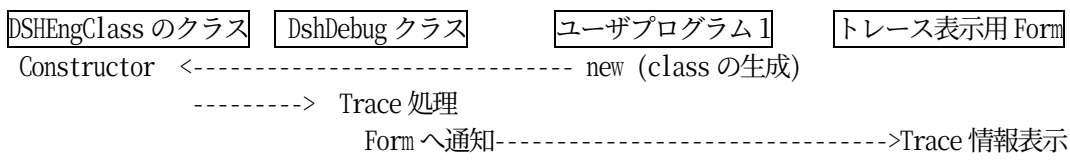
なし。

【説明】

先の 21.1.3.2 で設定したハンドルの Form に通知する Windows Message ID を設定します。

【補足】

たとえば、Construct 時のトレースの流れは次のようになります。



21．1．3．4　enable_trace_log()

DshDebug クラスに対して、トレース表示通知の有効／無効（通知を行うかどうか）を指定します。

【構文】

public static void enable_trace_log (bool flag)

【引数】

flag

有効にする場合は true、 無効にする場合は false を指定します。

【戻り値】

なし。

【説明】

トレース事象が発生したときに、トレース情報をユーザが指定したフォームに通知するかどうかを指定します。

true（有効）を指定すると通知することになり、false(無効)を指定すると通知しないことになります。

21．1．3．5　get_class_name()

クラス・インデクスで指定したクラスの名前を取得します。

【構文】

public static string get_class_name(int x)

【引数】

x

クラスを特定するクラス・インデクスです。（0～）
（DshDebug クラスの定数としてクラス名に対応するクラスインデクスの名前が定義されています。）

【戻り値】

クラス名が返却されます。存在しないいクラスの場合は “????” が返却されます。

【説明】

クラス・インデクスを指定してクラス名を取得します。

21．1．3．6　set_trace_flag()

クラス単位で、トレース表示の有効／無効を設定します。

【構文】

public static bool set_trace_flag(int x, bool flag)

【引数】

x

クラスを特定するクラス・インデクスです。（0～）
（DshDebug クラスの定数としてクラス名に対応するクラスインデクスの名前が定義されています。）

flag

有効／無効を true／false で設定します。

【戻り値】

正常に設定できた場合は true，インデクスのクラスが存在しなかった場合は false を返却します。

【説明】

クラス単位でトレース表示の有効／無効（通知するかどうか）を設定します。

21．1．3．7　get_trace_flag()

クラス単位で、トレース表示の有効／無効情報を取得します。

【構文】

public static bool get_trace_flag(int x,)

【引数】

x

クラスを特定するクラス・インデクスです。（0～）

【戻り値】

有効であった場合は true，無効であった場合は false を返却します。

【説明】

クラス単位でトレース表示の有効／無効（通知するかどうか）情報を取得します。

21．1．3．8　reset_trace_count()

DshDebug クラスとして、ユーザに対し、トレース表示通知の有効／無効（通知を行うかどうか）を指定します。

【構文】

public static void reset_trace_count(int x)

【引数】

x

クラスを特定するクラス・インデクスです。（0～）

【戻り値】

なし。

【説明】

指定されたクラスのトレースカウンタを =0 にします。

21．1．3．9　get_trace_count()

クラス・インデクスで指定したクラスのトレース事象カウンター値を取得します。

【構文】

public static int get_trace_count(int x, ref int nc, ref int dc, ref int fc)

【引数】

x

クラスを特定するクラス・インデクスです。（0～）

nc

new で construct されたカウント格納用変数です。

d

Disposer が呼ばれたカウント格納用変数です。

fc

Finalizer が呼ばれたカウント格納用変数です。

【戻り値】

nc に設定された値を返却します。

【説明】

現時点までのトレース事象が発生した回数をそれぞれ指定された変数に格納返却します。