DSHEng5 装置／ホスト通信エンジンライブラリ(GEM+GEM300)

ソフトウェア・パッケージ

DSHEng5 GEM 通信エンジン・クラス説明書

Vol－5

通信メッセージ情報処理クラス

２０１９年１２月（改訂-1）

株式会社データマップ

文書番号　DSHENG5-19-30325-01

［取り扱い注意］

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

【改訂履歴】

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

目　次

１．はじめに

DSHEng5 GEM 通信エンジン説明書は、Vol-1 から６までの６つの Volume に分けられています。
本説明書の Vol 番号は**5** です。

本説明書では、SECSーII メッセージの情報を保存するクラスについて機能、コンストラクタ、プロパティ、メソッドなどについて説明します。

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

２．HSMS 通信関連情報クラス

DSHDR2　HSMS 通信ドライバーとのインタフェースのために使用する構造体、使用できるクラスについて説明します。

２．１　DSHMSG - SECS-II メッセージ構造体

HSMS 通信ドライバーと SECS-II 通信メッセージのやり取りをするための構造体です。
この DSHMSG 構造体は、クラスには属していません。

DSHMSG の構造は以下の通りです。

```
public struct DSHMSG
{
    public int stream;              // stream id
    public int function;            // function id
    public int wbit;                // wait bit
    public int length;              // msg length
    public IntPtr buffer;           // msg buffer
    public int error;               // error
    public int next;                // next item
    public int txtp;                // (内部処理用)
    public int txtc;                // (   〃     )
    public int work1;               // (   〃     )
    public int work2;               // (   〃     )
}
```

APP が参照できるメンバーの用途は以下の通りです。

	メンバー	用途
1	stream	stream ID
2	function	function ID
3	wbit	wait bit
4	length	message 長(byte)
5	buffer	message 格納メモリ
6	next	D_GetItem() で取得する次のﾃﾞｰﾀｱｲﾃﾑの format　(HSMS.ICODE_A など)

２．２　HSMS クラス － HSMS

HSMS 通信処理関連で使用できるクラスです。

ここでは、HSMS クラスの定数と APP が利用できる HSMS 処理関連クラスについて説明します。

なお、HSMS ドライバーの API 関数についての詳しい内容については**DSHDR2 HSMS 通信ドライバーの説明書**を参照してください。

２．２．１　アイテムコード定数

APP は、HSMS 通信メッセージの中で使用されるアイテムコード（format）の定数を、本 HSMS クラスで参照することができます。
例えば、ASCII の場合、**HSMS.ICODE_A** のように表現します。

アイテムコードの参照名と値は下表の通りです。

アイテムコード一覧表

	参照名	値(16 進)	意味
1	ICODE_L	0x00	List
2	ICODE_A	0x10	ASCII
3	ICODE_J	0x11	JIS
4	ICODE_B	0x08	Binary
5	ICODE_I1	0x19	8 bit Signed Integer
6	ICODE_I2	0x1A	16 “　“
7	ICODE_I4	0x1C	32 “　“
8	ICODE_I8	0x18	64 “　“
9	ICODE_U1	0x29	8 bit Unsigned Integer
10	ICODE_U2	0x2A	16 “　“
11	ICODE_U4	0x2C	32 “　“
12	ICODE_U8	0x28	64 “　“
13	ICODE_F4	0x24	float (32 bit)
14	ICODE_F8	0x20	double(64 bit)
15	ICODE_BOOLEAN	0x09	真理値
16	ICODE_END	0x3D	終端

２．２．２　HSMS　通信エラーコード定数

APP が送信あるいは受信メッセージの Encode/Decode を直接行う場合、HSMS 通信ドライバーの API 関数を使用することになります。

HSMS クラスが返却するエラーコードを下表に示します。

DSHDR2 HSMS　Driver API 関数の返却値一覧表

	名前	値	意味
1	EI_NORMAL	0	Normal End
2	EI_POLL_TRUE	1	Receive Data Available
3	EI_SEND_OK	2	Receive Data Available
4	EI_SEND_W2_OK	3	Send Normal End(wbit=2)
5	EI_NMGR_MSG	4	Not Effective Message
6	EI_ERROR	(-1)	Error End
7	EI_DVR_NOT_OPENED	(-2)	Driver Not Opened
8	EI_DVR_ALREADY_OPENED	(-3)	Driver Already Opened
9	EI_DEVICE_NOT_OPENED	(-4)	Device Not Opened
10	EI_DEVICE_ALREADY_OPENED	(-5)	Device Already Opened
11	EI_DEVICE_UNDEFINED	(-6)	Device Undefined
12	EI_PORT_NOT_OPENED	(-7)	Port Not Opened
13	EI_PORT_ALREADY_OPENED	(-8)	Port Already Opened
14	EI_PORT_UNDEFINED	(-9)	Port Undefined
15	EI_POLL_FALSE	(-10)	No Event Available
16	EI_SEND_FAIL	(-11)	Send Fail
17	EI_TRID_BUSY	(-12)	Transaction Busy
18	EI_TRID_NOT_FOUND	(-13)	TransactionID Not Found
19	EI_TR_TIMEOUT	(-14)	Transaction Timeout
20	EI_ABORT	(-15)	Transaction aborted
21	EI_DATA_ITEM_ERR	(-16)	Data Item Error
22	EI_TYPE_ERR	(-17)	Data type Error
23	EI_ITEM_POS_ERR	(-18)	position Error
24	EI_ARRAY_SIZE_ERR	(-19)	Array Size Error
25	EI_MSG_FORMAT_ERR	(-20)	Message Format Error
26	EI_MSG_SIZE_ERR	(-21)	SECS Msg Length Error
27	EI_SEND_W2_FAIL	(-22)	Send Error(wbit=2)
28	EI_NOT_READY	1	not selected (Selection 確認関数応答)

２．２．３　HSMS 関連処理用メソッド

DSHDR2 通信ドライバーが標準的に提供するメソッドについては、DSHDR2 HSMS 通信ドライバーの説明書を参照して下さい。なお、基本的なメソッドの構文などについては、**付録－A**　を参照ください。

DSHEng5 が通信ドライバー以外で APP が使用できるメソッドは下表の通りです。

	名前	説明
1	public static int D_GetITEM_L()	DSHMSG からアイテムコード L の値を取得する。
2	public static int D_GetITEM_LX()	DSHMSG からアイテムコード L の値を取得し、その値が指定した値と同じかを調べます。
3	public static int D_PutITEM_L()	DSHMSG に指定した値をアイテムコード L として設定します。
4	public static int D_GetITEM_A()	DSHMSG から文字列データを取得します。
5	public static int D_PutITEM_A()	DSHMSG に文字列データを設定します。
6	public static int get_fmt_data_size()	DSHMSG から次のアイテムデータのバイト長を取得します。
7	public static string get_ic_name()	アイテムコードの名前を取得します。

これらメソッドは全て static のメソッドです。

２．２．３．１　D_GetITEM_L() - List サイズの取得

DSHMSG 構造体に保存されているアイテムコード-L が期待されるメッセージのアイテムデータ取得位置からList サイズを取得します。

【構文】

public static int D_GetITEM_L(ref DSHMSG smsg)

【引数】

smsg

SECS-II メッセージ情報が保存されている構造体

【戻り値】

返却値	意　味
>0	取得した L-サイズ
(-1)	アイテムコードが L でなかった。

【説明】

smsg のデータ取得位置からアイテムコード L のリストサイズを取得します。
正常に取得した後は、DSHMSG smsg のデータ取得位置を次のデータアイテムの先頭位置に進めます。

２．２．３．２　D_GetITEM_LX() - サイズを指定して List サイズを取得

DSHMSG 構造体に保存されているアイテムコード-L が期待されるメッセージのアイテムデータ取得位置からList サイズを取得し、その値が引数に指定したサイズと同じかどうかを調べます。

【構文】

public static int D_GetITEM_LX(ref DSHMSG smsg, int n)

【引数】

smsg

SECS-II メッセージ情報が保存されている構造体

n

取得したいサイズ(Link データアイテム数)

【戻り値】

返却値	意　味
0	取得できた。(L,n であることが確認でき、次に進む)
(-1)	アイテムコードが L で、サイズが　n でなかった。

【説明】

smsg のデータ取得位置からアイテムコード L のリストサイズを取得し、その値が引数 n　の値と同じかどうかを調べます。値が一致すれば　0 を返却します。

正常に取得した後は、DSHMSG smsg のデータ取得位置を次のデータアイテムの先頭位置に進めます。

２．２．３．３　D_PutITEM_L() - List サイズの設定

指定サイズのデータアイテム L を DSHMSG のメッセージに追加設定します。

【構文】

```
public static int D_PutITEM_L(ref DSHMSG smsg, int n)
```

【引数】

smsg

　　SECS-II メッセージ情報が保存されている構造体

n

　　設定するリストサイズ

【戻り値】

返却値	意　味
>=0	設定できた。
(-1)	設定できなかった。(メッセージ buff サイズを超えた)

【説明】

smsg のデータ設定位置に n サイズの L アイテムコードを追加設定します。
正常に設定した後は、DSHMSG smsg のデータ設定位置を次のデータアイテムの位置に進めます。

２．２．３．４　D_GetITEM_A() - 文字列の取得

DSHMSG のメッセージの取得位置からアイテムコード A の文字列データを取得します。

【構文】

```
public static int D_GetITEM_A(ref DSHMSG smsg, ref string str)
```

【引数】

smsg

　　SECS-II メッセージ情報が保存されている構造体

str

　　取得した文字列を保存する領域

【戻り値】

返却値	意　味
>=0	取得できた。文字列のバイト長を返却します。
(-1)	取得できなかった。(アイテムコードが A でなかったなど。)

【説明】

smsg のデータ取得位置から、アイテムコード A のデータを取得し、それを str に格納し、取得した文字列のバイト長を返却します。
正常に設定した後は、DSHMSG smsg のデータ取得位置を次のデータアイテムの先頭位置に進めます。

２．２．３．５　D_PutITEM_A() - 文字列の設定

DSHMSG のメッセージの設定位置にアイテムコード A の文字列データを追加設定します。

【構文】

```
public static int D_PutITEM_A( ref DSHMSG smsg, string data)
```

【引数】

smsg

　　SECS-II メッセージ情報が保存されている構造体

data

　　設定したい文字列データ

【戻り値】

返却値	意　味
>=0	取得できた。文字列のバイト長を返却します。
(-1)	取得できなかった。(アイテムコードが A でなかったなど。)

【説明】

smsg のデータ設定位置に、引数 data をアイテムコード A のデータとして追加設定します。
正常に設定した後は、DSHMSG smsg のデータ設定位置を次のデータアイテムの位置に進めます。

２．２．３．６　get_fmt_data_size() - 次のデータ情報の取得

DSHMSG のメッセージの取得位置のデータのアイテムコード(format)とデータサイズを取得します。

【構文】

```
public static int get_fmt_data_size(ref DSHMSG smsg, ref int fmt, ref int size)
```

【引数】

smsg

　　SECS-II メッセージ情報が保存されている構造体

fmt

　　アイテムコード (format) の保存場所

size

　　アイテムデータのサイズ保存場所

【戻り値】

返却値	意　味
>=0	データサイズ (バイト単位)
(-1)	アイテムデータが無かった。

【説明】

smsg のメッセージの次に取得するデータのアイテムコード(format)とそのデータのバイト長を取得します。
データが存在していた場合、データサイズを返却します。

２．２．３．７　get_ic_name() - アイテムコードの記号取得

指定されたアイテムコード(format)の記号を取得します。

【構文】

public static string get_ic_name(int icode)

【引数】

icode

Item Code

【戻り値】

返却値	意　味
記号	記号(string)
“???”	icode の値のアイテムコードが無かった。

【説明】

icode に指定されたアイテムコードの記号を取得します。

アイテムコード記号一覧表

	参照名	ｱｲﾃﾑｺｰﾄﾞ	記号
1	ICODE_L	0x00	“L”
2	ICODE_A	0x10	“A”
3	ICODE_J	0x11	“J”
4	ICODE_B	0x08	“B”
5	ICODE_I1	0x19	“I1”
6	ICODE_I2	0x1A	“I2”
7	ICODE_I4	0x1C	“I4”
8	ICODE_I8	0x18	“I8”
9	ICODE_U1	0x29	“U1”
10	ICODE_U2	0x2A	“U2”
11	ICODE_U4	0x2C	“U4”
12	ICODE_U8	0x28	“U8”
13	ICODE_F4	0x24	“F4”
14	ICODE_F8	0x20	“F8”
15	ICODE_BOOLEAN	0x09	“BOOL”

２．３　HSMS_LIB クラス － HSMS 通信関連ライブラリ

HSMS 通信関連ライブラリクラスについて説明します。

２．３．１　コンストラクタ

なし。

２．３．２　プロパティ

なし。

２．３．３　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 static int get_data_fmt_size()	アイテムコードのバイトサイズを取得します。
2	public static void setup_DSHMSG ()	DSHMSG に stream, function, wait bit,
3	public static void free_DSHMSG()	DSHMSG 構造体の buffer メンバーのメモリを解放します。
4	public static int copy_DSHMSG()	DSHMSG 構造体の内容を別の DSHMSG 構造体にコピーします。
5	public static int set_data_format()	変数のフォーマットを設定します。 （VID, RPID, CEID など）
6	public static int get_data_format()	変数のフォーマットを取得します。
7	public static uint get_dataid()	アイテムデータ dataid を取得します。 （S2F41, S6F11 などのメッセージで使用）

２．３．３．１　get_data_fmt_size() - アイテムコードデータのバイト長取得

指定アイテムコードのデータ１個のバイト長を取得します。

【構文】

```
public static int get_data_fmt_size(int fmt)
```

【引数】

fmt

アイテムコード（format）

【戻り値】

返却値	意　味
>=0	取得したサイズ
(-1)	アイテムコードが L でなかった。

【説明】

アイテムコードに割り当てられるデータの単位バイト長を取得します。

アイテムコード一覧表

	参照名	値(16 進)	単位ﾃﾞｰﾀ長
1	ICODE_L	0x00	0
2	ICODE_A	0x10	1
3	ICODE_J	0x11	1
4	ICODE_B	0x08	1
5	ICODE_I1	0x19	1
6	ICODE_I2	0x1A	2
7	ICODE_I4	0x1C	4
8	ICODE_I8	0x18	8
9	ICODE_U1	0x29	1
10	ICODE_U2	0x2A	2
11	ICODE_U4	0x2C	4
12	ICODE_U8	0x28	8
13	ICODE_F4	0x24	4
14	ICODE_F8	0x20	8
15	ICODE_BOOLEAN	0x09	1

２．３．３．２　setup_DSHMSG() - DSHMSG 構造体の初期設定

DSHMSG 構造体のメンバーに初期データ設定を行います。

【構文】

```
public static void setup_DSHMSG(ref DSHMSG smsg, int s, int f, int w, int buff_size)
```

【引数】

smsg
　　DSHMSG 構造体の実体
s
　　stream
f
　　function
w
　　W bit
buff_size
　　メッセージ Text 部の最大バイト長

【戻り値】

なし。

【説明】

smsg のメンバーに引数で指定された値を設定します。

メンバー	引数	備考
stream	s	
function	f	
wbit	w	
buffer	-	length バイトのメモリを取得し、設定します。
length	buff_size	

buffer に割り当てるメモリは非管理メモリです。

２．３．３．３　free_DSHMSG() - DSHMSG バッファメモリの開放

DSHMSG 構造体のメンバーの buffer に使用していたメモリを解放します。

【構文】

```
public static void free_DSHMSG(ref DSHMSG smsg)
```

【引数】

smsg

　　DSHMSG 構造体の実体

【戻り値】

なし。

【説明】

DSHMSG 構造体のメンバーの buffer に使用していたメモリを解放します。
解放後、buffer の値を IntPtr.Zero にします。

２．３．３．４　copy_DSHMSG () - DSHMSG のコピー

DSHMSG 構造体の内容を別の DSHMSG 構造体にコピーします。

【構文】

```
public static int copy_DSHMSG(ref DSHMSG dst, DSHMSG src)
```

【引数】

dst

　　コピー先 DSHMSG 構造体

src

　　コピー元の DSHMSG 構造体

【戻り値】

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

【説明】

src の DSHMSG 構造体の内容を dst の構造体へコピーします。
buffer メンバーについては、別のメモリを確保してからコピーします。

２．３．３．５　set_data_format()- 変数のフォーマットの設定変更

変数のフォーマット（アイテムコード）を設定変更します。

【構文】

```
public static int set_data_format(int index, int format)
```

【引数】

index

　　変数を指定するインデクス

format

　　設定したいフォーマット

【戻り値】

返却値	意　味
0	設定できた。
(-1)	index が正しくなかった。

【説明】

変数のフォーマットを変更設定します。変数インデクス名と初期設定フォーマットは下表の通りです。
例えば、ALID を ICODE_U2 に変更したい場合は、次のように設定します。

```
HSMS_LIB.set_data_format( HSMS_LIB.X_FMT_ALID, HSMS.ICODE_U2);
```

変数インデクス一覧表

index 値	index 名	初期設定 format	注釈
0	X_FMT_ALID	ICODE_U4	
1	X_FMT_VID	ICODE_U4	
2	X_FMT_ECID	ICODE_U4	
3	X_FMT_DVID	ICODE_U4	
4	X_FMT_SVID	ICODE_U4	
5	X_FMT_LIMITID	ICODE_B	
6	X_FMT_TRID	ICODE_A	Trace ID
7	X_FMT_SMPLN	ICODE_U2	Trace Sampling
8	X_FMT_RPID	ICODE_U4	
9	X_FMT_CEID	ICODE_U4	
10	X_FMT_DATAID	ICODE_U4	
11	X_FMT_PPID	ICODE_A	
12	X_FMT_PPBODY	ICODE_A	
13	X_FMT_LENGTH	ICODE_U4	
14	X_FMT_PTN	ICODE_U1	port id
15	X_FMT_DATA_LENGTH	ICODE_U4	
16	X_FMT_OPID	ICODE_U4	
17	X_FMT_ERRCODE	ICODE_U1	
18	X_FMT_CCODE	ICODE_A	FPP command code
19	X_FMT_RPGSZ	ICODE_U4	Trace group record size

２．３．３．６　get_data_format() - 変数のフォーマットの取得

変数のフォーマット（アイテムコード）を取得します。

【構文】

public static int get_data_format(int index)

【引数】

index

変数を指定するインデクス

【戻り値】

返却値	意　味
format	取得できたフォーマット(Item Code)
(-1)	index が正しくなかった。

【説明】

変数のフォーマットを取得します。

変数インデクス名と初期設定フォーマットは、前節、２．２．４．５の一覧表を参照してください。

２．３．３．７　get_dataid() - メッセージの DATAID の取得

S6F11 などの１番目のアイテムとして使用される DATAID の値を取得します。

【構文】

public static uint get_dataid(int stream)

【引数】

stream

SECS メッセージのヘッダの stream の値です。

【戻り値】

返却値	意　味
DATAID 値	取得できた

【説明】

DATAID はメッセージの stream 別に発行されます。

発行後、当該 stream に対する DATAID 値　は　+1 されます。

３．通信メッセージ関連情報クラス

本章では、VOL-1,2 では説明されなかった、SECS-II メッセージのエンコード／デコードのために APP が使用する可能性のあり、通信のためだけに使用される通信情報クラスについて説明します。

対象になるメッセージは、以下のメッセージになります。

（1）1 次メッセージで、メッセージでキャリア、ポートに対する要求情報を保存するためのクラス

（2）一般的に、1 次メッセージに対する応答メッセージの情報を保存するためのクラス。

３．１　TDATA_ITEM - データアイテム クラス

TDATA_ITEM クラスは、フォーマット（アイテムコード）を別にアイテムデータを保存するために使用します。
本クラスは、S2F41、S2F49 のコマンドパラメータデータ(TOBJ_PARA)に付属します。

また、本クラスには、コマンドパラメータデータを保存する３つの形態があります。CASE_1, 2, 3 です。
これは、TDATA_ITEM の　link_case プロパティに記録されます。

これは、S2F49 のメッセージ構造の中で、コマンドパラメータの名前とパラメータ値が３つの違う形態で設定されるため、１つの手段としてこのような３つのケースにプロパティに分けて処理を行います。

以下のケースの説明では、<CPNAME>、<CPVAL> の並びが TOBJ_PARA クラスの基本計、 <CPVAL>が TDATA_ITEM クラスを意味します。

（1）CASE_1 ： リンクデータがない。(S2F41,　S2F49)

<CPNAME>
<CPVAL>
} CPNAME に 1 個の CPVAL(TDATA_ITEM)が付く。

（2）CASE_2 ： TDATA_ITEM クラスのリンクリストがある。(S2F49)

<CP_NAME>
<Lm>
<$CPVAL_1$>　　// L に CPVAL が配列になる。
<$CPVAL_2$>
・

（3）CASE_3 ： TOBJ_PARA クラスのリンクリストがある。(S2F49)

<CP_NAME>
<Ln>
<L,2>　　// L,2 は TOBJ_PARA クラス
<CPNAME>
<$CPVAL_1$>
} TOBJ_PARA
<L,2>

３．１．１　コンストラクタ

TDATA_ITEM クラスのインスタンスを生成します。

３．１．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public const int CASE_1 = 0	ケース-1 のプロパティに情報を保存します。
2	public const int CASE_2 = 1	ケース-2 のプロパティに情報を保存します。
3	public const int CASE_3 = 2	ケース-3 のプロパティに情報を保存します。
4	public int link_case	当該インスタンスのケースの識別です。(初期値 = CASE_1) CASE_1, CASE_2 または CASE_3 の値になります。
5	public int format	CASE_1 の value の format です。
6	public int size	“ の value のデータサイズです。
7	public IntPtr value	“ の value です。
8	public int link_size	CASE_2 の link サイズ(Lm)です。
9	public int link_pos	“ の次のデータの設定位置です。
10	public int link_count	“ の link_list の設定された要素数です。
11	public TDATA_ITEM[] link_list	“ のパラメータ値保存リストです。
12	public int linkx_size	CASE_3 の link サイズ(Ln)です。
13	public int linkx_count	“ の設定した要素数です。
14	public TOBJ_PARA[] linkx_list	“ のコマンドパラメータ保存リストです。

３．１．３　メソッド

TDATA_ITEM クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_case_1() public void set_case_2() public void set_case_2()	CASE_1 の初期化をします。 CASE_2　　“ CASE_3　　“
4	public void set_data()	CASE_1 ： パラメータ値を設定します。
5	public int add_link_data()	CASE_2 ： パラメータ値を link_list に追加します。
6	public int add_link_para()	CASE_3 ： パラメータを linkx_list に追加します・。 (TOBJ_PARA のインスタンス)
7	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．１．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．１．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．１．３．３　set_case_1,2,3-CASE_1,2,3の設定

当該インスタンスで保存するパラメータの種類識別コードを指定します。

本メソッドは、パラメータの設定メソッドを使用する前に実行してください。

【構文】

```
public void set_case_1()
public void set_case_2(int size)
public void set_case_3(int size)
```

【引数】

size

パラメータのリストの要素数を指定します。
CASE_2　の場合は、link_list のサイズです。
CASE_3　の場合は、linkx_list　　“

【戻り値】

なし。

【説明】

プロパテイの link_case に S2F41, S2F49 に含まれるパラメータのリスト構造の種類を識別するためのメソッドです。

CASE_1 ：　リンクはなく、データ値だけが１個設定されます。
CASE_2 ：　パラメータ値のリストがリンクされます。引数 size がリンクされるデータ数であり、プロパティ size に設定されます。
CASE_3 ：　パラメータ（名前+値）のリストがリンクされます。引数 size がリンクされるパラメータ数であり、プロパティ size に設定されます。

３．１．３．４　set_data()- パラメータ値の設定

CASE_1 ケースで使用され、パラメータ値を１個設定します。

【構文】

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

【引数】

format
　　パラメータ値のフォーマットです。
size
　　パラメータ値のデータサイズです。
value
　　パラメータ値が保存されているメモリまたは文字列(string)です。

戻り値】

返却値	意　味
= 0	設定できた。
(-1)	設定できなかった。 CASE_1 でなかった、または format が L であった。

【説明】

引数 value が string の場合は、format= A のデータにした上で処理します。

３．１．３．５　add_link_data()- コマンドパラメータ値の追加

CASE_2 のケースで使用され、パラメータデータリストにデータを追加します。

【構文】

public int add_link_data(int format, int size, IntPtr value)
public int add_link_data(string value)

【引数】

format
　　パラメータ値のフォーマットです。
size
　　パラメータ値のデータサイズです。
value
　　パラメータ値が保存されているメモリまたは文字列(string)です。

【戻り値】

返却値	意　味
= 0	追加できた。
(-1)	追加できなかった。 CASE_2 でなかった、または format が L であった。

【説明】

コマンドパラメータ値をプロパティ link_list に追加し、list_count+1 します。
引数 value が string の場合は、format= A のデータにした上で処理します

３．１．３．６　add_link_para()- コマンドパラメータの追加

CASE_3 のケースで使用され、パラメータ(TOBJ_PARA)を linkx_list に追加します。

【構文】

public int add_lin_para(TOBJ_PARA para)

【引数】

para

　　パラメータです。

【戻り値】

返却値	意　味
= 0	追加できた。
(-1)	追加できなかった。 para=null、linx_size =0 または linkx_count >= linkx_size の場合

【説明】

コマンドパラメータ名とその値をプロパティ linkx_list に設定し、linkx_count+1 します。

３．１．３．７　copy()- インスタンスのコピー

TDATA_ITEM のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．２　TOBJ_PARA - データアイテム クラス

TOBJ_PARA クラスは、S2F41、S2F49 のリモートコマンドのパラメータを保存のためのクラスです。

３．２．１　コンストラクタ

TOBJ_PARA クラスのインスタンスを生成します。

	メソッド名	説明
1	public TOBJ_PARA()	インスタンスを生成します。
2	public TOBJ_PARA(string name, int format, int size, IntPtr value)	インスタンス生成後、引数で指定されたパラメータをプロパティに設定します。
3	public TOBJ_PARA(string name, string value)	インスタンス生成後、A-フォーマットのパラメータを設定します。

３．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string name	パラメータ名です。
2	public TDATA_ITEM val_data	パラメータ値保存用の TDATA_ITEM クラスのインスタンスです。 3.1 TDATA_ITEM クラス参照

３．２．３　メソッド

TOBJ_PARA クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_para()	パラメータを設定します。
4	public void set_case_2(int size)	CASE_2 の設定を指定します。 add_link_data()で設定すること。
5	public int add_link_data()	パラメータデータ値をリンクリストに追加します。
6	public void set_case_3(int size)	CASE_3 の設定を指定します。 add_link_para()で設定すること。
7	public int add_link_para()	val_data のプロパティ、TPARA_OBJ クラスのリストにパラメータを追加します。
8	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．２．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．２．３．３　set_para()- パラメータ値の設定

CASE_1 のケースでパラメータを設定します。パラメータは名前と値から成ります。

【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
　　パラメータ値のフォーマットです。
size
　　パラメータ値のデータサイズです。
value
　　パラメータ値が保存されているメモリまたは文字列(string)です。

【戻り値】

なし。

【説明】

引数 value が string の場合は、format= A のデータにした上で処理します。
引数 name をプロパテイ name に設定し、引数 format, size, value をプロパティ val_data に設定します。

３．２．３．４　set_case_2 - CASE_２の設定

当該インスタンスで保存するパラメータの種類識別コードを CASE_2 に指定します。
本メソッドは、パラメータデータの追加メソッドを使用する前に実行してください。

【構文】

public void set_case_2(int size)

【引数】

size
　　link_list のサイズです。

【戻り値】

なし。

【説明】

プロパテイの link_case に S2F41, S2F49 に含まれるパラメータのリスト構造の種類を識別するメソッドです。
CASE_2 は、パラメータ値のリストがリンクされます。引数 size がリンクされるデータ数であり、プロパティ link_size に設定されます。

３．２．３．５　add_link_data()-パラメータデータの追加

CASE_2 のケースで、パラメータデータをプロパティ val_data の link_list に追加します。

【構文】

public int add_link_data(int format, int size, IntPtr pval)
public int add_link_data(string pval)

【引数】

format
　　パラメータ値のフォーマットです。
size
　　パラメータ値のデータサイズです。
pval
　　パラメータ値が保存されているメモリまたは文字列(string)です。

【戻り値】

返却値	意　味
= 0	追加できた。
(-1)	val_data に追加できなかった。 ①link_case が CASE_2 でなかった。②format= L であった。 ③pval が IntPtr.Zero であった。④link_count >= link_size であった。

【説明】

引数 pval が string の場合は、format= A のデータにした上で処理します。
引数に与えられたパラメータデータをプロパティ val_data の link_list に追加し、link_count+1 します。

３．２．３．６　set_case_3 - CASE_3 の設定

当該インスタンスで保存するパラメータの種類識別コードを CASE_3 に指定します。
本メソッドは、パラメータの追加メソッドを使用する前に実行してください。

【構文】

public void set_case_3(int size)

【引数】

size
　　linkx_list のサイズです。

【戻り値】

なし。

【説明】

プロパテイの link_case に S2F41, S2F49 に含まれるパラメータのリスト構造の種類を識別するメソッドです。
CASE_3 は、パラメータのリストがリンクされます。引数 size がリンクされるパラメータ数であり、プロパティ size に設定されます。

３．２．３．７　add_link_para()-パラメータの追加

CASE_3 のケースで、パラメータをプロパティ val_data の linkx_list に追加します。

【構文】

public int add_link_para(TOBJ_PARA　para)

【引数】

para

　　パラメータです。

【戻り値】

返却値	意　味
= 0	追加できた。
(-1)	追加できなかった。 para=null or linx_size =0

【説明】

引数 para をプロパティ val_data の linkx_list に追加し、linkx_count+1 します。

３．２．３．８　copy()- インスタンスのコピー

TOBJ_PARA のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．３　TERR_INFO - オブジェクトエラー クラス

TERR_INFO クラスは、オブジェクト関連応答メッセージ情報を保存するためのクラスです。
(S14F10, S15F14, S1612, S2F42, S2F50 など)

３．３．１　コンストラクタ

TERR_INFO クラスのインスタンスを生成します。

３．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int errcode	エラーコードです。
2	public string errtext	エラーテキストです。

３．３．３　メソッド

TERR_INFO クラスのメソッドは下表のとおりです。

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

３．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイをすべてクリアします。

３．３．３．３　set() - エラー値の設定

エラー情報を設定します。

【構文】

public void set(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】

引数 code をプロパティ errcode に、text を errtext に設定します。

３．３．３．４　copy() - インスタンスのコピー

TERR_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．４　TERR_INFO_LIST - オブジェクトエラー クラス

TERR_INFO_LIST クラスは、オブジェクト関連応答メッセージ TERR_INFO 情報のリストを保存するためのクラスです。

３．４．１　コンストラクタ

TERR_INFO_LIST クラスのインスタンスを生成します。

３．４．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int err_count	エラー情報です。
2	public TERR_INFO[] err_list	エラー情報リストです。

３．４．３　メソッド

TERR_INFO_LIST クラスのメソッドは下表のとおりです。

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

３．４．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．４．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．４．３．３　add_TERR_INFO_LIST() - エラー値の設定

エラー情報を err_list に追加します。

【構文】

public void add_TERR_INFO_LIST(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】

err_list[err_count]に引数 code、text を設定追加します。そして、err_count+1 にします。

３．４．３．４　copy() - インスタンスのコピー

TERR_INFO_LIST のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．５　TRCMD_INFO - ホストコマンド情報クラス

TRCMD_INFO クラスは、ホストコマンド（リモートコマンド）情報を保存するために使用します。

３．５．１　コンストラクタ

TRCMD_INFO クラスのインスタンスを生成します。

３．５．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string cmd	ホストコマンド名です。
2	public int count	ホストコマンド パラメータの数です。
3	public TOBJ_PARA[] list	ホストコマンド パラメータの保存リストです。

３．５．３　メソッド

ホストコマンド TRCMD_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_cmd()	ホストコマンドを設定します。
4	public void add_para()	ホストコマンドのパラメータを list に追加します。
5	public int add_link_data(string val)	list の当該要素位置のパラメータにリンクデータを追加します。
6	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．５．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．５．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．５．３．３　set_cmd()- ホストコマンドの設定

ホストコマンド ID と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

【戻り値】

なし。

【説明】

ホストコマンドを cmd プロパティに設定します。

３．５．３．４　add_para()- コマンドパラメータの追加

ホストコマンドのパラメータを追加します。

【構文】

public void add_para(string pname, string value)

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

【引数】

pname

　　パラメータ名です。

format

　　パラメータ値のフォーマットです。

size

　　パラメータ値のデータサイズです。

val

　　パラメータ値が保存されているメモリです。

【戻り値】

なし。

【説明】

コマンドパラメータ名とその値をプロパティに設定します。

2 番目の引数が value(string)の場合には、フォーマット=A、size=para 文字列のバイト長で非管理メモリを確保してプロパティに設定します。

３．５．３．５　add_link_data()- コマンドパラメータのリンクデータ追加

ホストコマンドのパラメータを追加します。

【構文】

```
public int add_link_data(string val)
public int add_link_data(int format, int size, IntPtr pval)
```

【引数】

val
　　文字列のリンクデータの値です。
format
　　リンクデータのフォーマットです。
size
　　リンクデータのデータサイズです。
pval
　　リンクデータ値が保存されているメモリです。

【戻り値】

なし。

【説明】

前回追加したコマンドパラメータのリンクデータリストにリンクデータを１個追加します。
１番目の引数が val(string)の場合には、フォーマット=A、size=val 文字列のバイト長で非管理メモリを確保してリンクデータとしてリンクデータリストに追加します。

３．５．３．６　copy()- インスタンスのコピー

TRCMD_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

```
public static int copy(ref TRCMD_INFO dst, TRCMD_INFO src)
```

【引数】

dst
　　コピー先のインスタンス
src
　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．６　TERCMD_INFO - 拡張リモートコマンド情報クラス

TERCMD_INFO クラスは、拡張リモートコマンド（リモートコマンド）情報を保存するために使用します。

コマンドパラメータの設定時、S2F49 メッセージのパラメータリスト構造には３つの形態があります。そのため、TOBJ_PARA クラスのプロパテイの１つである TDATA_ITEM val_data への情報の設定を３つのケースに分けて行います。
ケース分けについては、**３．１　TDATA_ITEM クラス**の説明を参照ください。

３．６．１　コンストラクタ

TERCMD_INFO クラスのインスタンスを生成します。
引数 objspec, rcmd を付けた場合、生成時に引数をプロパティに設定します。

```
public TERCMD_INFO( string objspec, string rcmd){
            this.objspec = objspec;
            this.rcmd = rcmd;
        }
```

３．６．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string objspec	Object 仕様名です。
2	public string rcmd	拡張リモートコマンド名です。
3	public int count	拡張リモートコマンド パラメータの数です。
4	public TOBJ_PARA[] list	拡張リモートコマンド パラメータの保存リストです。 TOBJ_PARA クラスのプロパティの中に TDATA_ITEM クラスが含まれています。

３．６．３　メソッド

拡張リモートコマンド TERCMD_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_objspec()	objspec を設定します。
4	public void set_rcmd()	拡張リモートコマンド名を設定します。
5	public int set_para_case_1()	CASE_1 のパラメータ（パラメータ名 + 値）を設定します。
6	public void add_para()	CASE_1 のパラメータを設定します。 （set_para_case_1()と同じ機能です）
7	public void set_para_case_2()	CASE_2 のパラメータを設定します。 (TDATA_ITEM リストの要素数を指定します)
8	public int add_para_data_case_2()	CASE_2 のパラメータ内の link_list リストに 1 個 TDATA_ITEM クラスのインスタンスを追加します。
9	public void set_para_case_3()	CASE_3 のパラメータを設定します。
10	public int add_para_data_case_3()	CASE_3 のパラメータ内の linkx_list リストに 1 個 TOBJ_PATA クラスのインスタンスを追加します。
11	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．６．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．６．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．６．３．３　set_objspec() - Object Spec の設定

拡張リモートコマンド対象オブジェクト仕様名を設定します。

【構文】

public void set_objspec(string objspec)

【引数】

objspec

オブジェクト仕様名です。

【戻り値】

なし。

【説明】

引数のオブジェクト仕様名 objspec を　objspec プロパティに設定します。

３．６．３．４　set_cmd () - 拡張リモートコマンドの設定

拡張リモートコマンド名を設定します。

【構文】

public void set_cmd(string rcmd)

【引数】

cmd

拡張リモートコマンド名

【戻り値】

なし。

【説明】

拡張リモートコマンドを cmd プロパティに設定します。

３．６．３．５　set_para_case_1()・CASE_1 のパラメータの設定

CASE_1 ケースの拡張リモートコマンドのパラメータ情報を１個設定します。

【構文】

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

【引数】

pname
　　パラメータ名です。
format
　　パラメータ値のフォーマットです。
size
　　パラメータ値のデータサイズです。
val
　　パラメータ値が保存されているメモリです。

【戻り値】

なし。

【説明】

コマンドパラメータ名とパラメータデータをプロパティにそれぞれ pname と format, size, val に設定します。2 番目の引数が value(string)の場合には、format=A、size=para 文字列のバイト長で非管理メモリを確保した上でプロパティに設定します。

３．６．３．６　add_para()・コマンドパラメータの追加

３．６．３．５　set_para_case_1() 全く同様の機能です。拡張リモートコマンドのパラメータを追加します。

【構文】

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

【引数】

３．６．３．５　set_para_case_1()の引数と同じです。

【戻り値】

なし。

【説明】

３．６．３．５　set_para_case_1()の説明と同じです。

３．６．３．７　set_para_case_2()－CASE_2 のパラメータの設定

CASE_2 ケースの拡張リモートコマンドのパラメータの初期設定を行います。

CASE_2 は、TERCMD_INFO クラスのパラメータ 1 個に対し、複数のパラメータデータがリンクされるケースです。

【構文】

public void set_para_case_2(string pname, int size)

【引数】

pname

パラメータ名です。

size

パラメータにリンクするパラメータデータの数を指定します。

【戻り値】

なし。

【説明】

プロパティ TOBJ_PARA リストの list[count]に 1 個のインスタンスを生成します。そのインスタンスの中のプロパティ name に pname を設定します。

その後、TOBJ_PARA のインスタンスの中の、プロパティ link_size に、リンクされるデータ数　size を設定します。
これは、その TOBJ_PARA のインスタンス中の link_list 配列リストに size 個のパラメータデータをリンクすることを意味します。(TDATA_ITEM link_list[size])

３．６．３．８　add_para_data_case_2()-　コマンドパラメータのリンクデータ追加

CASE_2 のケースで、プロパティ list の要素であるパラメータ TOBJ_PATRA クラスのプロパティである TDATA_ITEM の val_data インスタンスの下の link_list リストに、引数で指定されたパラメータデータを追加します。
(list[count].val_data.link_list[].format, size, value)

【構文】

public int add_para_data_case_2(string val)
public int add_para_data_case_2(int format, int size, IntPtr pval)

【引数】

val
　　文字列のリンクデータの値です。
format
　　リンクデータのフォーマットです。
size
　　リンクデータのデータサイズです。
pval
　　リンクデータ値が保存されているメモリです。

戻り値】

返却値	意　味
= 0	追加できた。
(-1)	追加できなかった。指定されたサイズを超えていた。

【説明】

CASE_2 ケースのパラメータ list[count]のプロパティ val_data のプロパティ link_list リストにパラメータデータを 1 個追加します。

1 番目の引数が val(string)の場合には、format=A、size=val 文字列のバイト長で非管理メモリを確保してリンクデータとしてリンクデータリストに追加します。

３．６．３．９　set_para_case_3()－CASE_3 のパラメータの設定

CASE_3 ケースの拡張リモートコマンドのパラメータ初期設定を行います。

CASE_3 では、1 個の TERCMD_INFO クラスに付属するパラメータの TDATA_ITEM クラスの val_data の中に複数のパラメータがリンクされるケースです。

【構文】

public void set_para_case_3 月(string pname, int size)

【引数】

pname

パラメータ名です。

size

パラメータのデータにリンクする TOBJ_PARA クラスのインスタンス、linkx_list リストに設定するパラメータの数です。

【戻り値】

なし。

【説明】

プロパティ TOBJ_PARA リスト、list[count]に１個のインスタンスを生成します。まず、そのインスタンスの中のプロパティ name に pname を設定します。

その後、TOBJ_PARA のインスタンスの中の、プロパティ linkx_size に、リンクされるパラメータ数　size を設定します。これは、その TOBJ_PARA のインスタンス中の val_data プロパティのメンバーlinkx_list 配列リストに size 個のパラメータをリンクすることを意味します。

３．６．３．10　add_para_data_case_3()- コマンドパラメータのリンクデータ追加

CASE_3 のケースで、プロパティ list[count]のパラメータ TOBJ_PATRA クラスのインスタンスの linkx_list リストに、引数で指定されたパラメータを追加します。

【構文】

```
public int add_para_data_case_3(string pname, string val)
public int add_para_data_case_3(string pname, int format, int size, IntPtr pval)
```

【引数】

pname

　　パラメータ名です。

val

　　文字列のリンクデータの値です。

format

　　リンクデータのフォーマットです。

size

　　リンクデータのデータサイズです。

pval

　　リンクデータ値が保存されているメモリです。

戻り値】

返却値	意　味
= 0	追加できた。
(-1)	追加できなかった。指定されたサイズを超えていた。

【説明】

CASE_3 ケースのパラメータのリンクパラメータリストにリンクパラメータを 1 個追加します。

1 番目の引数が val(string)の場合には、format=A、size=val 文字列のバイト長で非管理メモリを確保してリンクデータとしてリンクデータリストに追加します。

３．６．３．11　copy()- インスタンスのコピー

TERCMD_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．７　TCACT_CONTENT - キャリアコンテント情報クラス

TCACT_CONTENT クラスは、キャリアコンテント情報を保存するために使用します。

S3F17 メッセージ情報を保存する TCACT_INFO クラスのプロパティの中で使用されます。

３．７．１　コンストラクタ

TCACT_CONTENT クラスのインスタンスを生成します。

３．７．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	キャリアコンテントに含まれる lotid, substid の数です。
2	public string[] lotid_list	LOTID リストです。
3	public string[] substid_list	基板 ID リストです。

３．７．３　メソッド

キャリアコンテント TCACT_CONTENT クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void add_content()	lotid_list, subst_ild リストにそれぞれの ID を設定します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．７．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．７．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．７．３．３　add_content() - ロット、基板 ID の設定

ロット ID と基板 ID をリストに追加します。

【構文】

public void add_content(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】

引数の lotid と substid をそれぞれ、プロパティ lotid_list[count], substid_list[count]に追加します。
１個づつ追加し、プロパティ count + 1 します。

３．７．３．４　copy() - インスタンスのコピー

TCACT_CONTENT のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．８　TCACT_SLOT_INFO - キャリアスロット情報クラス

TCACT_SLOT_INFO クラスは、キャリアスロット情報を保存するために使用します。

S3F17 メッセージ情報を保存する TCACT_INFO クラスのプロパティの中で使用されます。

３．８．１　コンストラクタ

TCACT_SLOT_INFO クラスのインスタンスを生成します。

３．８．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	キャリアに含まれるスロットの数です。
2	public int[] slot_list	スロット ID リストです。

３．８．３　メソッド

キャリアスロット TCACT_SLOT_INFO クラスのメソッドは下表のとおりです。

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

３．８．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．８．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．８．３．３　add_slot() - スロット ID の設定

ロット ID と基板 ID をリストに追加します。

【構文】

public void add_slot(int slotid)

【引数】

slotid

スロット ID です。

【戻り値】

なし。

【説明】

引数の slotid をプロパティ slot_list[count]に追加します。

１個づつ追加し、プロパティ count + 1 します。

３．８．３．４　copy() - インスタンスのコピー

TCACT_SLOT_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

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

【説明】

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

３．９　TCACT_ATTR - キャリアアクション属性情報クラス

TCACT_ATTR クラスは、キャリアアクション属性情報を保存するために使用します。

S3F17 メッセージ情報を保存する TCACT_INFO クラスのプロパティクラス情報として使用されます。

３．９．１　コンストラクタ

TCACT_ATTR クラスのインスタンスを生成します。

	メソッド名	説明
1	public() TCACT_ATTR(){}	インスタンスを生成します。
2	public TCACT_ATTR(int attr_index)	プロパティ attr_index に引数 attr_index を設定し、その attr_index に対する属性 ID をプロパティ attrid に設定します。
3	public TCACT_ATTR(int attr_index, int slot_count)	プロパティ attr_index に引数 attr_index を設定し、その attr_index に対する属性 ID をプロパティ attrid に設定します。 また、引数 slot_count をプロパティ slot_count に設定します。

生成時に、プロパティ capacity の値を DSHEng5 立ち上げ時に、装置起動ファイルに定義された capaciy の設定値を設定します。（ TCONFIG_INFO.kcar_capacity の値です。default 値=25 ）

３．９．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string attrid	属性 ID です。
2	public int attr_index	３．10 class_CACT_TAB クラスを参照ください。 （index 値と属性名との関連の説明があります。）
3	public string objtype	Object Type です。
4	public string objid	ObjectID です。(= Carrier ID)
5	public int capacity	収納容量です。(インスタンス生成時に設定されます) (EngAPI.config.kcar_capacity に値が保存されています)
6	public int car_access_status	キャリアアクセス状態語です。
7	public int id_status	キャリア ID 読み取り状態語です。
8	public string location	ロケーションです。
9	public int map_status	マップ状態です。
10	public int subst_count	基板数です。
11	public int slot_count	スロットカウントです。
12	public string usage	用途です。
13	public TCACT_CONTENT content_map	コンテントのマップです。
14	public TCACT_SLOT_INFO slot_map	スロットマップです。

３．９．３　メソッド

キャリアアクション属性 TCACT_ATTR クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int set_attr_index()	属性インデクスを設定します。 同時に index 値から属性 ID を取得して attrid に設定します。
4	public int set_cact_content()	キャリアが保存している情報を設定します。
5	public int set_slot_map ()	スロット情報を設定します。
6	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．９．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．９．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．９．３．３　set_attr_index()- スロットIDの設定

属性インデクスを設定します。

【構文】

public int set_attr_index(int index)

【引数】

attr_index

属性インデクスです。

【戻り値】

返却値	意　味
= 0	設定できた。
(-1)	index 値が無効であった。

【説明】

属性インデクスから属性 ID を求めます。

有効であれば、引数 index を attr_index に、求めた属性 ID を attrid に設定します。

index から属性を求めるために使用するクラスとメソッドは　class_TCACT_TAB,get_cact_action_name()です。

３．９．３．４　set_cact_content()- キャリアコンテントの設定

キャリアコンテントを設定します。

【構文】

public int set_cact_content(TCACT_CONTENT content)

【引数】

content

キャリアの中身情報を設定します。

【戻り値】

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

【説明】

引数、TCACT_TCONTENT クラスのインスタンスの内容をプロパティ content_map に設定します。

TCACT_CONTENT クラスについては３．７　TCACT_CONTENT クラスの説明を参照下さい。

３．９．３．５　set_slot_map() - スロットマップ設定

スロット ID リスト情報を設定します。

【構文】

public int set_slot_map(TCACT_SLOT_INFO sinfo)

【引数】

sinfo

スロット ID リストです。

【戻り値】

なし。

【説明】

引数の sinfo の内容をプロパティ slot_map に設定します。

３．９．３．６　copy() - インスタンスのコピー

TCACT_ATTR のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

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

【説明】

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

３．10　TCACT_INFO - キャリアアクション情報クラス

TCACT_INFO クラスは、キャリアアクション情報を保存するために使用します。
S3F17 キャリアアクションメッセージのエンコード／デコードで情報保存のために使用されます。

３．10．１　コンストラクタ

TCACT_INFO クラスのインスタンスを生成します。
アクション、キャリア、ポートを引数とする次の２つのコンストラクター―を使用することができます。

（1）public TCACT_INFO(int action_index, string carid, int port)

```
{
        this.action_index = action_index;
        this.carid = carid;
        this.port = port;
        action = class_CACT_TAB.get_cact_action_name(action_index);
}
```

（2）public TCACT_INFO(string action, string carid, int port)

```
{
        this.action = action;
        this.action_index = class_CACT_TAB.get_cact_action_index(action);
        this.carid = carid;
        this.port = port;
}
```

３．10．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string action	キャリアアクション名です。
2	public int action_index	キャリアアクション名に対するインデクス値です。
3	public string carid	キャリア ID です。
4	public int port	ポート ID です
5	public int attr_count	パラメータカウンタ(cp_list の)
6	public TCACT_ATTR[] attr_list	属性情報リストです。

３．10．３　メソッド

キャリアアクション TCACT_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int set_action_index()	アクションインデクスを設定し、それからアクション名を求めて action に設定します。
4	public int set_action()	アクション名を設定し、それからアクションインデクスを求めて action_index に指定します。
5	public void set_carid()	キャリア ID を設定します。
6	public void set_port()	ポート ID を設定します。
7	public int add_attr()	アクション属性情報を設定します。
8	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．10．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．10．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．10．３．３　set_action_index()- アクションインデクスの設定

アクションインデクスを設定します。
有効なインデクスであれば、アクション名を取得し、設定します。

【構文】

public int set_action_index(int index)

【引数】

index

キャリアアクションを指定するインデクスです。

【戻り値】

返却値	意　味
= 0	設定できた。
(-1)	設定できなかった。指定された index が有効でなかった。

【説明】

アクションインデクスからアクション名を求め、それぞれプロパティ action_index, action に設定します。
index から action 名を取り出すために、class_CACT_TAB クラスの get_cact_action_name()メソッドを使用します。index が有効でない場合は、(-1)を返却します。

３．10．３．４　set_action ()-アクションの設定

キャリアアクション名を設定します。
アクション名が有効であれば、アクションインデクスを取得し、設定します。、

【構文】

public int set_action(string action_name)

【引数】

action

キャリアアクション名

【戻り値】

返却値	意　味
= 0	設定できた。
(-1)	設定できなかった。指定された action_name が有効でなかった。

【説明】

アクション名からアクションインデクスを求め、それぞれプロパティ action, action_index に設定します。
action_name から action_index を取り出すため、class_CACT_TAB クラスの get_cact_action_index()メソッドを使用します。action_name が有効でない場合は、(-1)を返却します。

３．10．３．５　set_carid() - キャリア ID の設定

キャリア ID を設定します。

【構文】

```
public void set_carid( string carid)
```

【引数】

carid

キャリア ID です。

【戻り値】

なし。

【説明】

引数 carid をプロパティ carid に設定します。

３．10．３．６　set_port() - ポート ID の設定

ポート ID を設定します。

【構文】

```
public void set_port( int port)
```

【引数】

port

ポート ID です。

【戻り値】

なし。

【説明】

引数 port をプロパティ port に設定します。

３．10．３．７　add_attr()－　属性情報の追加

属性情報を追加します。属性情報は、TCACT_ATTR クラスのインスタンスで与えられます。

【構文】

public int add_attr(TCACT_ATTR　attr_info)

【引数】

attr_info

　　属性情報が保存されています。

戻り値】

返却値	意　味
= 0	追加できた。
(-1)	追加できなかった。attr_info が null であった。

【説明】

引数 attr_info で与えられた属性情報をプロパティ attr_list[attr_count]に設定します。

３．10．３．８　copy()－インスタンスのコピー

TCACT_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．11　TCACT_ERR_INFO - キャリアアクション、ポートアクション応答情報クラス

TCACT_ERR_INFO クラスは、キャリアアクション、ポートアクションのメッセージ S3F17, S3F23, S3F25 の応答情報を保存します。

S3F18, S3F24, S3F26 メッセージのエンコード／デコード処理で使用されます。

３．11．１　コンストラクタ

TCACT_ERR_INFO クラスのインスタンスを生成します。

３．11．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int caack	応答 ack です。
2	public int err_count	エラー情報のカウントです。
3	public TERR_INFO[] err_list	エラー情報保存リストです。

３．11．３　メソッド

キャリアアクション、ポートアクション応答 TCACT_ERR_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void add_err()	err_list にエラー情報を 1 個追加します。

３．11．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．11．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．11．３．３　add_err()－エラー情報の追加

エラー情報をエラーリストに追加します。

【構文】

public void add_err(int err_code, string err_text)

【引数】

err_code

　　エラーコードです。

err_text

　　エラーテキストです。

【戻り値】

なし。

【説明】

err_list に err_code と err_text を追加します。
追加したあと、err_count + 1 します。

３．12　TPORTG_INFO - ポートグループ情報クラス

TPORTG_INFO クラスは、ポートグループ情報を保存するために使用します。
S3F23 ポートグループメッセージのエンコード／デコードで情報保存のために使用されます。

３．12．１　コンストラクタ

TPORTG_INFO クラスのインスタンスを生成します。
アクション、ポートグループを引数とするコンストラクター―を使用することができます。

```
public TPORTG_INFO(string action, string group_name)
  {
      this.action = action;
      name = group_name;
  }
```

３．12．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string action	ポートアクション名です。
2	public string name	ポートグループ名です。
3	public int count	パラメータの数です。
4	public TPORT_PARA[] list	パラメータリストです

３．12．３　メソッド

ポートグループ TPORTG_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void init_set()	アクションとグループ名を設定します。
4	public void add_para()	ラメータを追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．12．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．12．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．12．３．３　init_set()- アクションとグループ名の設定

アクションイン名とポートグループ名を設定します。

【構文】

public void init_set(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 action, group_name をそれぞれ、プロパティ　action, name に設定します。

３．12．３．４　add_para()-　パラメータ情報の追加

パラメータ情報をプロパティ list に追加します。

【構文】

public void add_para(string name, int format, int size, IntPtr val)

public void add_para(string pname, string value)

【引数】

pname

　　パラメータ名です。

format

　　パラメータ値のフォーマットです。

size

　　パラメータ値のデータサイズです。

val

　　パラメータ値が格納されているメモリです。

value

　　パラメータ値が保存されているメモリまたは文字列(string)です。

【戻り値】

なし。

【説明】

引数 pname をプロパティ name に設定します。

そして、引数のパラメータ値をプロパティ list に追加し、count+1 します。

引数 value が string の場合は、format= A のデータにした上で処理します

３．12．３．５　copy()- インスタンスのコピー

TPORTG_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

```
public static int copy(ref TPORTG_INFO dst, TPORTG_INFO src)
```

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．13　TPORT_INFO - ポート情報クラス

TPORT_INFO クラスは、ポート情報を保存するために使用します。
S3F25 ポートメッセージのエンコード／デコードで情報保存のために使用されます。

３．13．１　コンストラクタ

TPORT_INFO クラスのインスタンスを生成します。
アクション、ポートを引数とするコンストラクターを使用することができます。

```
public TPORT_INFO(string action, int port)
  {
      this.action = action;
      this.port = port;
  }
```

３．13．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string action	ポートアクション名です。
2	public int port	ポート ID です。
3	public int count	パラメータの数です。
4	public TPORT_PARA[] list	パラメータリストです

３．13．３　メソッド

ポート TPORT_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void init_set()	アクションとポートを設定します。
4	public void add_para()	アクションパラメータを追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．13．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．13．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．13．３．３　init_set()- アクションとポート ID の設定

アクションイン名とポート ID を設定します。

【構文】

public void init_set(string action, int port)

【引数】

action

　　アクション名です。

port

　　ポート名 ID です。

【戻り値】

なし。

【説明】

引数 action, port をそれぞれ、プロパティ　action, port に設定します。

３．13．３．４　add_para()-　パラメータ情報の追加

パラメータ情報をプロパティ list に追加します。

【構文】

public void add_para(string name, int format, int size, IntPtr val)

public void add_para(string pname, string value)

【引数】

pname

　　パラメータ名です。

format

　　パラメータ値のフォーマットです。

size

　　パラメータ値のデータサイズです。

val

　　パラメータ値が格納されているメモリです。

value

　　パラメータ値が保存されているメモリまたは文字列(string)です。

【戻り値】

なし。

【説明】

引数 pname をプロパティ name に設定します。

そして、引数のパラメータ値をプロパティ list に追加し、count+1 します。

引数 value が string の場合は、format= A のデータにした上で処理します。

３．13．３．５　copy()- インスタンスのコピー

TPORT_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

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

【説明】

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

３．14　TACCESS_INFO - ポートアクセス情報クラス

TACCESS_INFO クラスは、ポートアクセス情報を保存するために使用します。
S3F27 ポートメッセージのエンコード／デコードで情報保存のために使用されます。

３．14．１　コンストラクタ

TACCESS_INFO クラスのインスタンスを生成します。
アクションモードを引数とするコンストラクター―を使用することができます。

```
public TACCESS_INFO(int mode)
{
      this.mode = mode;
}
```

３．14．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int mode	アクセスモードです。
2	public int count	ポート ID の数です。
3	public int[] list	ポート ID リストです。

３．14．３　メソッド

ポート TACCESS_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_mode()	アクセスモードを設定します。
4	public void add_port()	ポート ID リストにポート ID を追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．14．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．14．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．14．３．３　set_mode() - アクセスモードの設定

アクセスモードを設定します。

【構文】

public void set_mode(int mode)

【引数】

mode

アクセスモードです。

【戻り値】

なし。

【説明】

引数 mode をプロパティ mode に設定します。

３．14．３．４　add_port()　- ポート ID の追加

ポート ID をプロパティ list に追加します。

【構文】

public void add_port(int port)

【引数】

port

ポート ID です。

【戻り値】

なし。

【説明】

引数 port を、プロパティ list に追加します。
追加した後、 count + 1 します。

３．14．３．５　copy()- インスタンスのコピー

TACCESS_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．15　TACCESS_ERR_INFO - ポートアクセスエラー情報クラス

TACCESS_ERR_INFO クラスは、ポートアクセスエラー情報を保存するために使用します。

S3F28 ポートメッセージのエンコード／デコードで情報保存のために使用されます。

３．15．１　コンストラクタ

TACCESS_ERR_INFO クラスのインスタンスを生成します。

３．15．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int caack	応答 ACK です。
2	public int err_count	エラーカウントです。
3	public TACCESS_ERR_PORT[] err_list	エラー情報リストです

３．15．３　メソッド

ポート TACCESS_ERR_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_ack()	ACK を設定します。
4	public void add_err()	ポートエラー情報を err_list に追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．15．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．15．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．15．３．３　set_ack() - 応答 ACK の設定

S3F28 の caack を設定します。

【構文】

public void set_ack(int ack)

【引数】

ack

応答 ACK です。

【戻り値】

なし。

【説明】

引数 ack をプロパティ caack に設定します。

３．15．３．４　add_err()　- エラー情報の追加

エラーをプロパティ err_list に追加します。

【構文】

public void add_err(int port, int err_code, string err_text)

【引数】

port

ポート ID です。

err_code

エラーコードです。

err_text

エラーテキストです。

【戻り値】

なし。

【説明】

引数から TACCESS_ERR_PORT クラスのインスタンスを生成し、プロパティ err_ list に追加します。
追加した後、 err_count + 1 します。

３．15．３．５　copy()- インスタンスのコピー

TACCESS_ERR_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

```
public static int copy(ref TACCESS_ERR_INFO dst, TACCESS_ERR_INFO src)
```

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．16　TACCESS_ERR_PORT - ポートアクセスエラー クラス

TACCESS_ERR_PORT クラスは、S3F28 メッセージのポートアクセスエラーを保存するために使用します。

TACCESS_ERR_INFO クラスのプロパティの中に使用されます。

３．16．１　コンストラクタ

TACCESS_ERR_PORT クラスのインスタンスを生成します。
次のコンストラクターも有効です。

```
public TACCESS_ERR_PORT( int port, int err_code, string err_text)
 {
      set(port, err_code, err_text);
 }
```

３．16．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int port	ポーと ID です。
2	public int errcode	エラーコードです。
3	public string errtext	エラーテキストです

３．16．３　メソッド

ポート TACCESS_ERR_PORT クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	ポート ID、 エラーコード、エラーテキストを設定します。

３．16．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．16．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．16．３．３　set() - ポートアクセスモードエラーの設定

プロパティに引数で与えられるエラー情報を設定します。

【構文】

public void set(int port, int err_code, string err_text)

【引数】

port

　　ポート ID です。

err_code

　　エラーコードです。

err_text

　　エラーテキストです。

【戻り値】

なし。

【説明】

引数 port, err_code, err_text をプロパティ port, errcode, errtext に設定します。

３．17　TOBJ_S14_ERR_INFO - コントロールジョブメッセージ応答情報クラス

TOBJ_S14_ERR_INFO クラスは、コントロールジョブ生成／削除メッセージ S14F9, S14F11 に対する応答情報を保存します。

S14F10, S14F12 メッセージのエンコード／デコード処理で使用されます。

３．17．１　コンストラクタ

TOBJ_S14_ERR_INFO クラスのインスタンスを生成します。

３．17．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public TCJ_INFO cj_info	コントロールジョブ生成
2	public int objack	応答 ack です。
3	public int err_count	エラー情報のカウントです。
4	public TERR_INFO_LIST err_list	エラー情報保存リストです。

３．17．３　メソッド

コントロールジョブ、応答 TOBJ_S14_ERR_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int set_TCJ_INFO()	CJ 情報を cj_info に設定します。
4	public void set_objspec()	cj_info プロパティの objspec を設定します。
5	public void set_objack()	objack を設定します。
6	public int add_attr()	属性情報を追加します。
7	public void add_err_info()	エラー情報を追加します。
8	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．17．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

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

３．17．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．17．３．３　set_TCJ_INFO() - CJ 情報の設定

コントロールジョブ情報をプロパティに設定します。

【構文】

public int set_TCJ_INFO(TCJ_INFO obj_info)

【引数】

obj_info

コントロールジョブ情報です。

戻り値】

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

【説明】

引数 obj_info の内容をプロパティ cj_info に設定します。

３．17．３．４　set_objspec () - TCJ_INFO の設定

cj_info のプロパティ objspec に OBJSPEC を設定します。

【構文】

public void set_objspec(string objspec)

【引数】

objspec

OBJSPEC です。

【戻り値】

なし。

【説明】

引数 objspec をプロパティ cj_info のプロパティ objspec に設定します。

３．17．３．５　set_objack()－ACK の設定

S14F10 の objack を設定します。

【構文】

public void set_objack(int ack)

【引数】

objack

ACK です。

【戻り値】

なし。

【説明】

引数 ack をプロパティ objack に設定します。

３．17．３．６　add_attr()－CJ 属性情報の追加

コントロールジョブ情報の中に属性情報を追加します。

【構文】

public int add_attr (TOBJ_ATTR_INFO info)

【引数】

info

CJ の属性情報です。

戻り値】

返却値	意　味
= 0	追加できた。
(-1)	info が null であった。

【説明】

引数 info の内容をプロパティ cj_info のプロパティ attr_list 属性情報リストに追加します。

３．17．３．７　add_err_info()－エラー情報の追加

エラー情報をエラーリストに追加します。

【構文】

public void add_err(int err_code, string err_text)

【引数】

err_code

　　エラーコードです。

err_text

　　エラーテキストです。

【戻り値】

なし。

【説明】

err_list に err_code と err_text を追加します。
追加したあと、err_count + 1 します。

３．17．３．８　copy()－インスタンスのコピー

TOBJ_S14_ERR_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

public static int copy(ref TOBJ_S14_ERR_INFO dst, TOBJ_S14_ERR_INFO src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．18　TRCP_ERR_INFO - レシピ関連メッセージ応答情報クラス

TRCP_ERR_INFO クラスは、レシピ関連メッセージ S15F3, S15F5, S15F13 に対する応答情報を保存します。

S15F4, S15F6, S15F14 メッセージのエンコード／デコード処理で使用されます。

３．18．１　コンストラクタ

TRCP_ERR_INFO クラスのインスタンスを生成します。
ack を引数とするコンストラクターもあります。

```
public TRCP_ERR_INFO(int ack) { rmack = ack; }
```

３．18．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int rmack	応答 ack です。
2	public int err_count	エラー数です。
3	public TERR_INFO[] err_list	エラー情報リストです。

３．18．３　メソッド

レシピ関連 S15F14, S15F4, S15F6 の応答 TRCP_ERR_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_rmack(int ack)	rmack を設定します。
4	public void add_err()	引数 err_code と err_text のエラー情報を追加します。
5	public void add_err_info()	引数 TERR_INFO クラスのインスタンス rsp_info の内容を追加します。
6	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．18．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．18．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．18．３．３　set_rmack()－ACK 情報の設定

S15F14 応答メッセージの rmack をプロパティ acka に設定します。

【構文】

public void set_rmack(int ack)

【引数】

rmacka

応答 ACK です。

【戻り値】

なし。

【説明】

引数の ack を設定します。プロパティ rmack に設定します。

３．18．３．４　add_err()－エラー情報の追加

S15F14 応答メッセージの err_list にエラー情報を追加します。

【構文】

public void add_err(int err_code, string err_text)

【引数】

err_code

エラーコードです。

err_text

エラーテキストです。

【戻り値】

なし。

【説明】

引数の err_code, err_text をプロパティ list に追加します。

３．18．３．５　add_err_info()－エラー情報の追加

S15F14 応答メッセージの err_list にエラー情報を追加します。

【構文】

public void add_err_info(TERR_INFO rsp_info)

【引数】

rsp_info

　　エラーコードとエラーテキストが保存されています。

err_text

　　エラーコードです。

【戻り値】

なし。

【説明】

引数の rsp_info のエラー情報をプロパティ err_list に追加します。

３．18．３．６　copy()－インスタンスのコピー

TRCP_ERR_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．19　TRCP_ACT_INFO - レシピ名アクション情報クラス

TRCP_ACT_INFO クラスは、レシピ名アクション情報を保存します。

S15F3 メッセージのエンコード／デコード処理で使用されます。

３．19．１　コンストラクタ

TRCP_ACT_INFO クラスのインスタンスを生成します。

３．19．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string rmnsspec	レシピ ID です。
2	public int rmnscmd	レシピ名アクションです。

３．19．３　メソッド

レシピアクション、ポートアクション応答 TRCP_ACT_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void clear()	すべてのプロパティの値をクリアします。
2	public void set_rcpid()	レシピ ID を設定します。
3	public void set_action()	レシピ名アクションを設定します。

３．19．３．１　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．19．３．２　set_rcpid() - レシピ ID の設定

レシピ ID をプロパティ rmnsspec に設定します。

【構文】

public void set_rcpid(string id)

【引数】

id

レシピ ID です。

【戻り値】

なし。

【説明】

引数 id をプロパティ rmnsspec に設定します。

３．19．３．３　set_action() - レシピ名アクションコマンドの設定

レシピ名アクションコマンドをプロパティ rmnscmd に設定します。

【構文】

public void set_action(int cmd)

【引数】

action

レシピ名アクションコマンドです。

【戻り値】

なし。

【説明】

引数 action をプロパティ rmnscmd に設定します。

３．20　TRCP_RENAME_INFO - レシピ改名情報クラス

TRCP_RENAME_INFO クラスは、レシピ名の改名情報を保存します。
S15F5 メッセージのエンコード／デコード処理で使用されます。

３．20．１　コンストラクタ

TRCP_RENAME_INFO クラスのインスタンスを生成します。

３．20．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string rmnsspec	レシピ ID です。
2	public string rmnewns	新レシピ ID です。

３．20．３　メソッド

レシピアクション、ポートアクション応答 TRCP_RENAME_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void clear()	すべてのプロパティの値をクリアします。
2	public void set_rcpid()	レシピ ID を設定します。
3	public void set_rmnewns ()	新レシピ名（ID）を設定します。

３．20．３．１　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．20．３．２　set_rcpid() - レシピ ID の設定

レシピ ID をプロパティ rmnsspec に設定します。

【構文】

public void set_rcpid(string id)

【引数】

id

レシピ ID です。

【戻り値】

なし。

【説明】

引数 id をプロパティ rmnsspec に設定します。

３．20．３．３　set_rmnewns()－新レシピ ID の設定

新レシピ名を設定します。

【構文】

public void set_rmnewns(string new_id)

【引数】

new_id

新レシピ ID です。

【戻り値】

なし。

【説明】

引数 new_id をプロパティ rmnewns に設定します。

３．21　TRCP_S15F8_INFO - S15F7 メッセージ応答情報クラス

TRCP_S15F8_INFO クラスは、レシピ関連メッセージ S15F7 に対する応答情報を保存します。

S15F8 メッセージのエンコード／デコード処理で使用されます。

３．21．１　コンストラクタ

TRCP_S15F8_INFO クラスのインスタンスを生成し、　エラー情報 TRC_ERR_INFO のインスタンスも生成します。
引数を伴うコンストラクター―もあります。

```
public TRCP_S15F8_INFO(uint rmspace, int rmack)
{
    rcp_rsp = new TRCP_ERR_INFO();
    this.rmspace = rmspace;
    this.rcp_rsp.rmack = rmack;
}
```

３．21．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public uint rmspace	レシピ名の空きスペースです。
2	public TRCP_ERR_INFO rcp_rsp	エラー応答情報です。

３．21．３　メソッド

レシピ関連 S15F8 の応答 TRCP_S15F8_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_rmspace()	rmspace を設定します。
4	public void add_err()	引数 err_code と err_text のエラー情報を追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．21．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

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

３．21．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．21．３．３　set_rmspace()- レシピ名保存スペースの設定

引数 space をプロパティ rmspace に設定します。

【構文】

public void set_rmspace(uint space)

【引数】

space

　　レシピ名保存スペースです。

【戻り値】

なし。

【説明】

引数 space をプロパティ rmspace に設定します。

３．21．３．４　add_err()- エラー情報の追加

S15F8 応答メッセージのエラー情報を rsp_rsp プロパティに追加します。

【構文】

public void add_err(int err_code, string err_text)

public void add_err(TERR_INFO rsp_info)

【引数】

err_code

　　エラーコードです。

err_text

　　エラーテキストです。

rsp_rsp

　　エラー情報が保存されている TERR_INFO のインスタンスです。

【戻り値】

なし。

【説明】

最初のメソッドは、引数 err_code と err__text のエラー情報を、プロパティ rcp_rsp に追加します。

２つ目のメソッドも同様に rsp_info を rcp_rsp にエラー情報を追加します。

３．21．３．５　copy ()　－ インスタンスのコピー

TRCP_S15F8_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

```
public static int copy(ref TRCP_S15F8_INFO dst, TRCP_S15F8_INFO src)
```

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．22　TRCP_S15F10_INFO - S15F9 メッセージ応答情報クラス

TRCP_S15F10_INFO クラスは、レシピ関連メッセージ S15F9 に対する応答情報を保存します。
S15F10 メッセージのエンコード／デコード処理で使用されます。

３．22．１　コンストラクタ

TRCP_S15F10_INFO クラスのインスタンスを生成し、　エラー情報 TRCP_ERR_INFO のインスタンスも生成します。
引数を伴うコンストラクター―もあります。

```
public TRCP_S15F10_INFO( int stat, string ver, int ack);
```

各引数を各プロパティに設定します。

３．22．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int rcpstat	レシピ状態です 。
2	public string rcpver	レシピのバージョンです。
3	public TRCP_ERR_INFO rcp_rsp	エラー応答情報です。

３．22．３　メソッド

レシピ関連 S15F10 の応答 TRCP_S15F10_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_status_version()	rmack, status と version を設定します。
4	public void set_rmack()	rmack を設定します。 rcp_rsp.rmack =rmack;
5	public void add_err()	引数 err_code と err_text のエラー情報を追加します。
6	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．22．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．22．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．22．３．３　set_status_version()- レシピ状態とバージョンの設定

引数 status，version をそれぞれ、プロパティ rcpstat と rcpver に設定します。

【構文】

public void set_status_version(int status, string version)

【引数】

status

　　レシピのステータスです。

version

　　レシピのバージョンです。

【戻り値】

なし。

【説明】

引数 status，version をそれぞれ、プロパティ rcpstat と rcpver に設定します。

３．22．３．４　add_err()- エラー情報の追加

S15F10 応答メッセージのエラー情報を rsp_rsp プロパティに追加します。

【構文】

public void add_err(int err_code, string err_text)

【引数】

err_code

　　エラーコードです。

err_text

　　エラーテキストです。

【戻り値】

なし。

【説明】

引数 err_code と err_text、プロパティ rcp_rsp にエラー情報を追加します。

３．23　TRCP_RETRIEVE_INFO - S15F17 メッセージ情報クラス

TRCP_RETRIEVE_INFO クラスは、レシピ関連メッセージ S15F17 に対する情報を保存します。
S15F17 メッセージのエンコード／デコード処理で使用されます。

３．23．１　コンストラクタ

TRCP_RETRIEVE_INFO クラスのインスタンスを生成します。

３．23．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string rcpid	レシピ ID です 。
2	public int seccode	レシピのセクションです。

３．23．３　メソッド

レシピ関連 S15F10 の応答 TRCP_RETRIEVE_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void clear()	すべてのプロパティの値をクリアします。
2	public void set_rcpid()	レシピ ID を設定します。
3	public void set_seccode()	レシピセクションコードを設定します。

３．23．３．１　clear()- プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．23．３．２　set_rcpid()- レシピ ID の設定

レシピ ID を設定します。

【構文】

public void set_rcpid(string id)

【引数】

id

レシピ ID です。

【戻り値】

なし。

【説明】

引数の id をプロパティ rcpid に設定します。

３．23．３．３　set_seccode()- レシピセクションコードの設定

レシピセクションコードを設定します。

【構文】

public void set_seccode(int code)

【引数】

code

レシピセクションコードです。

【戻り値】

なし。

【説明】

引数の code をプロパティ seccode に設定します。

３．24　TRCP_S15F18_INFO - S15F17 メッセージ応答情報クラス

TRCP_S15F18_INFO クラスは、レシピ検索要求メッセージ S15F17 に対する応答情報を保存します。

S15F18 メッセージのエンコード／デコード処理で使用されます。

３．24．１　コンストラクタ

TRCP_S15F18_INFO クラスのインスタンスを生成し、　エラー情報 TRC_ERR_INFO のインスタンスも生成します。
引数を伴うコンストラクター―もあります。

```
public TRCP_S15F18_INFO(  string rcpbody,  int rmack)
 {
      this.rcpbody = rcpbody;
      this.rmack = rmack;
 }
```

３．24．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int q_count	S15F18 の１番目の format L の情報数(L-q)です。
2	public int r_count	上の L-q の下にリンクされる情報数(L-r)です。
3	public string rcpsecm	レシピのセクション名です。
4	public TRCP_SECNM g_secnm	包括的属性情報を保存します。
5	public string rcpbody	レシピ本体です。
6	public int m_count	エージェント固有のデータセット数です。 (L-m)
7	public TRCP_SECNM[] m_secnm	エージェント固有の属性情報リストです。
8	public int rmack	ACK です。
9	public int err_count	エラー情報の数です。
10	public TERR_INFO[] err_list	エラー情報保存用リストです。

３．24．３　メソッド

レシピ関連 S15F18 の応答 TRCP_S15F18_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_rcpbody()	レシピ本体を設定します。
4	public void set_rmack()	RMACK を設定します。
5	public void add_g_sect_info()	包括的属性情報を設定します。
6	public void add_m_sect_info()	エージェント固有のデータセット情報を追加します。 m_secrm[]リストに
7	public void add_err_info()	応答 error 情報を追加します。
8	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．24．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．24．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．24．３．３　set_rcpbody() - レシピ本体の設定

rcpbody を設定します。

【構文】

public void set_rcpbody(string rcpbody)

【引数】

rcpbody

レシピ本体です。

【戻り値】

なし。

【説明】

引数 rcpbody をプロパティ rcpbody に設定します。

３．24．３．４　set_rmack() - RMACK の設定

rmack を設定します。

【構文】

public void set_rmack(int ack)

【引数】

ack

ACK です。

【戻り値】

なし。

【説明】

引数 ack をプロパティ rmack に設定します。

３．24．３．５　add_g_sect_info() -包括的属性情報の追加

包括的属性情報を追加します。

【構文】

public void add_g_sect_info(TRCP_SECNM sinfo)

【引数】

sinfo

属性情報です。

【戻り値】

なし。

【説明】

引数 sinfo をプロパティ g_secnm に追加します。

３．24．３．６　add_m_sect_info() -エージェント固有データセット属性情報の追加

エージェント固有データセット属性情報を追加します。

【構文】

public void add_m_sect_info(TRCP_SECNM sinfo)

【引数】

sinfo

属性情報です。

【戻り値】

なし。

【説明】

引数 sinfo をプロパティ m_secnm に追加します。

３．24．３．７　add_err()- エラー情報の追加

S15F18 応答メッセージのエラー情報を err_list リストプロパティに追加します。

【構文】

public void add_err(int err_code, string err_text)

【引数】

err_code

　　エラーコードです。

err_text

　　エラーテキストです。

【戻り値】

なし。

【説明】

引数 err_code と err_text のエラー情報をプロパティ err_list に追加します。

３．24．３．８　copy()-- インスタンスのコピー

TRCP_S15F18_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

public static int copy(ref TRCP_S15F18_INFO dst, TRCP_S15F18_INFO src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．25　TRCP_SECNM – データセット属性情報クラス

TRCP_SECNM クラスは、S15F18 の応答情報 TRCP_S15F18_INFO クラスのプロパティのセクション情報の保存に使用されます。
S15F18 メッセージのエンコード／デコード処理で使用されます。

３．25．１　コンストラクタ

TRCP_SECNM クラスのインスタンスを生成します。
引数を伴うコンストラクター―もあります。

```
public TRCP_SECNM( rcpsecrm)
{
      this. rcpsecrm = rcpsecrm;
}
```

３．25．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string rcpsecnm	セクション名です。
2	public int attr_count	属性カウントです。
3	public TRCP_ATTR[] attr_list	セクション情報リストです。 TRCP_ATTR ： Vol-2 12.3 参照

３．25．３　メソッド

レシピ関連 S15F18 の応答 TRCP_SECNM クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_rcpsecnm()	セクション名をセットします。
4	public void add_attr()	属性情報を設定します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．25．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

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

３．25．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．25．３．３　set_rcpsecrm() - セクション名の設定

レシピセクション名を設定します。

【構文】

public void set_rcpsecrm (string rcpsecrm)

【引数】

rcpsecnm

　　レシピセクション名です。

【戻り値】

なし。

【説明】

引数 rcpsecrm をプロパティ rcpsecrm に設定します。

３．25．３．４　add_attr() - エラー情報の追加

属性情報を属性情報リストプロパティに追加します。

【構文】

public void add_attr(string attrid, int format, int size, IntPtr data)

【引数】

attrid

　　属性 ID です。

format

　　データのフォーマットです。

size

　　データサイズです。

data

　　属性値の保存メモリです。

【戻り値】

なし。

【説明】

引数 attrid と、format, size, data の属性値を attr_list リストに追加します。追加した後、attr_count + 1 します。

３．25．３．５　copy()-- インスタンスのコピー

TRCP_SECNM のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．26　TPRJ_ERR_INFO - プロセスジョブ関連メッセージ応答情報クラス

TPRJ_ERR_INFO クラスは、プロセスジョブ関連メッセージ S16F11, S16F5 に対する応答情報を保存します。

S16F12, S16F6 メッセージのエンコード／デコード処理で使用されます。

３．26．１　コンストラクタ

TPRJ_ERR_INFO クラスのインスタンスを生成します。
ack を引数とするコンストラクター―もあります。

```
public TPRJ_ERR_INFO(bool ack) { acka = ack; }
```

３．26．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int acka	応答 ack です。
2	public int err_count	エラー数です。
3	public TERR_INFO[] err_list	エラー情報リストです。

３．26．３　メソッド

プロセスジョブ関連 S16F14, S16F4, S16F6 の応答 TPRJ_ERR_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	prjid と acka を設定します。
4	public void add_err()	引数 err_code と err_text のエラー情報を追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．26．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

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

３．26．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．26．３．３　set() - プロセスジョブ ID と ACKA の設定

S16F14 応答メッセージの prjid と acka をそれぞれプロパティ prjid, acka に設定します。

【構文】

public void set(string prjid, bool acka)

【引数】

prjid

　　プロセスジョブ ID です。

acka

　　応答 ACK です。

【戻り値】

なし。

【説明】

引数の prjid と acka をそれぞれプロパティ prjid, acka に設定します。

３．26．３．４　add_err() - エラー情報の追加

S16F14 応答メッセージの err_list にエラー情報を追加します。

【構文】

public void add_err(int err_code, string err_text)

【引数】

err_code

　　エラーコードです。

err_text

　　エラーコードです。

【戻り値】

なし。

【説明】

引数の err_code, err_text をプロパティ err_list に追加します。

３．26．３．５　copy()- インスタンスのコピー

TPRJ_ERR_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．27　TMPRJ_ERR_INFO - マルチプロセスジョブ関連メッセージ応答情報クラス

TMPRJ_ERR_INFO クラスは、マルチプロセスジョブ関連メッセージ S16F15 に対する応答情報を保存します。

S16F16 メッセージのエンコード／デコード処理で使用されます。

３．27．１　コンストラクタ

TMPRJ_ERR_INFO クラスのインスタンスを生成します。
ack を引数とするコンストラクターもあります。

```
public TMPRJ_ERR_INFO(bool ack) { acka = ack; }
```

３．27．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int prj_count	プロセスジョブの数です。 (prj_list に設定されている ID の数です)
2	public string[] prj_list	プロセスジョブ ID リストです。
3	public int acka	応答 ack です。
4	public int err_count	エラー数です。
5	public TERR_INFO[] err_list	エラー情報リストです。

３．27．３　メソッド

プロセスジョブ関連 S16F14, S16F4, S16F6 の応答 TMPRJ_ERR_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	prjid とその数、acka を設定します。
4	public void add_err()	引数 err_code と err_text のエラー情報を追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．27．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

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

３．27．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．27．３．３　set() - プロセスジョブ ID リストと ACKA の設定

S16F14 応答メッセージの prjid と acka をそれぞれプロパティ prjid, acka に設定します。

【構文】

public void set(string[] prj_list, int prj_count, bool acka)

【引数】

prjid_list

　　プロセスジョブ ID リストです。

prj_count

　　プロセスジョブ ID の数です。

acka

　　応答 ACK です。

【戻り値】

なし。

【説明】

引数の prjid_list リストにある pprj_count 分のプロセスジョブ ID と acka をそれぞれプロパティ prjid_list, acka に設定します。

３．27．３．４　add_err() - エラー情報の追加

S16F14 応答メッセージの err_list にエラー情報を追加します。

【構文】

public void add_err(int err_code, string err_text)

【引数】

err_code

　　エラーコードです。

err_text

　　エラーコードです。

【戻り値】

なし。

【説明】

引数の err_code, err_text をプロパティ err_list に追加します。

３．27．３．５　copy()- インスタンスのコピー

TMPRJ_ERR_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

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

【説明】

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

３．28　TCJ_CMD_ERR_INFO - コントロールジョブコマンドメッセージ応答情報クラス

TCJ_CMD_ERR_INFO クラスは、コントロールジョブコマンドメッセージ S16F27 に対する応答情報を保存します。

S16F28 メッセージのエンコード／デコード処理で使用されます。

３．28．１　コンストラクタ

TCJ_CMD_ERR_INFO クラスのインスタンスを生成します。

３．28．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int acka	応答 ack です。
2	public TERR_INFO　rsp_info	エラー情報です。 =null ならば、エラー情報なしとします。

３．28．３　メソッド

コントロールジョブコマンド S16F27 の応答 TCJ_CMD_ERR_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_acka(int acka)	acka を設定します。
4	public void set_acka_err(int acka, int err_code, string err_text)	acka とエラー情報を設定します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．28．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

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

３．28．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．28．３．３　set_acka()－ACK 情報の設定

S16F28 応答メッセージの acka をプロパティ acka に設定します。

【構文】

public void set_acka(int acka)

【引数】

acka

応答 ACK です。

【戻り値】

なし。

【説明】

引数の acka をプロパティ acka に設定します。

３．28．３．４　set_acka_err()－ACK とエラー情報の設定

S16F28 応答メッセージの acka とエラー情報を設定します。

【構文】

public void set_acka_err(int acka, int err_code, string err_text)

【引数】

acka

応答 ACK です。

err_code

エラーコードです。

err_text

エラーテキストです。

【戻り値】

なし。

【説明】

引数の acka をプロパティに設定します。

また、err_code と err_text は、プロパティ rsp_info を生成し、err_code と err_text を設定します。

３．28．３．５　copy()- インスタンスのコピー

TCJ_CMD_ERR_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．29　TSPOOL_ERR_INFO – S2F44 メッセージ情報クラス

TSPOOL_ERR_INFO クラスは、Spool 関連メッセージ S2F43 に対する応答情報を保存します。
S2F44 メッセージのエンコード／デコード処理で使用されます。

３．29．１　コンストラクタ

TSPOOL_ERR_INFO クラスのインスタンスを生成します。

３．29．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int rsack	rsack です 。
2	public int err_count	エラー数です。
3	public TSTRE_ERR_INFO[] stre_list	エラー詳細項目です。

本クラスは、下記クラスを使用します。

```
public class TSTRE_ERR_INFO
{
    public int flag;            // 有効/無効      1=有効
    public int strack;
    public int stream;
    public int f_count;
    public int[] func_list;
}
```

３．29．３　メソッド

レシピ関連 S15F10 の応答 TSPOOL_ERR_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void init(int rsack)	this.rsack=rsack err_count=0 stre_list[128]にします。
3	public void clear()	rs_ack=0, err_count=0 stre_list[128]にします。
4	public int set_strack(int stream, int strack)	stream で指定された stre_list の位置に strack を設定します。
5	public int add_err(int stream, int func)	stream で指定された stre_list の func_list に func を追加設定します。

３．29．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

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

３．29．３．２　init() - プロパティの初期化

当該インスタンスの内容をすべて消去し、rsack を設定します。

【構文】

public void init(int rsack)

【引数】

rsack
　　S2F44 メッセージの ack 値
　　(0 or 1)

【戻り値】

なし。

【説明】

当該インスタンスの内容をすべて消去し、rsack を設定します。

３．29．３．３　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．29．３．４　set_strack ()－stream 指定の strack の設定

指定された stream に対する strack を設定します。

【構文】

public int set_strack(int stream, int strack)

【引数】

stream

SxFy の stream

strack

stream の ack

【戻り値】

stream が 1～127 の値ならば 0 を、それ以外は、(-1)を返却します。

【説明】

stre_lis プロパティの stream の strack の値を設定します。

(stre_list[stream].strack = strack)

３．29．３．５　add_err()－stream, function の設定

指定された stream に対する strack を設定します。

【構文】

public int add_err(int stream, int func)

【引数】

stream

　　SxFy の stream

func

　　SxFy の function

【戻り値】

stream または function の値が範囲外の場合は(-1)を返却します。

【説明】

stre_lis プロパテイの stream の func_list に func を設定します。

（ stre_list[stream].func_list[stre_list[stream].f_count] = func ）

そして、stre_list[stream].f_count + 1、stre_list[stream].flag = 1 にします。

３．30　TRCMD_ERR_INFO – S2F42 メッセージ情報クラス

TRCMD_ERR_INFO クラスは、ホストコマンド関連メッセージ S2F41, S2F49 に対する応答情報を保存します。
S2F42、F50 メッセージのエンコード／デコード処理で使用されます。

３．30．１　コンストラクタ

TRCMD_ERR_INFO クラスのインスタンスを生成します。

３．30．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int hc_ack	受信結果を示す hc_ack です 。
2	public int err_count	エラー数です。
3	public TCP_ACK_INFO[] err_list	エラー詳細項目です。

３．30．３　メソッド

レシピ関連 S2F42，S2F50 の応答 TRCMD_ERR_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	this.hc_ack=0 err_count=0 err_list[0]にします。
3	public void add_err(string name, int ack)	err_list に name，ack を追加設定します。
4	public static int copy(ref TCP_ACK_INFO dst, TCP_ACK_INFO src)	stream で指定された err_list の位置に strack を設定します。

３．30．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

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

３．30．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．30．３．３　add_err()－エラー情報の追加設定

hc_ack != 0 の理由、パラメータ名と、その値を追加設定します。

【構文】

public void add_err(string name, int ack)

【引数】

name

　　パラメータ名

ack

　　理由を示す値

【戻り値】

なし。

【説明】

hc_ack != 0 を理由のパラメータ名と、その値を err_list[]に追加設定します。
設定したあと、err_count の値を　+1 します。

３．30．３．４　copy()- インスタンスのコピー

TCP_ACK_INFO クラスのインスタンスを他のインスタンスにコピーします。

【構文】

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

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

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

【説明】

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

３．31　TCP_ACK_INFO – S2F42 の ack 理由パラメータ情報クラス

TCP_ACK_INFO クラスは、ホストコマンド関連メッセージ S2F41, S2F49 に対する ack が正常でなかった場合に付加するパラメータ情報を保存します。
本クラスは、TRCMD_ERR_INFO クラスのプロパティのメンバーです。

３．31．１　コンストラクタ

TCP_ACK_INFO クラスのインスタンスを生成します。

３．31．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string cp_name	パラメータ名です 。
2	public int cp_ack	理由コードです。

３．31．３　メソッド

レシピ関連 S15F10 の応答 TCP_ACK_INFO クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	this.hc_ack=0 err_count=0 err_list[0]にします。

３．31．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

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

３．31．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイをすべてクリアします。

３．32　TTERM_MULTI_INFO – 端末表示テキストリスト情報クラス

TTERM_MULTI_INFO クラスは、テキスト・リスト情報を保存するために使用します。
S10F5 メッセージの送信時に使用します。

３．32．１　コンストラクタ

３．32．１．１　コンストラクタ

TTERM_MULTI_INFO クラスのインスタンスを生成します。

３．32．１．２　デストラクタ

TTERM_MULTI_INFO のインスタンスを破棄します。
デストラクタはシステムから呼び出され、Dispose()メソッドを実行します。
Dispose()は、そのインスタンスが使用している資源（Unmanaged Memory）をシステムに返却します。

３．32．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int tid	端末 ID
2	public int text_count	保存されているテキスト（文字列）数
3	public string[] text_list	テキスト保存リスト

３．32. ３　メソッド

コントロールジョブ情報クラス TTERM_MULTI_INFO のメソッドは下表のとおりです。

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

3．32．3．1　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

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

3．32．3．2　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．32．３．３　add_msg() - テキストの追加

テキスト１個を text_list 配列リストに追加します。

【構文】

public void add_msg (string text)

【引数】

text

テキスト（文字列）

【戻り値】

なし。

【説明】

テキストをプロパティ text_list リストに追加します。
追加した後、プロパティ、text_count +1 します。

６．３．４．４　copy() - インスタンスのコピー

TTERM_MULTI_INFO クラスのインスタンスをコピーします。

【構文】

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

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

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

【説明】

src から dst インスタンスへコピーします。

付録－Ａ　DSHDR2 通信ドライバーが提供するメソッド一覧表

	機能概略	構文
1	ﾃﾞｰﾀｱｲﾃﾑ取得の初期化	public static extern int D_InitItemGet(ref DSHMSG msg)
2	ﾃﾞｰﾀｱｲﾃﾑの取得	public static extern int D_GetItem(ref DSHMSG msg, int icode, IntPtr data, int size); public static extern int D_GetItem(ref DSHMSG msg, int icode, byte[] data, int size); public static extern int D_GetItem_B(ref DSHMSG msg, int icode, // B ref byte data,int size); public static extern int D_GetItem_BOOL(ref DSHMSG msg, int icode, // Bool ref byte data, int size); public static extern int D_GetItem_I1(ref DSHMSG msg, int icode, // I1 ref byte data, int size); public static extern int D_GetItem_I2(ref DSHMSG msg, int icode, // I2 ref Int16 data, int size); public static extern int D_GetItem_I4(ref DSHMSG msg, int icode, // I4 ref Int32 data, int size); public static extern int D_GetItem_I8(ref DSHMSG msg, int icode, // I8 ref Int64 data, int size); public static extern int D_GetItem_U1(ref DSHMSG msg, int icode, // U1 ref byte data, int size); public static extern int D_GetItem_U2(ref DSHMSG msg, int icode, // U2 ref UInt16 data, int size); public static extern int D_GetItem_U4(ref DSHMSG msg, int icode, // U4 ref UInt32 data, int size); public static extern int D_GetItem_U8(ref DSHMSG msg, int icode, // U8 ref UInt64 data, int size); public static extern int D_GetItem_F4(ref DSHMSG msg, int icode, // F4 ref Single data, int size); public static extern int D_GetItem_F8(ref DSHMSG msg, int icode, // F8 ref Double data, int size); public static extern int D_GetItem_Bool(ref DSHMSG msg, int icode, // Boolean ref Boolean data, int size);
3	ﾃﾞｰﾀｱｲﾃﾑ設定の初期化	public static extern int D_InitItemPut(ref DSHMSG msg);
4	ﾃﾞｰﾀｱｲﾃﾑの設定	public static extern int D_PutItem(ref DSHMSG msg, int icode, IntPtr data, int size); public static extern int D_PutItem(ref DSHMSG msg, int icode, byte[] data, int size); public static extern int D_PutItem_B(ref DSHMSG msg, int icode, // B ref byte data,int size); public static extern int D_PutItem_BOOL(ref DSHMSG msg, int icode, // Bool ref byte data, int size); public static extern int D_PutItem_I1(ref DSHMSG msg, int icode, // I1 ref byte data, int size); public static extern int D_PutItem_I2(ref DSHMSG msg, int icode, // I2

		ref Int16 data, int size); public static extern int D_PutItem_I4(ref DSHMSG msg, int icode, // I4 ref Int32 data, int size); public static extern int D_PutItem_I8(ref DSHMSG msg, int icode, // I8 ref Int64 data, int size); public static extern int D_PutItem_U1(ref DSHMSG msg, int icode, // U1 ref byte data, int size); public static extern int D_PutItem_U2(ref DSHMSG msg, int icode, // U2 ref UInt16 data, int size); public static extern int D_PutItem_U4(ref DSHMSG msg, int icode, // U4 ref UInt32 data, int size); public static extern int D_PutItem_U8(ref DSHMSG msg, int icode, // U8 ref UInt64 data, int size); public static extern int D_PutItem_F4(ref DSHMSG msg, int icode, // F4 ref Single data, int size); public static extern int D_PutItem_F8(ref DSHMSG msg, int icode, // F8 ref Double data, int size); public static extern int D_PutItem_Bool(ref DSHMSG msg, int icode, // Boolean ref Boolean data, int size);