DshGemMsgPro　GEM メッセージ・エンコード/デコード

ソフトウェア・ライブラリ

API 関数説明書

(C, C++, .Net-Vb,C#)

Vol-２／３

３．API 関数（続き）

S3Fx ： S3F17, S3F23, S3F25, S3F27

S5Fx ： S5F1, S5F3, S5F5

S6Fx ： S6F1, S6F11, S6F15, S6F19

S7Fx ： S7F1, S7F3, S7F5, S7F17, S7F19, S7F23, S7F25, S7F29

２０１３年９月

株式会社データマップ

文書番号　DshGemMsgPro-13-30322-00

［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.	2013 年 9 月	初版	

目　次

[GEM-PRO　関連ドキュメント]

GEM-PRO ドキュメント一覧表

	文書番号	タイトル名と内容
1	DshGemMsgPro-13-30321-00　Vol-1	DshGemMsgPro GEM ﾒｯｾｰｼﾞ・ｴﾝｺｰﾄﾞ/ﾃﾞｺｰﾄﾞ　API 関数説明書 1. 概要 2. 機能概略 3. API 関数 3.1　GEM-PRO 初期化関数とバージョン取得関数 3.2　S1Fx, S2Fx ﾒｯｾｰｼﾞｴﾝｺｰﾄﾞ・ﾃﾞｺｰﾄﾞ関数
	DshGemMsgPro-13-30322-00　Vol-2	(3.2)　S3Fx, S5Fx, S6Fx, S7Fx
	DshGemMsgPro-13-30323-00　Vol-3	(3.2)　S10Fx, S14Fx, S15Fx, S16Fx
2	DshGemMsgPro-13-30331-00　Vol-1	DshGemMsgPro GEM ﾒｯｾｰｼﾞ・ｴﾝｺｰﾄﾞ/ﾃﾞｺｰﾄﾞ　LIB 関数説明書 ・変数(EC、SV、DVVAL)関連 ・ﾚﾎﾟｰﾄ、収集ｲﾍﾞﾝﾄ(CE)関連 ・ｱﾗｰﾑ関連 ・ﾌﾟﾛｾｽ・ﾌﾟﾛｸﾞﾗﾑ(PP、FPP)関連 ・ﾚｼﾋﾟ関連 ・ﾌﾟﾛｾｽ・ｼﾞｮﾌﾞ関連 ・ｺﾝﾄﾛｰﾙ・ｼﾞｮﾌﾞ関連
	DshGemMsgPro-13-30332-00　Vol-2	・ﾘﾓｰﾄｺﾝﾄﾛｰﾙ、拡張ﾘﾓｰﾄｺﾝﾄﾛｰﾙ関連 ・ｷｬﾘｱｱｸｼｮﾝ、ﾎﾟｰﾄ制御関連 ・端末表示関連 ・ｽﾌﾟｰﾙ関連 ・その他の汎用関数
3	DshGemMsgPro-13-30320-00	DshGemMsgPro GEM ﾒｯｾｰｼﾞ・ｴﾝｺｰﾄﾞ/ﾃﾞｺｰﾄﾞ　定数、構造体説明書
4	DshGemMsgPro-13-30381-00	DshGemMsgPro GEM ﾒｯｾｰｼﾞ・ｴﾝｺｰﾄﾞ/ﾃﾞｺｰﾄﾞ　ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ説明書

GEM-PRO に関する概要、機能については、”GEM-PRO API 関数説明書-VOL-1 “の１，２章をを参照してください。

３．API 関数　(Vol-1 からの続き)

３．２．19　S3F17 メッセージ　－　キャリアアクション要求情報の送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS3F17()	S3F17 をｴﾝｺｰﾄﾞします。	ｷｬﾘｱｱｸｼｮﾝ要求情報をｴﾝｺｰﾄﾞします。
2	DSH_DecodeS3F17()	S3F17 をﾃﾞｺｰﾄﾞします。	ｷｬﾘｱｱｸｼｮﾝ要求情報にﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS3F18()	S3F18 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	応答情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS3F18()	S3F18 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	応答情報を取得します。

（2）S3F17 のユーザインタフェース情報
情報の引き渡しは構造体 TCACT_INFO を使って行います。

①ｷｬﾘｱｱｸｼｮﾝ要求とパラメータを保存する構造体

```
typedef struct{
        TDATAID     dataid;
        char        *caction;          // cact
        int         action_index;
        char        *carspec;          // carrier spec ( carid )
        int         ptn;               // port no.
        int         cp_count;          // parameter count
        TCACT_PARA  **cp_list;         // paramete list
}TCACT_INFO;
```

②ｷｬﾘｱｱｸｼｮﾝ要求情報構造体に含む１個のパラメータ情報を保存する構造体

```
typedef struct{
        char        *cattrid;          // cattrid
        int         attr_index;
        void        *cattrdata;        // cattrdata
}TCACT_PARA;
```

③ｷｬﾘｱ・ｺﾝﾃﾝﾄ情報を保存する構造体（属性 ID が "ContentMap" のｹｰｽで使用する）

```
typedef struct{
        int         count;
        char        **lotid;           // lotid
        char        **substid;         // substrate id
}TCACT_CONTENT;
```

（3）TCACT_INFO 構造体への情報設定処理関連関数

C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。

.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTCACT_INFO	TCACT_INFO を初期設定する。
2	DshPutTCACT_INFO	TCACT_INFO に 1 個のﾊﾟﾗﾒｰﾀを加える。
3	DshMakeTCACT_PARA	１個の属性情報の設定とﾊﾟﾗﾒｰﾀ設定の準備をする。
4	DshPutTCACT_CONTENT	“ContentMap” 属性のﾊﾟﾗﾒｰﾀ情報を設定する。
5	DshFreeTCACT_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

（4）S3F18 のユーザインタフェース情報

応答情報を TCACT_ERR_INFO 構造体を使用します。

①S3F18 に設定する応答情報を格納するための構造体です。

```
typedef struct{
        int         caack;
        int         err_count;
        TERR_INFO   **err_list;
} TCACT_ERR_INFO;
```

②ｴﾗｰﾊﾟﾗﾒｰﾀ情報を１個分格納するための構造体です。（ｴﾗｰｺｰﾄﾞとｴﾗｰﾃｷｽﾄ）

```
typedef struct{
        int     errcode;
        char    *errtext;
} TERR_INFO;
```

（5）TCACT_ERR_INFO 構造体への情報設定処理関連関数

番号	関数名	機能
1	DshInitTCACT_ERR_INFO	TCACT_ERR_INFO を初期設定する。
2	DshPutTCACT_ERR_INFO	TCACT_ERR_INFO に 1 個のﾊﾟﾗﾒｰﾀを加える。
3	DshFreeTCACT_ERR_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

３．２．19．１　DSH_EncodeS3F17() －　S3F17 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS3F17(
        BYTE *buffer,
        int buff_size,
        TCACT_INFO *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS3F17(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef info As TCACT_INFO,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS3F17(
        IntPtr buffer,
        int buff_size,
        ref TCACT_INFO info,
        ref int msg_len
       );
```

（2）引数

buffer	：S3F17 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	：buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	：ｷｬﾘｱｱｸｼｮﾝ要求情報を格納するための構造体です。
msg_len	：ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S3F17 メッセージを作成します。
info で指定された構造体 TCACT_INFO 内に含まれるキャリアアクション要求情報を S3F17 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

（5）例

①C/C++

```
int      ei, i;
BYTE     buff[1024];
int      msg_len;
TCACT_INFO info;
TCACT_PARA pinfo;
char  lotid_tab [25][32];
char  substidtid_tab [25][32];
for ( i=0; i < 25; i++ ){
    sprintf_s( &lotid_tab[i][0],  32,  "LOTID-%02d", i );
    sprintf_s( &substid_tab[i][0], 32, "SUBSTID-%02d", i );
}

DshInitTCACT_INFO( &info, CA_ProceedWithCarrier, CARID, PORTID, ATR_COUNT );
DshMakeTCACT_PARA( &pinfo, CA_ContentMap,  (void*)MAX_SLOT );

for ( i=0; i < MAX_SLOT; i++ ){
        DshPutTCACT_CONTENT( &pinfo, i, lotid_tab[i], substid_tab[i] );
}
DshPutTCACT_INFO( &info, &pinfo );

ei = DSH_EncodeS3F17( buff, 1024, &info, &msg_len );           // encode
.
.
DshFreeTCACT_PARA( &pinfo );
DshFreeTCACT_INFO( &info );
```

（注）CA_ProceedWithCarrier は、アクション名であり、DshGemProInf.h で定義されています。

②c#

```
string  CARID      = "CAR100";
int     PORTID     = 1;
int     ATR_COUNT = 1;

int     MAX_SLOT  = 25;
IntPtr  MAX_SLOT_PTR = new IntPtr(25);

string[] lotid_tab = new string[MAX_SLOT];
string[] substid_tab = new string[MAX_SLOT];

int ei;
int msg_len = 0;
TCACT_INFO info  = new TCACT_INFO();
TCACT_PARA pinfo = new TCACT_PARA();

IntPtr buff = Marshal.AllocCoTaskMem(1024);
```

```
for ( int i=0; i < 25; i++ ){
    lotid_tab[i]   = "LOTID-" + i.ToString();
    substid_tab[i] = "SUBSTID-"+ i.ToString();
}

DshInitTCACT_INFO(ref info, CA_ProceedWithCarrier, CARID, PORTID, ATR_COUNT); ・

DshMakeTCACT_PARA(ref pinfo, CA_ContentMap, MAX_SLOT_PTR);
for (int i = 0; i < MAX_SLOT; i++)
{
    DshPutTCACT_CONTENT(ref pinfo, i, lotid_tab[i], substid_tab[i]);
}
DshPutTCACT_INFO(ref info, ref pinfo);
DshFreeTCACT_PARA( ref pinfo);

ei =DSH_EncodeS3F17(buff, 1024, ref info, ref msg_len);         // encode S3F17
.
.

Marshal.FreeCoTaskMem(buff);
DshFreeTCACT_INFO(ref info);
```

(注) CA_ProceedWithCarrier は、アクション名であり、DshGemProInf.cs で定義されています。

３．２．19．２　DSH_DecodeS3F17() －　S3F17 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS3F17(
        BYTE *buffer,
        int msg_len,
        TCACT_INFO *info
        );
```

[VB.Net]

```
Function DSH_DecodeS3F17(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TCACT_INFO
       ) As Integer
```

[C#]

```
int DSH_DecodeS3F17(
        IntPtr buffer,
       int msg_len,
       ref TCACT_INFO info
       );
```

（2）引数

buffer　　：S3F17 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。

msg_len　　：S3F17 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。
(Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

info　　：ｷｬﾘｱｱｸｼｮﾝ要求情報を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S3F17 メッセージのデコードを行います。
デコード結果のキャリアアクション要求情報は、info 構造体に格納されます。

（5）例

①c、C++

```
BYTE buff[2000];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S3F17 受信)

int msg_len = 195;                  // 受信した S3F17 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TCACT_INFO info;
int ei;
ei = DSH_DecodeS3F17( buff, msg_len, &info );
.
.
DshFreeTCACT_INFO( &info );
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S3F17 受信)

int msg_len = 195;                  // 受信した S3F17 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TCACT_INFO info = new TCACT_INFO();
int ei = DSH_DecodeS3F17( buff, msg_len, ref info );
.
.
DshFreeTCACT_INFO( ref info );
Marshal.FreeCoTaskMem(buff);
```

３．２．19．３　DSH_EncodeS3F18() －　S3F18 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX EncodeS3F18(
        BYTE *buffer,
        int buff_size,
        TCACT_ERR_INFO *erinfo,
        int *msg_len
        );
```

[VB.Net]

```
Function EncodeS3F18(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef erinfo As TCACT_ERR_INFO,
        ByRef msg_len As Integer
       ) As Integer
```

[C#]

```
int EncodeS3F18(
        IntPtr buffer,
        int buff_size,
        ref TCACT_ERR_INFO erinfo,
        ref int msg_len
       );
```

（2）引数

buffer	: S3F18 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
erinfo	: S3F18 の応答情報が保存されている構造体です。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに、erinfo に含まれる S3F18 応答情報をエンコードします。
作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

① C/C++

```
int    CAACK = 0;
int    ERR_CODE = 3;
char*  ERR_TEXT = "ERROR-100";

int    ei;

BYTE   buff[1000];
int            msg_len;
TCACT_INFO     info;
TCACT_ERR_INFO erinfo;

DshInitTCACT_ERR_INFO( &erinfo, CAACK, 1 );
DshPutTCACT_ERR_INFO( &erinfo, 0, ERR_CODE, ERR_TEXT );
ei = DSH_EncodeS3F18(buff, 1000, &erinfo, &msg_len );
.
.
DshFreeTCACT_ERR_INFO( &erinfo );
```

② C#

```
int    ERR_CODE = 3;
string ERR_TEXT = "ERROR-100";

int ei;
int msg_len = 0;
int ack = 1;

TCACT_ERR_INFO erinfo = new TCACT_ERR_INFO();

IntPtr buff = Marshal.AllocCoTaskMem(1000);
comm_lib.set_dshmsg(ref rmsg, 3, 18, 0, buff, 1000);

DshInitTCACT_ERR_INFO(ref erinfo, ack, 1);
DshPutTCACT_ERR_INFO(ref erinfo, 0, ERR_CODE, ERR_TEXT);

DSH_EncodeS3F18(buff, 1000, ref erinfo, ref msg_len);    // encode S3F17
.
.
Marshal.FreeCoTaskMem(buff);
DshFreeTCACT_ERR_INFO(ref erinfo);
```

３．２．19．４　DSH_DecodeS3F18 () －　受信した S3F18 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS3F18 (
        BYTE *buffer,
        int msg_len,
        TCACT_ERR_INFO *erinfo
        );
```

[VB.Net]

```
Function DSH_DecodeS3F18 (
        buffer As IntPtr,
        msg_len As Integer,
        ByRef erinfo As TCACT_ERR_INFO
        ) As Integer
```

[C#]

```
int DSH_DecodeS3F18 (
        IntPtr buffer,
        int msg_len,
        ref TCACT_ERR_INFO erinfo
        );
```

（2）引数

buffer	：	S3F18 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S3F18 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
erinfo	：	S3F18 の応答情報を保存する構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾒｯｾｰｼﾞ形式が正しくなかった。(ﾘｽﾄ構造の違い、ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞの違いなど)

（4）説明

buffer で指定されたバッファに格納されている S3F18 メッセージのデコードを行い、 得られた情報を erinfo 構造体にセットします。

正常にデコードできた場合は、0 を返却します。また、メッセージフォーマットが SEMI 仕様に合致しなかった場合は、(-1)が返却されます。

３．２．20　S3F23 メッセージ　－　ﾎﾟｰﾄｸﾞﾙｰﾌﾟｱｸｼｮﾝ要求情報の送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS3F23()	S3F23 をｴﾝｺｰﾄﾞします。	ﾎﾟｰﾄｸﾞﾙｰﾌﾟｱｸｼｮﾝ要求情報をｴﾝｺｰﾄﾞします。
2	DSH_DecodeS3F23()	S3F23 をﾃﾞｺｰﾄﾞします。	ﾎﾟｰﾄｸﾞﾙｰﾌﾟｱｸｼｮﾝ要求情報にﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS3F24()	S3F24 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	応答情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS3F24()	S3F24 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	応答情報を取得します。

（2）S3F23 のユーザインタフェース情報
情報の引き渡しは構造体 TPORTG_INFO を使って行います。

①ﾎﾟｰﾄｸﾞﾙｰﾌﾟｱｸｼｮﾝ要求とパラメータを保存する構造体

```
typedef struct{
        char        *portgrpaction;     // group action
        char        *portgrpname;       // port group name
        int         pn_count;           // parameter count
        TPORTG_PARA **pn_list;          // paramete list
}TPORTG_INFO;
```

②ﾎﾟｰﾄｸﾞﾙｰﾌﾟｱｸｼｮﾝ要求に含む１個のパラメータ情報を保存する構造体

```
typedef struct{
        char        *paramname;         // paramname
        int         pval_fmt;           // paramval  item fmt
        int         pval_size;          // paramval  data array size
        void        *paramval;          // paramval data
}TPORTG_PARA;
```

（3）TPORTG_INFO 構造体への情報設定処理関連関数
C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。
.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTPORTG_INFO	TPORTG_INFO を初期設定する。
2	DshPutTPORTG_INFO	TPORTG_INFO に１個のﾊﾟﾗﾒｰﾀを加える。
3	DshFreeTPORTG_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

（4）S3F24 のユーザインタフェース情報

応答情報を TCACT_ERR_INFO 構造体を使用します。

①S3F24 に設定する応答情報を格納するための構造体です。

```
typedef struct{
        int         caack;
        int         err_count;
        TERR_INFO   **err_list;
} TCACT_ERR_INFO;
```

②ｴﾗｰﾊﾟﾗﾒｰﾀ情報を１個分格納するための構造体です。（ｴﾗｰｺｰﾄﾞとｴﾗｰﾃｷｽﾄ）

```
typedef struct{
        int     errcode;
        char    *errtext;
} TERR_INFO;
```

（5）TCACT_ERR_INFO 構造体への情報設定処理関連関数

番号	関数名	機能
1	DshInit TCACT_ERR_INFO	TCACT_ERR_INFO を初期設定する。
2	DshPut TCACT_ERR_INFO	TCACT_ERR_INFO １個のﾊﾟﾗﾒｰﾀを加える。
3	DshFree TCACT_ERR_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

３．２．20．１　DSH_EncodeS3F23() －　S3F23 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS3F23(
        BYTE *buffer,
        int buff_size,
        TPORTG_INFO *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS3F23(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef info As TPORTG_INFO,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS3F23(
        IntPtr buffer,
        int buff_size,
        ref TPORTG_INFO info,
        ref int msg_len
       );
```

（2）引数

buffer	: S3F23 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	: ﾎﾟｰﾄｸﾞﾙｰﾌﾟｱｸｼｮﾝ要求情報を格納するための構造体です。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S3F23 メッセージを作成します。
info で指定された構造体TPORTG_INFO 内に含まれるポートグループアクション要求情報をS3F23 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
char*    ACTION    = "ReserveAtPort";
char*    PGROUP    = "Group-1";
int      PARA_COUNT = 1;
char*    PARA_NAME= "PARA1";
char*    PARA_VAL = "PVAL100";
int              ei;
BYTE     buff[1000];
int              msg_len;
TPORTG_INFO info;

DshInitTPORTG_INFO( &info, ACTION, PGROUP, PARA_COUNT );
DshPutTPORTG_INFO( &info, PARA_NAME, ICODE_A, strlen(PARA_VAL), PARA_VAL );

ei = DSH_EncodeS3F23( buff, SND_BUFF_SIZE, &info, &msg_len );            // encode
.
.
DshFreeTPORTG_INFO( &info );
```

②c#

```
  string ACTION = "ReserveAtPort";
  string PGROUP = "Group-1";

  int PARA_COUNT = 1;

  string PARA_NAME = "PARA1";
  string PARA_VAL = "PVAL100";

  int ei;
  int msg_len = 0;
  TPORTG_INFO info = new TPORTG_INFO();

  IntPtr buff = Marshal.AllocCoTaskMem(BUFF_SIZE);

  DshInitTPORTG_INFO(ref info, ACTION, PGROUP, PARA_COUNT);  //

  DshPutTPORTG_INFO(ref info, PARA_NAME, ICODE_A, DshStrLen(PARA_VAL), PARA_VAL);

  ei = DSH_EncodeS3F23(buff, BUFF_SIZE, ref info, ref msg_len);         // encode S3F23
  .
  .
  Marshal.FreeCoTaskMem(buff);
  DshFreeTPORTG_INFO(ref info);
```

３．２．20．２　DSH_DecodeS3F23() －　S3F23 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS3F23(
        BYTE *buffer,
        int msg_len,
        TPORTG_INFO *info
        );
```

[VB.Net]

```
Function DSH_DecodeS3F23(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TPORTG_INFO
       ) As Integer
```

[C#]

```
int DSH_DecodeS3F23(
        IntPtr buffer,
       int msg_len,
       ref TPORTG_INFO info
       );
```

（2）引数

buffer	：	S3F23 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S3F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
info	：	ﾎﾟｰﾄｸﾞﾙｰﾌﾟ ｱｸｼｮﾝ要求情報を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S3F23 メッセージのデコードを行います。
デコード結果のﾎﾟｰﾄｸﾞﾙｰﾌﾟ ｱｸｼｮﾝ要求情報は、info 構造体に格納されます。

（5）例

①ｃ、C++

```
BYTE buff[2000];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S3F23 受信)

int msg_len = 205;                  // 受信した S3F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TPORTG_INFO info;
int ei;

ei = DSH_DecodeS3F23( buff, msg_len, &info );
.
.
DshFreeTPORTG_INFO( &info );
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S3F23 受信)

int msg_len = 205;                  // 受信した S3F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TPORTG_INFO info = new TPORTG_INFO();

int ei = DSH_DecodeS3F23( buff, msg_len, ref info );
.
.
DshFreeTPORTG_INFO( ref info );
Marshal.FreeCoTaskMem(buff);
```

３．２．20．３　DSH_EncodeS3F24() －　S3F24 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX EncodeS3F24(
        BYTE *buffer,
        int buff_size,
        TCACT_ERR_INFO *erinfo,
        int *msg_len
        );
```

[VB.Net]

```
Function EncodeS3F24(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef erinfo As TCACT_ERR_INFO,
        ByRef msg_len As Integer
       ) As Integer
```

[C#]

```
int EncodeS3F24(
        IntPtr buffer,
        int buff_size,
        ref TCACT_ERR_INFO erinfo,
        ref int msg_len
       );
```

（2）引数

buffer	: S3F24 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
erinfo	: S3F24 の応答情報が保存されている構造体です。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに、erinfo に含まれる S3F24 応答情報をエンコードします。
作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

① C/C++

```
BYTE   CAACK = 0;
int    ERR_CODE = 3;
char*  ERR_TEXT = "ERROR-100";

int    ei;
BYTE   buff[1000];
int    msg_len;
TPORTG_INFO     info;
TCACT_ERR_INFO erinfo;

DshInitTCACT_ERR_INFO( &erinfo, CAACK, 1 );
DshPutTCACT_ERR_INFO( &erinfo, 0, ERR_CODE, ERR_TEXT );

ei = DSH_EncodeS3F24(buff, 1000, &erinfo, &msg_len );

DshFreeTCACT_ERR_INFO( &erinfo );
```

② C#

```
int CAACK = 0;
int CANAK = 1;
int ERR_CODE = 3;
string ERR_TEXT = "ERROR-100";

int ei;
int msg_len = 0;
int ack = CAACK;
TPORTG_INFO info = new TPORTG_INFO();

TCACT_ERR_INFO erinfo = new TCACT_ERR_INFO();

IntPtr buff = Marshal.AllocCoTaskMem(BUFF_SIZE);

DshInitTCACT_ERR_INFO(ref erinfo, ack, 1);
DshPutTCACT_ERR_INFO(ref erinfo, 0, ERR_CODE, ERR_TEXT);

DSH_EncodeS3F24(buff, 1000, ref erinfo, ref msg_len);    // encode S3F23
.
.
Marshal.FreeCoTaskMem(buff);
DshFreeTCACT_ERR_INFO(ref erinfo);
```

３．２．20．４　DSH_DecodeS3F24 () －　受信した S3F24 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS3F24 (
        BYTE *buffer,
        int msg_len,
        TCACT_ERR_INFO *erinfo
        );
```

[VB.Net]

```
Function DSH_DecodeS3F24 (
        buffer As IntPtr,
        msg_len As Integer,
        ByRef erinfo As TCACT_ERR_INFO
        ) As Integer
```

[C#]

```
int DSH_DecodeS3F24 (
        IntPtr buffer,
        int msg_len,
        ref TCACT_ERR_INFO erinfo
        );
```

（2）引数

buffer	：	S3F24 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S3F24 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
erinfo	：	S3F24 の応答情報を保存する構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾒｯｾｰｼﾞ形式が正しくなかった。(ﾘｽﾄ構造の違い、ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞの違いなど)

（4）説明

buffer で指定されたバッファに格納されている S3F24 メッセージのデコードを行い、 得られた情報を erinfo 構造体にセットします。

正常にデコードできた場合は、0 を返却します。また、メッセージフォーマットが SEMI 仕様に合致しなかった場合は、(-1)が返却されます。

３．２．21　S3F25 メッセージ　－　PORT ﾎﾟｰﾄｱｸｼｮﾝ要求情報の送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS3F25()	S3F25 をｴﾝｺｰﾄﾞします。	ﾎﾟｰﾄｱｸｼｮﾝ要求情報をｴﾝｺｰﾄﾞします。
2	DSH_DecodeS3F25()	S3F25 をﾃﾞｺｰﾄﾞします。	ﾎﾟｰﾄｱｸｼｮﾝ要求情報にﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS3F26()	S3F26 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	応答情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS3F26()	S3F26 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	応答情報を取得します。

（2）S3F25 のユーザインタフェース情報

情報の引き渡しは構造体 TPORT_INFO を使って行います。

①ﾎﾟｰﾄｱｸｼｮﾝ要求に含む１個のパラメータ情報を保存する構造体

```
typedef struct{
        char        *paramname;         // paramname
        int         pval_fmt;           // paramval  item fmt
        int         pval_size;          // paramval  data array size
        void        *paramval;          // paramval data
}TPORT_PARA;
```

②ﾎﾟｰﾄｱｸｼｮﾝ要求とパラメータを保存する構造体

```
    typedef struct{
            char        *portaction;        // port action
            int         ptn;
            int         pn_count;           // parameter count
            TPORT_PARA  **pn_list;          // paramete list
    }TPORT_INFO;
```

（3）TPORT_INFO 構造体への情報設定処理関連関数

C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。

.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTPORT_INFO	TPORT_INFO を初期設定する。
2	DshPutTPORT_INFO	TPORT_INFO に１個のﾊﾟﾗﾒｰﾀを加える。
3	DshFreeTPORT_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

（4）S3F26 のユーザインタフェース情報

応答情報を TCACT_ERR_INFO 構造体を使用します。

①ｴﾗｰﾊﾟﾗﾒｰﾀ情報を１個分格納するための構造体です。（ｴﾗｰｺｰﾄﾞとｴﾗｰﾃｷｽﾄ）

```
typedef struct{
        int     errcode;
        char    *errtext;
} TERR_INFO;
```

②S3F26 に設定する応答情報を格納するための構造体です。

```
typedef struct{
        int         caack;
        int         err_count;
        TERR_INFO   **err_list;
} TCACT_ERR_INFO;
```

（5）TCACT_ERR_INFO 構造体への情報設定処理関連関数

番号	関数名	機能
1	DshInit TCACT_ERR_INFO	TCACT_ERR_INFO を初期設定する。
2	DshPut TCACT_ERR_INFO	TCACT_ERR_INFO １個のﾊﾟﾗﾒｰﾀを加える。
3	DshFree TCACT_ERR_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

３．２．21．１　DSH_EncodeS3F25() －　S3F25 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS3F25(
        BYTE *buffer,
        int buff_size,
        TPORT_INFO *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS3F25(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef info As TPORT_INFO,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS3F25(
        IntPtr buffer,
        int buff_size,
        ref TPORT_INFO info,
        ref int msg_len
       );
```

（2）引数

buffer	： S3F25 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	： PORT ﾎﾟｰﾄｱｸｼｮﾝ要求情報を格納するための構造体です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S3F25 メッセージを作成します。
info で指定された構造体 TPORT_INFO 内に含まれる PORT ポートアクション要求情報を S3F25 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
char*   ACTION   = "ReserveAtPort";
int     PORTNO     =  1;
int     PARA_COUNT = 1;
char*   PARA_NAME= "PARA1";
char*   PARA_VAL = "PVAL100";
int             ei;
BYTE    buff[1000];
int     msg_len;
TPORT_INFO info;

DshInitTPORT_INFO( &info, ACTION, PORTNO, PARA_COUNT );
DshPutTPORT_INFO( &info, PARA_NAME, ICODE_A, strlen(PARA_VAL), PARA_VAL );

ei = DSH_EncodeS3F25( buff, SND_BUFF_SIZE, &info, &msg_len );          // encode
.
.
DshFreeTPORT_INFO( &info );
```

②c#

```
  string ACTION = "ReserveAtPort";
  int    PORTNO = 1;

  int PARA_COUNT = 1;

  string PARA_NAME = "PARA1";
  string PARA_VAL = "PVAL100";

  int ei;
  int msg_len = 0;
  TPORT_INFO info = new TPORT_INFO();

  IntPtr buff = Marshal.AllocCoTaskMem(BUFF_SIZE);

  DshInitTPORT_INFO(ref info, ACTION, PORTNO, PARA_COUNT);  //

  DshPutTPORT_INFO(ref info, PARA_NAME, ICODE_A, DshStrLen(PARA_VAL), PARA_VAL);

  ei = DSH_EncodeS3F25(buff, BUFF_SIZE, ref info, ref msg_len);        // encode S3F25
  .
  .
  Marshal.FreeCoTaskMem(buff);
  DshFreeTPORT_INFO(ref info);
```

３．２．21．２　DSH_DecodeS3F25() －　S3F25 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS3F25(
        BYTE *buffer,
        int msg_len,
        TPORT_INFO *info
        );
```

[VB.Net]

```
Function DSH_DecodeS3F25(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TPORT_INFO
       ) As Integer
```

[C#]

```
int DSH_DecodeS3F25(
        IntPtr buffer,
        int msg_len,
        ref TPORT_INFO info
        );
```

（2）引数

buffer	： S3F25 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S3F25 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
info	： ﾎﾟｰﾄｱｸｼｮﾝ要求情報を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S3F25 メッセージのデコードを行います。
デコード結果の PORT ポートアクション要求情報は、info 構造体に格納されます。

（5）例

①c、C++

```
BYTE buff[2100];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S3F25 受信)
int msg_len = 215;                // 受信した S3F25 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TPORT_INFO info;
int ei;

ei = DSH_DecodeS3F25( buff, msg_len, &info );
.
.
DshFreeTPORT_INFO( &info );
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2100);
(S3F25 受信)
int msg_len = 215;                // 受信した S3F25 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TPORT_INFO info = new TPORT_INFO();

int ei = DSH_DecodeS3F25( buff, msg_len, ref info );
.
.
DshFreeTPORT_INFO( ref info );
Marshal.FreeCoTaskMem(buff);
```

３．２．21．３　DSH_EncodeS3F26() －　S3F26 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX EncodeS3F26(
        BYTE *buffer,
        int buff_size,
        TCACT_ERR_INFO *erinfo,
        int *msg_len
        );
```

[VB.Net]

```
Function EncodeS3F26(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef erinfo As TCACT_ERR_INFO,
        ByRef msg_len As Integer
       ) As Integer
```

[C#]

```
int EncodeS3F26(
        IntPtr buffer,
        int buff_size,
        ref TCACT_ERR_INFO erinfo,
        ref int msg_len
       );
```

（2）引数

buffer	： S3F26 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
erinfo	： S3F26 の応答情報が保存されている構造体です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに、erinfo に含まれる S3F26 応答情報をエンコードします。
作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

① C/C++

```
BYTE   CAACK = 0;
int    ERR_CODE = 3;
char*  ERR_TEXT = "ERROR-100";

int    ei;
BYTE   buff[1000];
int    msg_len;
TPORT_INFO      info;
TCACT_ERR_INFO erinfo;

DshInitTCACT_ERR_INFO( &erinfo, CAACK, 1 );
DshPutTCACT_ERR_INFO( &erinfo, 0, ERR_CODE, ERR_TEXT );

ei = DSH_EncodeS3F26(buff, 1000, &erinfo, &msg_len );

DshFreeTCACT_ERR_INFO( &erinfo );
```

② C#

```
int CAACK = 0;
int CANAK = 1;
int ERR_CODE = 3;
string ERR_TEXT = "ERROR-100";

int ei;
int msg_len = 0;
int ack = CAACK;
TPORT_INFO info = new TPORT_INFO();

TCACT_ERR_INFO erinfo = new TCACT_ERR_INFO();

IntPtr buff = Marshal.AllocCoTaskMem(BUFF_SIZE);

DshInitTCACT_ERR_INFO(ref erinfo, ack, 1);
DshPutTCACT_ERR_INFO(ref erinfo, 0, ERR_CODE, ERR_TEXT);

DSH_EncodeS3F26(buff, 1000, ref erinfo, ref msg_len);    // encode S3F25
.
.
Marshal.FreeCoTaskMem(buff);
DshFreeTCACT_ERR_INFO(ref erinfo);
```

３．２．21．４　DSH_DecodeS3F26 () －　受信した S3F26 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS3F26 (
        BYTE *buffer,
        int msg_len,
        TCACT_ERR_INFO *erinfo
        );
```

[VB.Net]

```
Function DSH_DecodeS3F26 (
        buffer As IntPtr,
        msg_len As Integer,
        ByRef erinfo As TCACT_ERR_INFO
        ) As Integer
```

[C#]

```
int DSH_DecodeS3F26 (
        IntPtr buffer,
        int msg_len,
        ref TCACT_ERR_INFO erinfo
        );
```

（2）引数

buffer	：	S3F26 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S3F26 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
erinfo	：	S3F26 の応答情報を保存する構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾒｯｾｰｼﾞ形式が正しくなかった。(ﾘｽﾄ構造の違い、ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞの違いなど)

（4）説明

buffer で指定されたバッファに格納されている S3F26 メッセージのデコードを行い、 得られた情報を erinfo 構造体にセットします。

正常にデコードできた場合は、0 を返却します。また、メッセージフォーマットが SEMI 仕様に合致しなかった場合は、(-1)が返却されます。

３．２．22　S3F27 メッセージ　－　PORT ﾎﾟｰﾄｱｸｾｽ変更要求情報の送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS3F27()	S3F27 をｴﾝｺｰﾄﾞします。	ﾎﾟｰﾄｱｸｾｽ変更要求情報をｴﾝｺｰﾄﾞします。
2	DSH_DecodeS3F27()	S3F27 をﾃﾞｺｰﾄﾞします。	ﾎﾟｰﾄｱｸｾｽ変更要求情報にﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS3F28()	S3F28 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	応答情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS3F28()	S3F28 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	応答情報を取得します。

（2）S3F27 のユーザインタフェース情報

情報の引き渡しは構造体 TACCESS_INFO を使って行います。

①T ﾎﾟｰﾄｱｸｾｽ変更要求情報を保存する構造体

```
typedef struct{
        int         accessmode;         // access mode 0/1
        int         port_count;         // no. of port
        int         *port_list;         // port no. list
}TACCESS_INFO;
```

（3）TACCESS_INFO 構造体への情報設定処理関連関数

C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。

.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTACCESS_INFO	TACCESS_INFO を初期設定する。
2	DshPutTACCESS_INFO	TACCESS_INFO に 1 個のﾎﾟｰﾄを加える。
3	DshFreeTACCESS_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

（4）S3F28 のユーザインタフェース情報

応答情報を TACCESS_ERR_INFO 構造体を使用します。

①S3F28 情報を格納するための構造体です。（ﾎﾟｰﾄ、ｴﾗｰｺｰﾄﾞとｴﾗｰﾃｷｽﾄ）

```
typedef struct{
        int         port;               // port no.
        int         errcode;            // ok/ng - port
        char        *errtext;           // error text - port
}TACCESS_ERR_PORT;
```

（5）TACCESS_ERR_INFO 構造体への情報設定処理関連関数

番号	関数名	機能
1	DshInit TACCESS_ERR_INFO__Ext	TACCESS_ERR_INFO を初期設定する。
2	DshPut TACCESS_ERR_INFO_Ext	TACCESS_ERR_INFO 1 個のﾎﾟｰﾄを加える。
3	DshFree TACCESS_ERR_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

３．２．22．１　DSH_EncodeS3F27() －　S3F27 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS3F27(
        BYTE *buffer,
        int buff_size,
        TACCESS_INFO *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS3F27(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef info As TACCESS_INFO,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS3F27(
        IntPtr buffer,
        int buff_size,
        ref TACCESS_INFO info,
        ref int msg_len
       );
```

（2）引数

buffer	: S3F27 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	: ﾎﾟｰﾄｱｸｾｽ変更要求情報を格納するための構造体です。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S3F27 メッセージを作成します。
info で指定された構造体 TACCESS_INFO 内に含まれるポートアクセス変更要求情報を S3F27 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
int     MODE     = 1;
int     PTID_1   = 1;
int     PTID_2   = 2;
int     PORT_COUNT = 2;

int     ei;
BYTE    buff[1000];
int     msg_len;
TACCESS_INFO info;

DshInitTACCESS_INFO( &info, MODE, PORT_COUNT );
DshPutTACCESS_INFO( &info, PTID_1 );
DshPutTACCESS_INFO( &info, PTID_2 );

ei = DSH_EncodeS3F27( buff, SND_BUFF_SIZE, &info, &msg_len );          // encode
.
.
DshFreeTACCESS_INFO( &info );
```

②c#

```
  int MODE = 1;
  int PTID_1 = 1;
  int PTID_2 = 2;

  int PORT_COUNT = 2;

  int ei;
  int msg_len = 0;
  TACCESS_INFO info = new TACCESS_INFO();

  IntPtr buff = Marshal.AllocCoTaskMem(1000);

  DshInitTACCESS_INFO(ref info, MODE, PORT_COUNT);

  DshPutTACCESS_INFO(ref info, PTID_1);
  DshPutTACCESS_INFO(ref info, PTID_2);

  ei = DSH_EncodeS3F27(buff, 1000, ref info, ref msg_len); // encode S3F27
  .
  .
  Marshal.FreeCoTaskMem(buff);
  DshFreeTACCESS_INFO(ref info);
```

３．２．22．２　DSH_DecodeS3F27() －　S3F27 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS3F27(
        BYTE *buffer,
        int msg_len,
        TACCESS_INFO *info
        );
```

[VB.Net]

```
Function DSH_DecodeS3F27(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TACCESS_INFO
       ) As Integer
```

[C#]

```
int DSH_DecodeS3F27(
        IntPtr buffer,
       int msg_len,
       ref TACCESS_INFO info
       );
```

（2）引数

buffer	： S3F27 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S3F27 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
info	： ﾎﾟｰﾄｱｸｾｽ変更要求情報を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S3F27 メッセージのデコードを行います。
デコード結果の PORT ﾎﾟｰﾄｱｸｾｽ変更要求情報は、info 構造体に格納されます。

（5）例

①c、C++

```
BYTE buff[2000];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S3F27 受信)
int msg_len = 23;                   // 受信した S3F27 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TACCESS_INFO info;
int ei;

ei = DSH_DecodeS3F27( buff, msg_len, &info );
.
.
DshFreeTACCESS_INFO( &info );
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S3F27 受信)
int msg_len = 23;                   // 受信した S3F27 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TACCESS_INFO info = new TACCESS_INFO();

int ei = DSH_DecodeS3F27( buff, msg_len, ref info );
.
.
Marshal.FreeCoTaskMem(buff);
DshFreeTACCESS_INFO( ref info );
```

３．２．22．３　DSH_EncodeS3F28() －　S3F28 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX EncodeS3F28(
        BYTE *buffer,
        int buff_size,
        TACCESS_ERR_INFO *erinfo,
        int *msg_len
        );
```

[VB.Net]

```
Function EncodeS3F28(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef erinfo As TACCESS_ERR_INFO,
        ByRef msg_len As Integer
       ) As Integer
```

[C#]

```
int EncodeS3F28(
        IntPtr buffer,
        int buff_size,
        ref TACCESS_ERR_INFO erinfo,
        ref int msg_len
       );
```

（2）引数

buffer	： S3F28 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
erinfo	： S3F28 の応答情報が保存されている構造体です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに、erinfo に含まれる S3F28 応答情報をエンコードします。
作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

① C/C++

```
int    CAACK = 1;
int    ERR_CODE1 =       3;
char*  ERR_TEXT1 =       "ERROR-100";

int    ei;
DSHMSG *rmsg;
BYTE   buff[1000];
int    msg_len;

TACCESS_ERR_INFO erinfo;

DshInitTACCESS_ERR_INFO_Ext( &erinfo, CAACK, 1 );
DshPutTACCESS_ERR_INFO_Ext( &erinfo, PTID_1, ERR_CODE1, ERR_TEXT1 );

ei = DSH_EncodeS3F28(buff, 1000, &erinfo, &msg_len );
.
.
DshFreeTACCESS_ERR_INFO( &erinfo );
```

② C#

```
int CAACK = 0;
int CANAK = 1;
int ERR_CODE = 3;
string ERR_TEXT = "ERROR-100";

int ei;
int msg_len = 0;
int ack;
ack = CANAK;

TACCESS_ERR_INFO erinfo = new TACCESS_ERR_INFO();

IntPtr buff = Marshal.AllocCoTaskMem(1000);

DshInitTACCESS_ERR_INFO(ref erinfo, ref info,  ack, 1);
DshPutTACCESS_ERR_INFO_Ext(ref erinfo, PTID_1, ERR_CODE, ERR_TEXT);

DSH_EncodeS3F28(buff, 1000, ref erinfo, ref msg_len);    // encode S3F27
.
.
Marshal.FreeCoTaskMem(buff);
DshFreeTACCESS_ERR_INFO(ref erinfo);
```

３．２．22．４　DSH_DecodeS3F28 () －　受信した S3F28 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS3F28 (
        BYTE *buffer,
        int msg_len,
        TACCESS_ERR_INFO *erinfo
        );
```

[VB.Net]

```
Function DSH_DecodeS3F28 (
        buffer As IntPtr,
        msg_len As Integer,
        ByRef erinfo As TACCESS_ERR_INFO
        ) As Integer
```

[C#]

```
int DSH_DecodeS3F28 (
        IntPtr buffer,
        int msg_len,
        ref TACCESS_ERR_INFO erinfo
        );
```

（2）引数

buffer	：	S3F28 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S3F28 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
erinfo	：	S3F28 の応答情報を保存する構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾒｯｾｰｼﾞ形式が正しくなかった。(ﾘｽﾄ構造の違い、ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞの違いなど)

（4）説明

buffer で指定されたバッファに格納されている S3F28 メッセージのデコードを行い、 得られた情報を erinfo 構造体にセットします。

正常にデコードできた場合は、０を返却します。また、メッセージフォーマットが SEMI 仕様に合致しなかった場合は、(-1)が返却されます。

３．２．23　S5F1 メッセージ　－　アラーム情報通知

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS5F1()	S5F1 をｴﾝｺｰﾄﾞします。	ｱﾗｰﾑ通知情報をｴﾝｺｰﾄﾞします。
2	DSH_DecodeS5F1()	S5F1 をﾃﾞｺｰﾄﾞします。	ｱﾗｰﾑ通知情報にﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS5F2()	S5F2 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	ack をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS5F2()	S5F2 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	ack を取得します。

（2）S5F1 のユーザインタフェース情報

情報の引き渡しは構造体 TAL_S5F1_INFO を使って行います。

①ｱﾗｰﾑ通知情報を保存する構造体

```
typedef struct{
        int         on_off;
        TALID       alid;
        TALCD       alcd;
        char        *altx;
} TAL_S5F1_INFO;
```

（3）TAL_S5F1_INFO 構造体への変数値の設定処理関連関数

C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。

.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshMakeTAL_S5F1_INFO	TAL_S5F1_INFO にｱﾗｰﾑ通知情報を設定する。
2	DshFreeTAL_S5F1_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

（4）S5F2 のユーザインタフェース

ACK を関数の引数を使って受け渡しします。

３．２．23．１　DSH_EncodeS5F1() －　S5F1 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS5F1(
        BYTE *buffer,
        int buff_size,
        TAL_S5F1_INFO *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS5F1(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef info As TAL_S5F1_INFO,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS5F1(
        IntPtr buffer,
        int buff_size,
        ref TAL_S5F1_INFO info,
        ref int msg_len
       );
```

（2）引数

buffer	: S5F1 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	: ｱﾗｰﾑ通知情報を格納するための構造体です。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S5F1 メッセージを作成します。
info で指定された構造体 TAL_S5F1_INFO 内に含まれるアラーム情報を S5F1 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
int    ALCD     = 2;
int    ON_OFF   = 1;
char* ALTX     = "Chamber Pressure-1 Over                    ";

BYTE   buff[100];
int    msg_len;
int    ei;
TAL_S5F1_INFO info;

DshMakeTAL_S5F1_INFO( &info, AL_AlarmPressure_1_Low, ALCD, ALTX, ON_OFF );

ei = DSH_EncodeS5F1( buff, 100, &info, &msg_len );
.
.
DshFreeTAL_S5F1_INFO( &info );
```

②c#

```
int    ALCD     = 2;
int    ON_OFF   = 1;
string ALTX     = "Chamber Pressure-1 Over                    ";

int ei;
int msg_len = 0;
TAL_S5F1_INFO info = new INFO.TAL_S5F1_INFO();
IntPtr buff = Marshal.AllocCoTaskMem(100);

DshMakeTAL_S5F1_INFO( ref info, AL_AlarmPressure_1_Low, ALCD, ALTX, ON_OFF );

ei = DSH_EncodeS5F1(buff, 100, ref info, ref msg_len);     // encode S5F1.
.
.
DshFreeTAL_S5F1_INFO( ref info );
Marshal.FreeCoTaskMem(buff);
```

３．２．23．２　DSH_DecodeS5F1() －　S5F1 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS5F1(
        BYTE *buffer,
        int msg_len,
        TAL_S5F1_INFO *info
        );
```

[VB.Net]

```
Function DSH_DecodeS5F1(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TAL_S5F1_INFO
       ) As Integer
```

[C#]

```
int DSH_DecodeS5F1(
        IntPtr buffer,
        int msg_len,
        ref TAL_S5F1_INFO info
        );
```

（2）引数

buffer	：	S5F1 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S5F1 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
info	：	ｱﾗｰﾑ通知情報を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S5F1 メッセージのデコードを行います。
デコード結果は、info に格納されます。

（5）例

①C/C++

```
BYTE buff[2000];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S5F1 受信)
int msg_len = 53;                  // 受信した S5F1 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TAL_S5F1_INFO info;
int ei;
ei = DSH_DecodeS5F1( buff, msg_len, &info );
.
.
DshFreeTAL_S5F1_INFO( &info );
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S5F1 受信)

int msg_len = 53;                  // 受信した S5F1 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TAL_S5F1_INFO info = new TAL_S5F1_INFO();

int ei = DSH_DecodeS5F1( buff, msg_len, 53, ref info );
.
.
DshFreeTAL_S5F1_INFO( ref info );
Marshal.FreeCoTaskMem(buff);
```

３．２．23．３　DSH_EncodeS5F2() －　S5F2 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX EncodeS5F2(
        BYTE *buffer,
        int buff_size,
        int ack,
        int *msg_len
        );
```

[VB.Net]

```
Function EncodeS5F2(
        buffer As IntPtr,
        buff_size As Integer,
        ack As Integer,
        ByRef msg_len As Integer
       ) As Integer
```

[C#]

```
int EncodeS5F2(
        IntPtr buffer,
        int buff_size,
        int ack,
        ref int msg_len
       );
```

（2）引数

buffer	： S5F2 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ack	： S5F2 の ACK です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに ack を含めて S5F2 メッセージを作成します。
作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

３．２．23．４　DSH_DecodeS5F2 () －　受信した S5F2 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS5F2 (
        BYTE *buffer,
        int msg_len,
        int *ack
        );
```

[VB.Net]

```
Function DSH_DecodeS5F2 (
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ack As Integer
        ) As Integer
```

[C#]

```
int DSH_DecodeS5F2 (
        IntPtr buffer,
        int msg_len,
        ref int ack
       );
```

（2）引数

buffer	: S5F2 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	: S5F2 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ack	: S5F2 の ACK 格納用です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾒｯｾｰｼﾞ形式が正しくなかった。(ﾘｽﾄ構造の違い、ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞの違いなど)

（4）説明

buffer で指定されたバッファに格納されている S5F2 メッセージのデコードを行い、 ACK の値を ack に返却します。

正常にデコードできた場合は、0 を返却します。また、メッセージフォーマットが SEMI 仕様に合致しなかった場合は、(-1)が返却されます。

３．２．24　S5F3 メッセージ　－　アラーム報告/無効の送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS5F3()	S5F3 をｴﾝｺｰﾄﾞします。	ｱﾗｰﾑ報告/無効をｴﾝｺｰﾄﾞします。
2	DSH_DecodeS5F3()	S5F3 をﾃﾞｺｰﾄﾞします。	ｱﾗｰﾑ報告/無効をﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS5F4()	S5F4 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	ack をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS5F4()	S5F4 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	ack を取得します。

（2）S5F3 のユーザインタフェース情報
アラーム ID,　有効／無効フラグ、全 ID を対象にするかどうかのフラグがあります。
これらを関数の引数を使って行います。

（3）S5F2 のユーザインタフェース
ACK を関数の引数を使って受け渡しします。

３．２．24．１　DSH_EncodeS5F3() －　S5F3 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS5F3(
        BYTE *buffer,
        int buff_size,
        TALID  alid,
        int aled,
        int all_flag,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS5F3(
        buffer As IntPtr,
        buff_size As Integer,
        alid As Integer,
        aled Ad Integer,
        all_flag As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS5F3(
        IntPtr buffer,
        int buff_size,
        uint alid,
        int aled,
        int all_flag,
        ref int msg_len
        );
```

（2）引数

buffer	： S5F3 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
alid	： ｱﾗｰﾑ ID です。(all_flag=1 の場合は、値は使用されません。)
aled	： ALED(有効/無効ﾌﾗｸﾞ)です。 0=無効、128(0x80)=有効
all_flag	： 全ｱﾗｰﾑ ID 指定かどうかのﾌﾗｸﾞ。0=1 個だけ指定、 1=全ｱﾗｰﾑ ID を指定する。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S5F3 メッセージを作成します。
alid, aled, all_flag で指定された引数を S5F3 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、0 を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

（5）例

①C/C++

```
uint   AL_AlarmPressure_1_Low   = 101;
int    ON_OFF    = 0x80;                    // 有効
int    ALL_FLAG = 0;

BYTE   buff[100];
int    msg_len;
int    ei;
TAL_S5F3_INFO info;

ei = DSH_EncodeS5F3( buff, 100, AL_AlarmPressure_1_Low, ON_OFF, ALL_FLAG, &msg_len );.
.
.
```

②c#

```
uint   AL_AlarmPressure_1_Low   = 101;
int    ON_OFF    = 0x80;
int    ALL_FLAG = 0;

int ei;
int msg_len = 0;
IntPtr buff = Marshal.AllocCoTaskMem(100);

DshMakeTAL_S5F3_INFO( ref info, AL_AlarmPressure_1_Low, ALCD, ALTX, ON_OFF );

ei = DSH_EncodeS5F3(buff, 100, AL_AlarmPressure_1_Low, ON_OFF, ALL_FLAG, ref msg_len);     .
.
Marshal.FreeCoTaskMem(buff);
```

３．２．24．２　DSH_DecodeS5F3() －　S5F3 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS5F3(
        BYTE *buffer,
        int msg_len,
        TALID  *alid,
        int *aled,
        int *all_flag
        );
```

[VB.Net]

```
Function DSH_DecodeS5F3(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef  alid As Integer,
        ByRef  aled As Integer,
        ByRef  all_flag As Integer
       ) As Integer
```

[C#]

```
int DSH_DecodeS5F3(
        IntPtr buffer,
        int msg_len,
        ref uint  alid,
        ref int aled,
        ref int all_flag
       );
```

（2）引数

buffer	： S5F3 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S5F3 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
alid	： ｱﾗｰﾑ ID 格納用です。(all_flag=1 の場合は、値は使用されません。)
aled	： ALED(有効/無効ﾌｧﾌｸﾞ)格納用です。 0=無効、128=有効
all_flag	： 全ｱﾗｰﾑ ID 指定かどうかのﾌﾗｸﾞ格納用です。0=1 個だけ指定、1=全ｱﾗｰﾑ ID。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S5F3 メッセージのデコードを行います。
デコード結果は、info に格納されます。

（5）例

①c、C++

```
BYTE buff[100];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S5F3 受信)
int msg_len = 13;                  // 受信した S5F3 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TALID alid;
int   aled;
int   all_flag;

int ei;
ei = DSH_DecodeS5F3( buff, msg_len, &alid, &aled, &all_flag );
.
.
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(100);
(S5F3 受信)
int msg_len = 13;                  // 受信した S5F3 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

uint  alid = 0;
int   aled = 0;
int   all_flag = 0;

int ei = DSH_DecodeS5F3( buff, msg_len, ref alid, ref aled, ref all_flag );
.
.
```

３．２．24．３　DSH_EncodeS5F4() －　S5F4 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX EncodeS5F4(
        BYTE *buffer,
        int buff_size,
        int ack,
        int *msg_len
        );
```

[VB.Net]

```
Function EncodeS5F4(
        buffer As IntPtr,
        buff_size As Integer,
        ack As Integer,
        ByRef msg_len As Integer
       ) As Integer
```

[C#]

```
int EncodeS5F4(
        IntPtr buffer,
        int buff_size,
       int ack,
       ref int msg_len
       );
```

（2）引数

buffer	: S5F4 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ack	: S5F4 の ACK です。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに ack を含めて S5F4 メッセージを作成します。
作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

３．２．24．４　DSH_DecodeS5F4 () －　受信した S5F4 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS5F4 (
        BYTE *buffer,
        int msg_len,
        int *ack
        );
```

[VB.Net]

```
Function DSH_DecodeS5F4 (
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ack As Integer
        ) As Integer
```

[C#]

```
int DSH_DecodeS5F4 (
        IntPtr buffer,
        int msg_len,
        ref int ack
        );
```

（2）引数

buffer	： S5F4 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S5F4 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ack	： S5F4 の ACK 格納用です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾒｯｾｰｼﾞ形式が正しくなかった。(ﾘｽﾄ構造の違い、ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞの違いなど)

（4）説明

buffer で指定されたバッファに格納されている S5F4 メッセージのデコードを行い、 ACK の値を ack に返却します。

正常にデコードできた場合は、0 を返却します。また、メッセージフォーマットが SEMI 仕様に合致しなかった場合は、(-1)が返却されます。

３．２．25　S5F5 メッセージ　－　アラーム（装置状態変数）変数名リストの要求

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS5F5()	S5F5 をｴﾝｺｰﾄﾞします。	ｱﾗｰﾑ情報を求めたい ALID をｴﾝｺｰﾄﾞします。
2	DSH_DecodeS5F5()	S5F5 をﾃﾞｺｰﾄﾞします。	ﾃﾞｺｰﾄﾞし、ALID を取得します。 ALID 配列領域に取得します。
3	DSH_EncodeS5F6()	S5F6 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	ｱﾗｰﾑﾘｽﾄを S5F6 にｴﾝｺｰﾄﾞします。
4	DSH_DecodeS5F6()	S5F6 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	ﾃﾞｺｰﾄﾞし、ｱﾗｰﾑﾘｽﾄを取得します。 TAL_S5F6_LIST 構造体に格納します。

（2）S5F5 のユーザインタフェース情報

情報の引き渡しは、符号なし３２ビット整数の配列と ALID 数になります。

uint alid_list[] と alid count　　(uint は Unsigned 32 bits Integer の意味です。)

（3）S5F6 のユーザインタフェース情報

情報の引き渡しは TAL_S5F6_LIST を使って行います。

①アラーム情報１個分を保存する構造体

```
typedef struct{
        int         on_off;
        TALID       alid;
        TALCD       alcd;
        char        *altx;
} TAL_S5F1_INFO;
```

②複数のアラーム情報を保存する構造体

```
    typedef struct{
            int         count;
            TAL_S5F1_INFO    **al_list;
    } TAL_S5F6_LIST;
```

（4）TAL_S5F6_LIST 構造体へのアラーム情報の設定処理関連関数

C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。

.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTAL_S5F6_LIST	TAL_S5F6_LIST を初期設定する。
2	DshPutTAL_S5F6_LIST	１個のｱﾗｰﾑ情報をﾘｽﾄに加える。
3	DshMakeTAL_S5F1_INFO	１個のｱﾗｰﾑ情報を作成する。(TAL_S5F1_INFO 使用)
4	DshFreeTAL_S5F6_LIST	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

３．２．25．１　DSH_EncodeS5F5() －　S5F5 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS5F5(
        BYTE *buffer,
        int buff_size,
        TALID *alid_list,
        int  count,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS5F5(
        buffer As IntPtr,
        buff_size As Integer,
        alid_list As UInteger(),
        count As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS5F5(
        IntPtr buffer,
        int buff_size,
        uint[] alid_list,
        int count,
        ref int msg_len
       );
```

（2）引数

buffer	：S5F5 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	：buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
alid_list	：ALID が格納されている配列ﾘｽﾄです。
count	：alid_list 配列に格納されている ALID の数です。
msg_len	：ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S5F5 メッセージを作成します。
alid_list 配列に count 分の ALID をエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、0 を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

（5）例

①C/C++

```
TALID AL_AlarmTempOver         1
TALID AL_AlarmPressure_1_Low   101
TALID AL_AlarmPressure_2_Low   102
TALID AL_COUNT  3
TALID alid_list[AL_COUNT] = {
      AL_AlarmTempOver, AL_AlarmPressure_1_Low, AL_AlarmPressure_2_Low          // 3個
      };
BYTE buffer[64];
int ei;
int msg_len;

ei = DSH_EncodeS5F5( buff, 64, alid_list, AL_COUNT, &msg_len );
.
.
```

②C#

```
uint   AL_AlarmTempOver         1
uint   AL_AlarmPressure_1_Low   101
uint   AL_AlarmPressure_2_Low   102
uint   AL_COUNT  3
uint[] alid_lis] = {
      AL_AlarmTempOver, AL_AlarmPressure_1_Low, AL_AlarmPressure_2_Low          // 3個
      };

IntPtr buff = Marshal. AllocCoTaskMem(64);

int  msg_len = 0;
int ei = DSH_EncodeS5F5( buff, 64, alid_list, AL_COUNT, ref msg_len );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．25．２　DSH_DecodeS5F5() －　S5F5 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS5F5(
        BYTE *buffer,
        int msg_len,
        TALID alid_list[],
        int max_count,
        int *vount
        );
```

[VB.Net]

```
Function DSH_DecodeS5F5(
        buffer As IntPtr,
        msg_len As Integer,
        alid_list As UInteger(),
        max_count As UInteger,
        ByRef count As Integer,
       ) As Integer
```

[C#]

```
int DSH_DecodeS5F5(
        IntPtr buffer,
        int msg_len,
        uint[] alid_list,
        int max_count,
        ref int count
        );
```

（2）引数

buffer	: S5F5 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	: S5F5 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
alid_list	: ALID を格納する配列です。
max_count	: alid_list に収納できる最大個数
count	: ﾃﾞｺｰﾄﾞで得られた ALID の数

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S5F5 メッセージのデコードを行います。
デコード結果は、alid_list に格納し、デコードで得られた ALID 数は、count に格納します。

（5）例

①c、C++

```
BYTE buff[1000];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S5F5 受信)
int msg_len = 32;                   // 受信した S5F5 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

uint alid_list[64];
int count;
int ei;
ei = DSH_DecodeS5F5( buff, msg_len, 64, &count );
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(1000);
(S5F5 受信)
int msg_len = 24;                   // 受信した S5F5 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

uint[] alid_list = new uint[64];
int count = 0;
int ei = DSH_DecodeS5F5( buff, msg_len, 64, ref count );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．25．３　DSH_EncodeS5F6() －　S5F6 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS5F6(
        BYTE *buffer,
        int buff_size,
        TAL_S5F6_LIST *list,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS5F6(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef list As TAL_S5F6_LIST,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS5F6(
        IntPtr buffer,
        int buff_size,
        ref TAL_S5F6_LIST list,
        ref int msg_len
       );
```

（2）引数

buffer　　：S5F6 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size　　：buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
list　　：ｱﾗｰﾑ情報を格納用構造体です。
msg_len　　：ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。
(Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S5F6 メッセージを作成します。
list で指定された構造体 TAL_S5F6_LIST 内に含まれるアラーム情報を S5F6 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
TALID AL_AlarmTempOver         1
TALID AL_AlarmPressure_1_Low   101
TALID AL_AlarmPressure_2_Low   102

int             ALCD1 = 2;
int             ALCD2 = 2;
int             ALCD3 = 2;

char* ALTX1 = "Chamber Temperature Over";
char* ALTX2 = "Chamber Pressure-1 Over";
char* ALTX3 = "Chamber Pressure-2 Over";

TAL_S5F6_LIST list;
TAL_S5F1_INFO info;

DshInitTAL_S5F6_LIST( &list, 3 );
DshMakeTAL_S5F1_INFO( &info, AL_AlarmTempOver, ALCD1, ALTX1, 1 );
DshPutTAL_S5F6_LIST( &list, &info );
DshFreeTAL_S5F1_INFO( &info );

DshMakeTAL_S5F1_INFO( &info, AL_AlarmPressure_1_Low, ALCD2, ALTX2, 1 );
DshPutTAL_S5F6_LIST( &list, &info );
DshFreeTAL_S5F1_INFO( &info );

DshMakeTAL_S5F1_INFO( &info, AL_AlarmPressure_2_Low, ALCD3, ALTX3, 1 );
DshPutTAL_S5F6_LIST( &list, &info );
DshFreeTAL_S5F1_INFO( &info );

ei = DSH_EncodeS5F6(buff, RSP_BUFF_SIZE, &list, &msg_len );
DshFreeTAL_S5F6_LIST( &list );
.
```

②c#

```
uint[] alid_list = {                        // ALID list
       AL_AlarmTempOver,
       AL_AlarmPressure_1_Low,
       AL_AlarmPressure_2_Low
      };

int ALCD_AL_AlarmTempOver = 2;
int ALCD_AL_AlarmPressure_1_Low = 2;
int ALCD_AL_AlarmPressure_2_Low = 3;

string ALTX_AL_AlarmTempOver = "Chamber Temperature Over";
string ALTX_AL_AlarmPressure_1_Low = "Chamber Pressure-1 Over";
string ALTX_AL_AlarmPressure_2_Low = "Chamber Pressure-2 Over";
```

```
TAL_S5F6_LIST list = new TAL_S5F6_LIST();
TAL_S5F1_INFO info = new TAL_S5F1_INFO();

IntPtr buff = Marshal.AllocCoTaskMem(1000);
DshInitTAL_S5F6_LIST( ref list, 3 );

DshMakeTAL_S5F1_INFO( ref info,
      AL_AlarmTempOver, ALCD_AL_AlarmTempOver, ALTX_AL_AlarmTempOver, 1 );
DshPutTAL_S5F6_LIST( ref list, ref info );
DshFreeTAL_S5F1_INFO( ref info );

DshMakeTAL_S5F1_INFO( ref info,
      AL_AlarmPressure_1_Low, ALCD_AL_AlarmPressure_1_Low, ALTX_AL_AlarmPressure_1_Low, 1 );
DshPutTAL_S5F6_LIST( ref list, ref info );
DshFreeTAL_S5F1_INFO( ref info );

DshMakeTAL_S5F1_INFO( ref info,
      AL_AlarmPressure_2_Low, ALCD_AL_AlarmPressure_2_Low, ALTX_AL_AlarmPressure_2_Low, 1 );
DshPutTAL_S5F6_LIST( ref list, ref info );
DshFreeTAL_S5F1_INFO( ref info );

ei = DSH_EncodeS5F6(buff, 1000, ref list, ref msg_len);
.
.
DshFreeTAL_S5F6_LIST(ref list);
Marshal.FreeCoTaskMem(buff);
```

３．２．25．４　DSH_DecodeS5F6() －　S5F6 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS5F6(
        BYTE *buffer,
        int msg_len,
        TAL_S5F6_LIST *list
        );
```

[VB.Net]

```
Function DSH_DecodeS5F6(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef list As TAL_S5F6_LIST
       ) As Integer
```

[C#]

```
int DSH_DecodeS5F6(
        IntPtr buffer,
        int msg_len,
        ref TAL_S5F6_LIST list
        );
```

（2）引数

buffer　　：S5F6 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。

msg_len　　：S5F6 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。
(Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

list　：アラームの名前情報を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S5F6 メッセージのデコードを行います。
デコード結果は、list に格納されます。

（5）例

①ｃ、C++

```
BYTE buff[2000];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S5F6 受信)
int msg_len = 161;                // 受信した S5F6 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TAL_S5F6_LIST list;
int ei;
ei = DSH_DecodeS5F6( buff, msg_len, &list );
.
.
DshFreeTAL_S5F6_LIST(ref list);
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S5F6 受信)
int msg_len = 161;                // 受信した S5F6 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TAL_S5F6_LIST list = new TAL_S5F6_LIST();

int ei = DSH_DecodeS5F6( buff, msg_len, ref list );
.
.
DshFreeTAL_S5F6_LIST(ref list);
Marshal.FreeCoTaskMem(buff);
```

３．２．26　S6F1 メッセージ　－　トレースデータの送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS6F1()	S6F1 をｴﾝｺｰﾄﾞします。	ﾄﾚｰｽﾃﾞｰﾀをｴﾝｺｰﾄﾞします。
2	DSH_DecodeS6F1()	S6F1 をﾃﾞｺｰﾄﾞします。	ﾄﾚｰｽﾃﾞｰﾀをﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS6F2()	S6F2 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	ack をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS6F2()	S6F2 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	ack を取得します。

（2）S6F1 のユーザインタフェース情報

情報の引き渡しは構造体 TTRACE_DATA を使って行います。

①EC を１個分を保存する構造体

```
typedef struct{
        int         format;             // data item code
        int         asize;              // array size
        void        *sv;                // status data value
} TTRACE_SV;
```

②複数の変数値を保存する構造体

```
typedef struct{
        void        *trid;              // trace id
        int         format;             // trace id format
        int         asize;              // trace id array size
        int         smpln;              // sampling no.
        char        *stime;             // start time
        int         count;              // no. of data
        TTRACE_SV   **sv_list;          // sv ptr list
} TTRACE_DATA;
```

（3）TTRACE_DATA 構造体への変数値の設定処理関連関数

C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。

.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTTRACE_DATA	TTRACE_DATA を初期設定する。
2	DshPutTTRACE_DATA	１個の SV のﾄﾚｰｽﾃﾞｰﾀを構造体に加える。
3	DshMakeTTRACE_SV	TTRACE_SV に１個のﾃﾞｰﾀを作成する。
4	DshFreeTTRACE_DATA	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

３．２．26．１　DSH_EncodeS6F1() －　S6F1 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS6F1(
        BYTE *buffer,
        int buff_size,
        TTRACE_DATA *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS6F1(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef info As TTRACE_DATA,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS6F1(
        IntPtr buffer,
        int buff_size,
        ref TTRACE_DATA info,
        ref int msg_len
       );
```

（2）引数

buffer	：S6F1 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	：buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	：ﾄﾚｰｽﾃﾞｰﾀ情報を格納されている構造体です。
msg_len	：ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S6F1 メッセージを作成します。
info で指定された構造体TTRACE_DATA 内に含まれるﾄﾚｰｽﾃﾞｰﾀ情報をS6F1 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
uint   SV_ControlState        = 8194                // U2
uint   SV_LoadCarAccessStatus = 8205                // U1

char*  TRID      = "TRID001";
USHORT SMPLIN    = 1;
char*  STIME     = "2013071013450412";
int              SV_COUNT = 2;
USHORT SV_VAL_1           = 1;
BYTE   SV_VAL_2           = 5;

int    ei;
BYTE   buff[1000];
int    msg_len;
TTRACE_DATA     info;
TTRACE_SV       sinfo;

DshInitTTRACE_DATA( &info, TRID, SMPLIN, STIME, SV_COUNT );

DshMakeTTRACE_SV( &sinfo, ICODE_U2, 1, &SV_VAL_1 );
DshPutTTRACE_DATA( &info, &sinfo );
DshFreeTTRACE_SV( &sinfo );

DshMakeTTRACE_SV( &sinfo, ICODE_U1, 1, &SV_VAL_2 );
DshPutTTRACE_DATA( &info, &sinfo );
DshFreeTTRACE_SV( &sinfo );

ei = DSH_EncodeS6F1( buff, 1000, &info, &msg_len );
.
.
DshFreeTTRACE_DATA( &info );
```

②c#

```
uint   SV_ControlState        = 8194                // U2
uint   SV_LoadCarAccessStatus = 8205                // U1

int SV_VAL_1 = 1;
int SV_VAL_2 = 5;

int ei;
int msg_len = 0;
IntPtr int_ptr = Marshal.AllocCoTaskMem(256);

IntPtr buff = Marshal.AllocCoTaskMem(1000);
TTRACE_DATA info = new TTRACE_DATA();
TTRACE_SV sinfo = new TTRACE_SV();
```

```
DshInitTTRACE_DATA(ref info, TRID, SMPLIN, STIME, SV_COUNT);

copy_int_to_ptr(int_ptr, SV_VAL_1);
DshMakeTTRACE_SV( ref sinfo, ICODE_U2, 1, int_ptr);
DshPutTTRACE_DATA(ref info, ref sinfo);
DshFreeTTRACE_SV( ref sinfo);

copy_int_to_ptr(int_ptr, SV_VAL_2);
DshMakeTTRACE_SV(ref sinfo, ICODE_U1, 1, int_ptr);
DshPutTTRACE_DATA(ref info, ref sinfo);
DshFreeTTRACE_SV(ref sinfo);

ei = DSH_EncodeS6F1(buff, 1000, ref info, ref msg_len);

DshFreeTTRACE_DATA(ref info);
Marshal.FreeCoTaskMem(buff);
Marshal.FreeCoTaskMem(int_ptr);
.
.


IntPtr copy_int_to_ptr(IntPtr ptr, int d)
{
    int[] d_list = new int[1];
    d_list[0] = d;
    Marshal.Copy(d_list, 0, ptr, 1);
    return ptr;
}
```

３．２．26．２　DSH_DecodeS6F1() －　S6F1 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS6F1(
        BYTE *buffer,
        int msg_len,
        TTRACE_DATA *info
        );
```

[VB.Net]

```
Function DSH_DecodeS6F1(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TTRACE_DATA
       ) As Integer
```

[C#]

```
int DSH_DecodeS6F1(
        IntPtr buffer,
        int msg_len,
        ref TTRACE_DATA info
        );
```

（2）引数

buffer	：	S6F1 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S6F1 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
info	：	変数ﾃﾞｰﾀ値を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S6F1 メッセージのデコードを行います。
デコード結果は、info に格納されます。

（5）例

①ｃ、C++

```
BYTE buff[2000];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S6F1 受信)
int msg_len = 55;                  // 受信した S6F1 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TTRACE_DATA info;
int ei;
ei = DSH_DecodeS6F1( buff, msg_len, &info );
.
.
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S6F1 受信)
int msg_len = 55;                  // 受信した S6F1 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TTRACE_DATA info = new TTRACE_DATA();
int ei = DSH_DecodeS6F1( buff, msg_len, 64, ref info );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．26．３　DSH_EncodeS6F2() －　S6F2 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX EncodeS6F2(
        BYTE *buffer,
        int buff_size,
        int ack,
        int *msg_len
        );
```

[VB.Net]

```
Function EncodeS6F2(
        buffer As IntPtr,
        buff_size As Integer,
        ack As Integer,
        ByRef msg_len As Integer
       ) As Integer
```

[C#]

```
int EncodeS6F2(
        IntPtr buffer,
        int buff_size,
        int ack,
        ref int msg_len
       );
```

（2）引数

buffer	： S6F2 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ack	： S6F2 の ACK です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに ack を含めて S6F2 メッセージを作成します。
作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

３．２．26．４　DSH_DecodeS6F2 () －　受信した S6F2 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS6F2 (
        BYTE *buffer,
        int msg_len,
        int *ack
        );
```

[VB.Net]

```
Function DSH_DecodeS6F2 (
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ack As Integer
        ) As Integer
```

[C#]

```
int DSH_DecodeS6F2 (
        IntPtr buffer,
        int msg_len,
        ref int ack
        );
```

（2）引数

buffer	: S6F2 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	: S6F2 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ack	: S6F2 の ACK 格納用です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾒｯｾｰｼﾞ形式が正しくなかった。(ﾘｽﾄ構造の違い、ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞの違いなど)

（4）説明

buffer で指定されたバッファに格納されている S6F2 メッセージのデコードを行い、 ACK の値を ack に返却します。

正常にデコードできた場合は、0 を返却します。また、メッセージフォーマットが SEMI 仕様に合致しなかった場合は、(-1)が返却されます。

３．２．27　S6F11 メッセージ　－　イベントレポートの送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS6F11()	S6F11 をｴﾝｺｰﾄﾞします。	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄをｴﾝｺｰﾄﾞします。
2	DSH_DecodeS6F11()	S6F11 をﾃﾞｺｰﾄﾞします。	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄをﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS6F12()	S6F12 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	ack をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS6F12()	S6F12 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	ack を取得します。

（2）S6F11 のユーザインタフェース情報
情報の引き渡しは構造体 TS6F11_CE_INFO を使って行います。

①1 個のｲﾍﾞﾝﾄ情報を格納する構造体（0 個以上のﾚﾎﾟｰﾄ情報を格納する。）

```
typedef struct{
        TCEID    ceid;
        int      rp_count;
        TS6F11_RP_INFO ** rp_list;
} TS6F11_CE_INFO;
```

②1 個のﾘﾝｸﾚﾎﾟｰﾄ情報を格納する構造体（0 個以上の変数情報を格納する。）

```
typedef struct{
        TRPID    rpid;
        int      v_count;
        TS6F11_V_INFO     **v_list;
} TS6F11_RP_INFO;
```

③S6F11 に含む 1 個の変数情報を格納する構造体

```
typedef struct{
        TVID     vid;
        int      format;
        int      asize;
        void     *value;
        void     **link;                    // format-L, LINK TS6F11_V_INFO を指す
} TS6F11_V_INFO;
```

（3）TS6F11_CE_INFO 構造体への変数値の設定処理関連関数

C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。

.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTS6F11_CE_INFO	TS6F11_CE_INFO を初期設定する。
2	DshPutTS6F11_CE_INFO	1 個のﾚﾎﾟｰﾄ情報をｲﾍﾞﾝﾄﾚﾎﾟｰﾄ構造体に加える。
3	DshFreeTS6F11_CE_INFO	TS6F11_CE_INFO 構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。
4	DshInitTS6F11_RP_INFO	TS6F11_RP_INFO を初期設定する。
5	DshPutTS6F11_RP_INFO	ﾚﾎﾟｰﾄ情報に変数値を 1 個加える。
6	DshFreeTS6F11_RP_INFO	TS6F11_RP_INFO 構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。
7	DshMakeTS6F11_V_INFO	変数値情報を TS6F11_V_INFO 内に作成する。
8	DshFreeTS6F11_V_INFO	TS6F11_V_INFO 構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

３．２．27．１　DSH_EncodeS6F11() －　S6F11 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS6F11(
        BYTE *buffer,
        int buff_size,
        TS6F11_CE_INFO *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS6F11(
      buffer As IntPtr,
      buff_size As Integer,
      ByRef info As TS6F11_CE_INFO,
      ByRef msg_len As Integer
) As Integer
```

[C#]

```
int DSH_EncodeS6F11(
       IntPtr buffer,
       int buff_size,
       ref TS6F11_CE_INFO info,
       ref int msg_len
);
```

（2）引数

buffer	： S6F11 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	： ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ情報を格納されている構造体です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S6F11 メッセージを作成します。
info で指定された構造体 TS6F11_CE_INFO 内に含まれるイベントレポート情報を S6F11 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
char*    CLOCK  = "2013071013450412";
USHORT C_STATE  = 1;
uint   RP_ControlState    = 100;

uint   RP_ControlState    = 100;

uint   SV_Clock =         = 8192;  // A
uint   SV_ControlState    = 8194;  // U2

int    ei;
BYTE   buff[1000];
int    msg_len;
TS6F11_CE_INFO  info;
TS6F11_RP_INFO  rinfo;
TS6F11_V_INFO   vinfo;

DshInitTS6F11_CE_INFO( &info, RP_ControlState, 1 );

DshInitTS6F11_RP_INFO( &rinfo, RP_ControlState, 2 );

DshMakeTS6F11_V_INFO( &vinfo, SV_Clock, ICODE_A, strlen(CLOCK), CLOCK );
DshPutTS6F11_RP_INFO( &rinfo, &vinfo );
DshFreeTS6F11_V_INFO( &vinfo );

DshMakeTS6F11_V_INFO( &vinfo, SV_ControlState, ICODE_U2, 1, &C_STATE );
DshPutTS6F11_RP_INFO( &rinfo, &vinfo );
DshFreeTS6F11_V_INFO( &vinfo );

DshPutTS6F11_CE_INFO( &info, &rinfo );
DshFreeTS6F11_RP_INFO( &rinfo );

ei = DSH_EncodeS6F11( buff, 1000, &info, &msg_len );//
.
.
DshFreeTS6F11_CE_INFO( &info );
```

②c#

```
string CLOCK    = "2013071013450412";
UInt16 C_STATE  = 1;
uint   RP_ControlState    = 100;

uint   RP_ControlState    = 100;

uint   SV_Clock =         = 8192;  // A
uint   SV_ControlState    = 8194;  // U2
```

```
int   ei;

int   msg_len;
TS6F11_CE_INFO  info;
TS6F11_RP_INFO  rinfo;
TS6F11_V_INFO   vinfo;
IntPtr buff = Marshal.AllocCoTaskMem(1000);

DshInitTS6F11_CE_INFO( ref info, RP_ControlState, 1 );

DshInitTS6F11_RP_INFO( ref rinfo, RP_ControlState, 2 );

DshMakeTS6F11_V_INFO( ref vinfo, SV_Clock, ICODE_A, DshStrLen(CLOCK), CLOCK );
DshPutTS6F11_RP_INFO( ref rinfo, ref vinfo );
DshFreeTS6F11_V_INFO( ref vinfo );

DshMakeTS6F11_V_INFO( ref vinfo, SV_ControlState, ICODE_U2, 1, ref C_STATE );
DshPutTS6F11_RP_INFO( ref rinfo, ref vinfo );
DshFreeTS6F11_V_INFO( ref vinfo );

DshPutTS6F11_CE_INFO( ref info, ref rinfo );
DshFreeTS6F11_RP_INFO( ref rinfo );

ei = DSH_EncodeS6F11( buff, 1000, ref info, ref msg_len );    //
.
.
DshFreeTS6F11_CE_INFO( ref info );
Marshal.FreeCoTaskMem(buff);
```

(注) DshStrLen()は、DshGemPro.LIB クラスの関数であり、文字列のバイト長を取得します。

３．２．27．２　DSH_DecodeS6F11() －　S6F11 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS6F11(
        BYTE *buffer,
        int msg_len,
        TS6F11_CE_INFO *info
        );
```

[VB.Net]

```
Function DSH_DecodeS6F11(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TS6F11_CE_INFO
) As Integer
```

[C#]

```
int DSH_DecodeS6F11(
        IntPtr buffer,
        int msg_len,
        ref TS6F11_CE_INFO info
);
```

（2）引数

buffer	: S6F11 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	: S6F11 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
info	: 変数ﾃﾞｰﾀ値を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S6F11 メッセージのデコードを行います。
デコード結果は、info に格納されます。

（5）例

①c、C++

```
BYTE buffer[2000];                  // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S6F11 受信)
int msg_len = 58;                   // 受信した S6F11 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TS6F11_CE_INFO info;
int ei;
ei = DSH_DecodeS6F11( buffer, msg_len, &info );
.
.
DshFreeTS6F11_CE_INFO( &info );
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S6F11 受信)
int msg_len = 55;                   // 受信した S6F11 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TS6F11_CE_INFO info = new TS6F11_CE_INFO();
int ei = DSH_DecodeS6F11( buff, msg_len, 64, ref info );
.
.
DshFreeTS6F11_CE_INFO( ref &info );
Marshal.FreeCoTaskMem(buff);
```

３．２．27．３　DSH_EncodeS6F12() －　S6F12 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX EncodeS6F12(
        BYTE *buffer,
        int buff_size,
        int ack,
        int *msg_len
        );
```

[VB.Net]

```
Function EncodeS6F12(
        buffer As IntPtr,
        buff_size As Integer,
        ack As Integer,
        ByRef msg_len As Integer
) As Integer
```

[C#]

```
int EncodeS6F12(
        IntPtr buffer,
        int buff_size,
        int ack,
        ref int msg_len
);
```

（2）引数

buffer	： S6F12 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ack	： S6F12 の ACK です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに ack を含めて S6F12 メッセージを作成します。
作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

３．２．27．４　DSH_DecodeS6F12 () －　受信した S6F12 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS6F12 (
        BYTE *buffer,
        int msg_len,
        int *ack
        );
```

[VB.Net]

```
Function DSH_DecodeS6F12 (
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ack As Integer
) As Integer
```

[C#]

```
int DSH_DecodeS6F12 (
        IntPtr buffer,
        int msg_len,
        ref int ack
);
```

（2）引数

buffer	：	S6F12 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S6F12 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ack	：	S6F12 の ACK 格納用です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾒｯｾｰｼﾞ形式が正しくなかった。(ﾘｽﾄ構造の違い、ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞの違いなど)

（4）説明

buffer で指定されたバッファに格納されている S6F12 メッセージのデコードを行い、ACK の値を ack に返却します。

正常にデコードできた場合は、0 を返却します。また、メッセージフォーマットが SEMI 仕様に合致しなかった場合は、(-1)が返却されます。

３．２．28　S6F15 メッセージ　－　イベントレポートの送信要求

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS6F15()	S6F15 をｴﾝｺｰﾄﾞします。	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ ID をｴﾝｺｰﾄﾞします。
2	DSH_DecodeS6F15()	S6F15 をﾃﾞｺｰﾄﾞします。	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ ID をﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS6F16()	S6F16 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS6F16()	S6F16 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ情報を取得します。

（2）S6F15 のユーザインタフェース情報
情報の引き渡しは 1 個の CEID です。

（3）S6F16 のユーザインタフェース情報
情報の引き渡しは構造体 TS6F11_CE_INFO を使って行います。

①1 個のｲﾍﾞﾝﾄ情報を格納する構造体（0 個以上のﾚﾎﾟｰﾄ情報を格納する。）

```
typedef struct{
        TCEID    ceid;
        int      rp_count;
        TS6F11_RP_INFO ** rp_list;
} TS6F11_CE_INFO;
```

②1 個のﾘﾝｸﾚﾎﾟｰﾄ情報を格納する構造体（0 個以上の変数情報を格納する。）

```
typedef struct{
        TRPID    rpid;
        int      v_count;
        TS6F11_V_INFO     **v_list;
} TS6F11_RP_INFO;
```

③S6F16 に含む 1 個の変数情報を格納する構造体

```
typedef struct{
        TCEID    vid;
        int      format;
        int      asize;
        void     *value;
        void     **link;                    // format-L, LINK TS6F11_V_INFO を指す
} TS6F11_V_INFO;
```

（4）TS6F11_CE_INFO 構造体への変数値の設定処理関連関数

C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。

.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTS6F11_CE_INFO	TS6F11_CE_INFO を初期設定する。
2	DshPutTS6F11_CE_INFO	1 個のﾚﾎﾟｰﾄ情報をｲﾍﾞﾝﾄﾚﾎﾟｰﾄ構造体に加える。
3	DshFreeTS6F11_CE_INFO	TS6F11_CE_INFO 構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。
4	DshInitTS6F11_RP_INFO	TS6F11_RP_INFO を初期設定する。
5	DshPutTS6F11_RP_INFO	ﾚﾎﾟｰﾄ情報に変数値を 1 個加える。
6	DshFreeTS6F11_RP_INFO	TS6F11_RP_INFO 構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。
7	DshMakeTS6F11_V_INFO	変数値情報を TS6F11_V_INFO 内に作成する。
8	DshFreeTS6F11_V_INFO	TS6F11_V_INFO 構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

３．２．28．１　DSH_EncodeS6F15() －　S6F15 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS6F15(
        BYTE *buffer,
        int buff_size,
        TCEID *ceid,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS6F15(
        buffer As IntPtr,
        buff_size As Integer,
        ceid As UInteger,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS6F15(
        IntPtr buffer,
        int buff_size,
        uint ceid,
        ref int msg_len
       );
```

（2）引数

buffer	: S6F15 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ceid	: CEID が格納されているﾎﾟｲﾝﾀです。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S6F15 メッセージを作成します。
ceid をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
#define RP_ControlState                    100

BYTE buff[64];
int ei;
int msg_len;

ei = DSH_EncodeS6F15( buff, 64, RP_ControlState, &msg_len );
.
.
```

②C#

```
uint RP_ControlState = 100;

IntPtr buff = Marshal. AllocCoTaskMem(64);
int  msg_len = 0;
int ei = DSH_EncodeS6F15( buff, 64, RP_ControlState, ref msg_len );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．28．２　DSH_DecodeS6F15() －　S6F15 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS6F15(
        BYTE *buffer,
        int msg_len,
        TCEID *ceid
        );
```

[VB.Net]

```
Function DSH_DecodeS6F15(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ceid As UInteger
        ) As Integer
```

[C#]

```
int DSH_DecodeS6F15(
        IntPtr buffer,
        int msg_len,
        ref uint ceid
        );
```

（2）引数

buffer	: S6F15 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	: S6F15 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ceid	: CEID ｲﾍﾞﾝﾄ ID を格納する領域のﾎﾟｲﾝﾀです。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S6F15 メッセージのデコードを行います。
デコードしたイベント ID は、ceid に格納します。

（5）例

①ｃ、C++

```
BYTE buff[2000];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S6F15 受信)
int msg_len = 32;                   // 受信した S6F15 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TCEID ceid;
int ei;
ei = DSH_DecodeS6F15( buff, msg_len, &ceid );
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S6F15 受信)
int msg_len = 32;                   // 受信した S6F15 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

uint ceid = 0;
int ei = DSH_DecodeS6F15( buff, msg_len, ref ceid);
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．28．３　DSH_EncodeS6F16() －　S6F16 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS6F16(
        BYTE *buffer,
        int buff_size,
        TS6F11_CE_INFO *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS6F16(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef info As TS6F11_CE_INFO,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS6F16(
        IntPtr buffer,
        int buff_size,
        ref TS6F11_CE_INFO info,
        ref int msg_len
        );
```

（2）引数

buffer	： S6F16 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	： ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ情報を格納されている構造体です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S6F16 メッセージを作成します。
info で指定された構造体 TS6F11_CE_INFO 内に含まれるイベントレポート情報を S6F16 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
char* CLOCK    = "2013071013450412";
USHORT C_STATE  = 1;
uint   RP_ControlState    = 100;

uint   RP_ControlState    = 100;

uint   SV_Clock =         = 8192;  // A
uint   SV_ControlState    = 8194;  // U2

int    ei;
BYTE   buff[1000];
int    msg_len;
TS6F11_CE_INFO  info;
TS6F11_RP_INFO  rinfo;
TS6F11_V_INFO   vinfo;

DshInitTS6F11_CE_INFO( &info, RP_ControlState, 1 );

DshInitTS6F11_RP_INFO( &rinfo, RP_ControlState, 2 );

DshMakeTS6F11_V_INFO( &vinfo, SV_Clock, ICODE_A, strlen(CLOCK), CLOCK );
DshPutTS6F11_RP_INFO( &rinfo, &vinfo );
DshFreeTS6F11_V_INFO( &vinfo );

DshMakeTS6F11_V_INFO( &vinfo, SV_ControlState, ICODE_U2, 1, &C_STATE );
DshPutTS6F11_RP_INFO( &rinfo, &vinfo );
DshFreeTS6F11_V_INFO( &vinfo );

DshPutTS6F11_CE_INFO( &info, &rinfo );
DshFreeTS6F11_RP_INFO( &rinfo );

ei = DSH_EncodeS6F16( buff, 1000, &info, &msg_len );//
.
.
DshFreeTS6F11_CE_INFO( &info );
```

②c#

```
string CLOCK    = "2013071013450412";
UInt16 C_STATE = 1;

uint   RP_ControlState    = 100;

uint   RP_ControlState    = 100;

uint   SV_Clock =         = 8192;  // A
uint   SV_ControlState    = 8194;  // U2
```

```
int    ei;

int    msg_len = 0;
TS6F11_CE_INFO  info;
TS6F11_RP_INFO  rinfo;
TS6F11_V_INFO   vinfo;
IntPtr buff = Marshal.AllocCoTaskMem(1000);

DshInitTS6F11_CE_INFO( ref info, RP_ControlState, 1 );

DshInitTS6F11_RP_INFO( ref rinfo, RP_ControlState, 2 );

DshMakeTS6F11_V_INFO( ref vinfo, SV_Clock, ICODE_A, DshStrLen(CLOCK), CLOCK );
DshPutTS6F11_RP_INFO( ref rinfo, ref vinfo );
DshFreeTS6F11_V_INFO( ref vinfo );

DshMakeTS6F11_V_INFO( ref vinfo, SV_ControlState, ICODE_U2, 1, ref C_STATE );
DshPutTS6F11_RP_INFO( ref rinfo, ref vinfo );
DshFreeTS6F11_V_INFO( ref vinfo );

DshPutTS6F11_CE_INFO( ref info, ref rinfo );
DshFreeTS6F11_RP_INFO( ref rinfo );

ei = DSH_EncodeS6F16( buff, 1000, ref info, ref msg_len );    //
.
.
DshFreeTS6F11_CE_INFO( ref info );
Marshal.FreeCoTaskMem(buff);
```

（注）DshStrLen()は、DshGemPro.LIB クラスの関数であり、文字列のバイト長を取得します。

３．２．28．４　DSH_DecodeS6F16() －　S6F16 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS6F16(
        BYTE *buffer,
        int msg_len,
        TS6F11_CE_INFO *info
        );
```

[VB.Net]

```
Function DSH_DecodeS6F16(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TS6F11_CE_INFO
       ) As Integer
```

[C#]

```
int DSH_DecodeS6F16(
        IntPtr buffer,
        int msg_len,
        ref TS6F11_CE_INFO info
        );
```

（2）引数

buffer	： S6F16 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S6F16 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
info	： 変数ﾃﾞｰﾀ値を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S6F16 メッセージのデコードを行います。
デコード結果は、info に格納されます。

（5）例

①c、C++

```
BYTE buffer[2000];                  // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S6F16 受信)
int msg_len = 58;                   // 受信した S6F15 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TS6F11_CE_INFO info;
int ei;
ei = DSH_DecodeS6F16( buffer, msg_len, &info );
.
.
DshFreeTS6F11_CE_INFO( &info );
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S6F16 受信)
int msg_len = 55;                   // 受信した S6F15 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TS6F11_CE_INFO info = new TS6F11_CE_INFO();

int ei = DSH_DecodeS6F16( buff, msg_len, ref info );
.
.
DshFreeTS6F11_CE_INFO( ref &info );
Marshal.FreeCoTaskMem(buff);
```

３．２．29　S6F19 メッセージ　－　個別レポートの送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS6F19()	S6F19 をｴﾝｺｰﾄﾞします。	ﾚﾎﾟｰﾄ ID をｴﾝｺｰﾄﾞします。
2	DSH_DecodeS6F19()	S6F19 をﾃﾞｺｰﾄﾞします。	ﾚﾎﾟｰﾄ ID をﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS6F20()	S6F20 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	ﾚﾎﾟｰﾄ情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS6F20()	S6F20 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	ﾚﾎﾟｰﾄ情報を取得します。

（2）S6F19 のユーザインタフェース情報
情報の引き渡しは１個の RPID です。

（3）S6F20 のユーザインタフェース情報
情報の引き渡しは構造体 TS6F11_RP_INFO を使って行います。

①1 個のﾘﾝｸﾚﾎﾟｰﾄ情報を格納する構造体（0 個以上の変数情報を格納する。）

```
typedef struct{
        TRPID    rpid;
        int      v_count;
        TS6F11_V_INFO     **v_list;
} TS6F11_RP_INFO;
```

②S6F20 に含む１個の変数情報を格納する構造体

```
typedef struct{
        TRPID    vid;
        int      format;
        int      asize;
        void     *value;
        void     **link;                    // format-L, LINK TS6F11_V_INFO を指す
} TS6F11_V_INFO;
```

（4）TS6F11_RP_INFO 構造体への変数値の設定処理関連関数
C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。
.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTS6F11_RP_INFO	TS6F11_RP_INFO を初期設定する。
2	DshPutTS6F11_RP_INFO	ﾚﾎﾟｰﾄ情報に変数値を１個加える。
3	DshFreeTS6F11_RP_INFO	TS6F11_RP_INFO 構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。
4	DshMakeTS6F11_V_INFO	変数値情報を TS6F11_V_INFO 内に作成する。
5	DshFreeTS6F11_V_INFO	TS6F11_V_INFO 構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

３．２．29．１　DSH_EncodeS6F19() －　S6F19 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS6F19(
        BYTE *buffer,
        int buff_size,
        TRPID *rpid,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS6F19(
        buffer As IntPtr,
        buff_size As Integer,
        rpid As UInteger,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS6F19(
        IntPtr buffer,
        int buff_size,
        uint rpid,
        ref int msg_len
       );
```

（2）引数

buffer	: S6F19 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
rpid	: RPID が格納されているﾎﾟｲﾝﾀです。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S6F19 メッセージを作成します。
rpid をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

（5）例

①C/C++

```
#define RP_ControlState                    100

BYTE buff[64];
int ei;
int msg_len;

ei = DSH_EncodeS6F19( buff, 64, RP_ControlState, &msg_len );
.
.
```

②C#

```
uint RP_ControlState = 100;

IntPtr buff = Marshal. AllocCoTaskMem(64);
int  msg_len = 0;
int ei = DSH_EncodeS6F19( buff, 64, RP_ControlState, ref msg_len );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．29．２　DSH_DecodeS6F19() －　S6F19 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS6F19(
        BYTE *buffer,
        int msg_len,
        TRPID *rpid
        );
```

[VB.Net]

```
Function DSH_DecodeS6F19(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef rpid As UInteger
       ) As Integer
```

[C#]

```
int DSH_DecodeS6F19(
        IntPtr buffer,
        int msg_len,
        ref uint rpid
        );
```

（2）引数

buffer	：	S6F19 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S6F19 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
rpid	：	RPID を格納する領域のﾎﾟｲﾝﾀです。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S6F19 メッセージのデコードを行います。
デコードしたレポート ID は、rpid に格納します。

（5）例

①c、C++

```
BYTE buff[2000];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S6F19 受信)
int msg_len = 6;                    // 受信した S6F19 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TRPID rpid;
int ei;
ei = DSH_DecodeS6F19( buff, msg_len, &rpid );
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S6F19 受信)
int msg_len = 6;                    // 受信した S6F19 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

uint rpid = 0;
int ei = DSH_DecodeS6F19( buff, msg_len, ref rpid);
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．29．３　DSH_EncodeS6F20() －　S6F20 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS6F20(
        BYTE *buffer,
        int buff_size,
        TS6F11_RP_INFO *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS6F20(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef info As TS6F11_RP_INFO,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS6F20(
        IntPtr buffer,
        int buff_size,
        ref TS6F11_RP_INFO info,
        ref int msg_len
       );
```

（2）引数

buffer	： S6F20 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	： ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ情報が格納されている構造体です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S6F20 メッセージを作成します。
info で指定された構造体 TS6F11_RP_INFO 内に含まれるイベントレポート情報を S6F20 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
char* CLOCK    = "2013071013450412";
USHORT C_STATE  = 1;

uint   RP_ControlState    = 100;

uint   SV_Clock =         = 8192;  // A
uint   SV_ControlState    = 8194;  // U2

int    ei;
BYTE   buff[1000];
int    msg_len;
TS6F11_RP_INFO  rinfo;
TS6F11_V_INFO   vinfo;

DshInitTS6F11_RP_INFO( &rinfo, RP_ControlState, 2 );

DshMakeTS6F11_V_INFO( &vinfo, SV_Clock, ICODE_A, strlen(CLOCK), CLOCK );
DshPutTS6F11_RP_INFO( &rinfo, &vinfo );
DshFreeTS6F11_V_INFO( &vinfo );

DshMakeTS6F11_V_INFO( &vinfo, SV_ControlState, ICODE_U2, 1, &C_STATE );
DshPutTS6F11_RP_INFO( &rinfo, &vinfo );
DshFreeTS6F11_V_INFO( &vinfo );

ei = DSH_EncodeS6F20( buff, 1000, &info, &msg_len );
.
.
DshFreeTS6F11_RP_INFO( &rinfo );
```

②c#

```
string CLOCK    = "2013071013450412";
UInt16 C_STATE = 1;

uint   RP_ControlState    = 100;

uint   SV_Clock =         = 8192;  // A
uint   SV_ControlState    = 8194;  // U2

int    ei;

int    msg_len = 0;
TS6F11_RP_INFO  rinfo;
TS6F11_V_INFO   vinfo;
IntPtr buff = Marshal.AllocCoTaskMem(1000);

DshInitTS6F11_RP_INFO( ref rinfo, RP_ControlState, 2 );
```

```
DshMakeTS6F11_V_INFO( ref vinfo, SV_Clock, ICODE_A, DshStrLen(CLOCK), CLOCK );
DshPutTS6F11_RP_INFO( ref rinfo, ref vinfo );
DshFreeTS6F11_V_INFO( ref vinfo );

DshMakeTS6F11_V_INFO( ref vinfo, SV_ControlState, ICODE_U2, 1, ref C_STATE );
DshPutTS6F11_RP_INFO( ref rinfo, ref vinfo );
DshFreeTS6F11_V_INFO( ref vinfo );

ei = DSH_EncodeS6F20( buff, 1000, ref info, ref msg_len );    //
.
.
DshFreeTS6F11_RP_INFO( ref rinfo );
Marshal.FreeCoTaskMem(buff);
```

(注) DshGemPro.LIB クラスの関数であり、文字列のバイト長を取得します。

３．２．29．４　DSH_DecodeS6F20() －　S6F20 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS6F20(
        BYTE *buffer,
        int msg_len,
        TS6F11_RP_INFO *info
        );
```

[VB.Net]

```
Function DSH_DecodeS6F20(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TS6F11_RP_INFO
       ) As Integer
```

[C#]

```
int DSH_DecodeS6F20(
        IntPtr buffer,
        int msg_len,
        ref TS6F11_RP_INFO info
        );
```

（2）引数

buffer	： S6F20 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S6F20 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
info	： ﾚﾎﾟｰﾄ情報を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S6F20 メッセージのデコードを行います。
デコード結果は、info に格納されます。

（5）例

①c、C++

```
BYTE buffer[2000];                  // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S6F20 受信)
int msg_len = 24;                   // 受信した S6F20 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

TS6F11_RP_INFO info;
int ei;

ei = DSH_DecodeS6F20( buffer, msg_len, &info );
.
.
DshFreeTS6F11_RP_INFO( &info );
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S6F20 受信)
int msg_len = 24;                   // 受信した S6F20 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TS6F11_RP_INFO info = new TS6F11_RP_INFO();

int ei = DSH_DecodeS6F20( buff, msg_len, ref info );
.
.
DshFreeTS6F11_RP_INFO( ref &info );
Marshal.FreeCoTaskMem(buff);
```

３．２．30　S6F23 メッセージ　－　スプールﾃﾞｰﾀ要求の送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS6F23()	S6F23 をｴﾝｺｰﾄﾞします。	RSDC をｴﾝｺｰﾄﾞします。 （RSDC ： ｽﾌﾟｰﾙ要求ｺｰﾄﾞ）
2	DSH_DecodeS6F23()	S6F23 をﾃﾞｺｰﾄﾞします。	RSDC をﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS6F24()	S6F24 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	RSDA をｴﾝｺｰﾄﾞします。 （RSDA ： ｽﾌﾟｰﾙ要求確認ｺｰﾄﾞ）
4	DSH_DecodeS6F24()	S6F24 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	RSDA を取得します。

（2）S6F23 のユーザインタフェース情報
　　情報の引き渡しは１個の RSDC（ｽﾌﾟｰﾙ要求ｺｰﾄﾞ）です。

（3）S6F24 のユーザインタフェース情報
　　情報の引き渡しは１個の RSDA（ｽﾌﾟｰﾙ要求確認ｺｰﾄﾞ）です。

３．２．30．１　DSH_EncodeS6F23() －　S6F23 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS6F23(
        BYTE *buffer,
        int buff_size,
        int rsdc,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS6F23(
        buffer As IntPtr,
        buff_size As Integer,
        rsdc As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS6F23(
        IntPtr buffer,
        int buff_size,
        int rsdc,
        ref int msg_len
       );
```

（2）引数

buffer	： S6F23 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
rsdc	： RSDC(ｽﾌﾟｰﾙ要求ｺｰﾄﾞ)が格納されているﾎﾟｲﾝﾀです。(0=転送要求, 1=破棄要求)
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S6F23 メッセージを作成します。
rsdc をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

（5）例

①C/C++

```
int rsdc = 0;

BYTE buff[64];
int ei;
int msg_len;

ei = DSH_EncodeS6F23( buff, 64, rsdc, &msg_len );
.
.
```

②C#

```
int rsdc = 0;

IntPtr buff = Marshal. AllocCoTaskMem(64);
int  msg_len = 0;
int ei = DSH_EncodeS6F23( buff, 64, rsdc, ref msg_len );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．30．２　DSH_DecodeS6F23() －　S6F23 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS6F23(
        BYTE *buffer,
        int msg_len,
        int *rsdc
        );
```

[VB.Net]

```
Function DSH_DecodeS6F23(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef rsdc As Integer
       ) As Integer
```

[C#]

```
int DSH_DecodeS6F23(
        IntPtr buffer,
        int msg_len,
        ref int rsdc
        );
```

（2）引数

buffer	： S6F23 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S6F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
rsdc	： ｽﾌﾟｰﾙ要求ｺｰﾄﾞを格納する領域のﾎﾟｲﾝﾀです。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S6F23 メッセージのデコードを行います。
デコードしたスプール要求コードは、rsdc に格納します。

（5）例

①c、C++

```
BYTE buff[2000];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S6F23 受信)
int msg_len = 6;                    // 受信した S6F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
int rsdc;
int ei;
ei = DSH_DecodeS6F23( buff, msg_len, &rsdc );
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S6F23 受信)
int msg_len = 6;                    // 受信した S6F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

uint rsdc = 0;
int ei = DSH_DecodeS6F23( buff, msg_len, ref rsdc);
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．30．３　DSH_EncodeS6F24() －　S6F24 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS6F24(
        BYTE *buffer,
        int buff_size,
        int rsda,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS6F24(
        buffer As IntPtr,
        buff_size As Integer,
        rsda As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS6F24(
        IntPtr buffer,
        int buff_size,
        int rsda,
       ref int msg_len
       );
```

（2）引数

buffer	: S6F24 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
rsda	: RSDA(ｽﾌﾟｰﾙ要求確認ｺｰﾄﾞ 0=OK, 1=拒否,busy, retry, 2=拒否,no spool data)
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S6F24 メッセージを作成します。
rsda を S6F24 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
int   rsda = 0;
int   ei;
BYTE  buff[1000;
int   msg_len;

ei = DSH_EncodeS6F24( buff, 100, rsda, &msg_len );  //
.
.
```

②c#

```
int   rsda = 0
int   ei;
int   msg_len = 0;

ei = DSH_EncodeS6F24( buff, 1000, rsda, ref msg_len );       //
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．30．４　DSH_DecodeS6F24() －　S6F24 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS6F24(
        BYTE *buffer,
        int msg_len,
        int *rsda
        );
```

[VB.Net]

```
Function DSH_DecodeS6F24(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef rsda As Integer
       ) As Integer
```

[C#]

```
int DSH_DecodeS6F24(
        IntPtr buffer,
        int msg_len,
        ref int rsda
        );
```

（2）引数

buffer	：	S6F24 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S6F24 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
rsda	：	RSDA 格納用です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S6F24 メッセージのデコードを行います。
デコード結果は、rsda に格納されます。

（5）例

①c、C++

```
BYTE buffer[100];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S6F24 受信)
int msg_len = 3;                     // 受信した S6F24 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

int rsda;
int ei;

ei = DSH_DecodeS6F24( buffer, msg_len, &rsda );
.
.
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S6F24 受信)
int msg_len = 3;                     // 受信した S6F24 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

int rsda =0 ;

int ei = DSH_DecodeS6F24( buff, msg_len, ref rsda );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．31　S7F1 メッセージ　－　プロセスプログラム・ロード問合せ

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS7F1()	S7F1 をｴﾝｺｰﾄﾞします。	PP ﾛｰﾄﾞ問合せﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。
2	DSH_DecodeS7F1()	S7F1 をﾃﾞｺｰﾄﾞします。	PP ﾛｰﾄﾞ問合せﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS7F2()	S7F2 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	ﾛｰﾄﾞ許可情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS7F2()	S7F2 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	ﾛｰﾄﾞ許可情報を取得します。

（2）S7F1 のユーザインタフェース情報
　　情報の引き渡しは、PPID と PP 情報のバイト長です。

（3）S7F2 のユーザインタフェース情報
　　情報の引き渡しはﾛｰﾄﾞ許可情報です。(=ACK の意味)

３．２．31．１　DSH_EncodeS7F1() －　S7F1 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F1(
        BYTE *buffer,
        int buff_size,
        char *ppid,
        int length,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F1(
        buffer As IntPtr,
        buff_size As Integer,
        ppid As String,
        length As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F1(
        IntPtr buffer,
        int buff_size,
        string ppid,
        int length,
        ref int msg_len
       );
```

（2）引数

buffer	： S7F1 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ppid	： PPID(ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID)です。
length	： 問合せ PP のﾊﾞｲﾄ長です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F1 メッセージを作成します。
ppid と length をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、0 を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

（5）例

①C/C++

```
char *ppid = “PPID1000”;
int length = 64;

BYTE buff[100];
int ei;
int msg_len;

ei = DSH_EncodeS7F1( buff, 100, ppid, length, &msg_len );
.
.
```

②C#

```
string ppid =  “PPID1000”;
int length = 64;

IntPtr buff = Marshal. AllocCoTaskMem(100);
int  msg_len = 0;
int ei = DSH_EncodeS7F1( buff, 100, ppid, length, ref msg_len );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．31．２　DSH_DecodeS7F1() －　S7F1 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F1(
        BYTE *buffer,
        int msg_len,
        char *ppid,
        int *length
        );
```

[VB.Net]

```
Function DSH_DecodeS7F1(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ppid As String,
        ByRef length As Integer
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F1(
        IntPtr buffer,
        int msg_len,
        ref IntPtr ppid,
        ref int length
        );
```

（2）引数

buffer	： S7F1 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S7F1 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ppid	： ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID 格納です。(PPID に必要な十分の領域を準備してください)
length	： PPID ﾊﾞｲﾄ長を格納する領域のﾎﾟｲﾝﾀです。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F1 メッセージのデコードを行います。
デコードした PPID と同バイト長はそれぞれ、ppid, length に格納します。

（5）例

①c、C++

```
BYTE buff[1000];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F1 受信)
int msg_len = 17;                  // 受信した S7F1 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
char ppid[128];
int length;
int ei;
ei = DSH_DecodeS7F1( buff, msg_len, ppid, &length );
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(1000);
(S7F1 受信)
int msg_len = 17;                  // 受信した S7F1 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
IntPtr ppid = Marshal.AllocCoTasmMem(128);
uint length = 0;

int ei = DSH_DecodeS7F1( buff, msg_len, ppid, ref length);
string pp_id = Marshal. PtrToStringAnsi(ppid);
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．31．３　DSH_EncodeS7F2() －　S7F2 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F2(
        BYTE *buffer,
        int buff_size,
        int ppgnt,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F2(
        buffer As IntPtr,
        buff_size As Integer,
        ppgnt As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F2(
        IntPtr buffer,
        int buff_size,
        int ppgnt,
        ref int msg_len
       );
```

（2）引数

buffer	：S7F2 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	：buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ppgnt	：ﾛｰﾄﾞ許可ｺｰﾄﾞです。
msg_len	：ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F2 メッセージを作成します。
ppgnt を S7F2 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
int   ppgnt = 0;
int   ei;
BYTE  buff[1000;
int   msg_len;

ei = DSH_EncodeS7F2( buff, 100, ppgnt, &msg_len );  //
.
.
```

②c#

```
int   ppgnt = 0
int   ei;
int   msg_len = 0;

ei = DSH_EncodeS7F2( buff, 1000, ppgnt, ref msg_len );       //
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．31．４　DSH_DecodeS7F2() －　S7F2 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F2(
        BYTE *buffer,
        int msg_len,
        int *ppgnt
        );
```

[VB.Net]

```
Function DSH_DecodeS7F2(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ppgnt As Integer
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F2(
        IntPtr buffer,
        int msg_len,
        ref int ppgnt
        );
```

（2）引数

buffer	：	S7F2 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S7F2 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ppgnt	：	ﾛｰﾄﾞ許可ｺｰﾄﾞです。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F2 メッセージのデコードを行います。
デコード結果は、ppgnt に格納されます。

（5）例

①ｃ、C++

```
BYTE buffer[100];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F2 受信)
int msg_len = 3;                     // 受信した S7F2 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

int ppgnt;
int ei;

ei = DSH_DecodeS7F2( buffer, msg_len, &ppgnt );
.
.
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S7F2 受信)
int msg_len = 3;                     // 受信した S7F2 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

int ppgnt =0 ;

int ei = DSH_DecodeS7F2( buff, msg_len, ref ppgnt );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．32　S7F3 メッセージ　－　PP プロセスプログラム送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS7F3()	S7F3 をｴﾝｺｰﾄﾞします。	PP 送信ﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。
2	DSH_DecodeS7F3()	S7F3 をﾃﾞｺｰﾄﾞします。	PP 送信ﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS7F4()	S7F4 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	確認情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS7F4()	S7F4 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	確認情報を取得します。

（2）S7F3 のユーザインタフェース情報
　　情報の引き渡しは、PPID と PPBODY 情報です。

（3）S7F4 のユーザインタフェース情報
　　情報の引き渡しはプロセスプログラム送信確認です。(=ackc7)

３．２．32．１　DSH_EncodeS7F3() －　S7F3 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F3(
        BYTE *buffer,
        int buff_size,
        char *ppid,
        char *ppbody,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F3(
        buffer As IntPtr,
        buff_size As Integer,
        ppid As String,
        ppbody As String,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F3(
        IntPtr buffer,
        int buff_size,
        string ppid,
        string ppbody,
        ref int msg_len
       );
```

（2）引数

buffer	: S7F3 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ppid	: PPID(ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID)です。
ppbody	: PP の本体です。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F3 メッセージを作成します。
ppid と ppbody をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、0 を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

（5）例

①C/C++

```
char *ppid = “PPID1000”;
char *ppbody = “PPBODY100200300”;

BYTE buff[1000];
int ei;
int msg_len;

ei = DSH_EncodeS7F3( buff, 1000, ppid, ppbody, &msg_len );
.
.
```

②C#

```
string ppid =  “PPID1000”;
string ppbody = “PPBODY100200300”;

IntPtr buff = Marshal. AllocCoTaskMem(1000);
int  msg_len = 0;
int ei = DSH_EncodeS7F3( buff, 1000, ppid, ppbody, ref msg_len );
.
.
Marshal.FreeCoTaskMem(send_buff);
```

３．２．32．２　DSH_DecodeS7F3() －　S7F3 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F3(
        BYTE *buffer,
        int msg_len,
        char *ppid,
        char *ppbody
        );
```

[VB.Net]

```
Function DSH_DecodeS7F3(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ppid As String,
        ByRef ppbody As String
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F3(
        IntPtr buffer,
        int msg_len,
        ref IntPtr ppid,
        ref IntPtr ppbody
        );
```

（2）引数

buffer	： S7F3 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S7F3 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ppid	： ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID 格納です。(PPID に必要な十分の領域を準備してください)
ppbody	： PP の本体情報を格納する領域のﾎﾟｲﾝﾀです。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F3 メッセージのデコードを行います。
デコードした PPID、PPBODY はそれぞれ、ppid, ppbody に格納します。

（5）例

①c、C++

```
BYTE buff[1000];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F3受信)
int msg_len = 17;                  // 受信したS7F3ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
char ppid[128];
char ppbody[512];
int ei;
ei = DSH_DecodeS7F3( buff, msg_len, ppid, ppbody );
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(1000);
(S7F3受信)
int msg_len = 17;                  // 受信したS7F3ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
IntPtr ppid = Marshal.AllocCoTasmMem(128);
IntPtr ppbody = Marshal.AllocCoTasmMem(512);

int ei = DSH_DecodeS7F3( buff, msg_len, ppid, ref ppbody);
string pp_id = Marshal. PtrToStringAnsi(ppid);
string pp_body = Marshal. PtrToStringAnsi(ppbody);
.
.
Marshal.FreeCoTaskMem(ppid);
Marshal.FreeCoTaskMem(ppbody);
Marshal.FreeCoTaskMem(buff);
```

３．２．32．３　DSH_EncodeS7F4() －　S7F4 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F4(
        BYTE *buffer,
        int buff_size,
        int ack,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F4(
        buffer As IntPtr,
        buff_size As Integer,
        ack As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F4(
        IntPtr buffer,
        int buff_size,
        int ack,
        ref int msg_len
       );
```

（2）引数

buffer	: S7F4 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ack	: PP 送信確認ｺｰﾄﾞです。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F4 メッセージを作成します。
ack を S7F4 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
int   ack = 0;
int   ei;
BYTE  buff[100;
int   msg_len;

ei = DSH_EncodeS7F4( buff, 100, ack, &msg_len );    //
.
.
```

②c#

```
int   ack = 0
int   ei;

IntPtr buff = Marshal. AllocCoTaskMem(1000);
int   msg_len = 0;

ei = DSH_EncodeS7F4( buff, 100, ack, ref msg_len ); //
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．32．４　DSH_DecodeS7F4() －　S7F4 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F4(
        BYTE *buffer,
        int msg_len,
        int *ack
        );
```

[VB.Net]

```
Function DSH_DecodeS7F4(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ack As Integer
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F4(
        IntPtr buffer,
        int msg_len,
        ref int ack
        );
```

（2）引数

buffer	： S7F4 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S7F4 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ack	： PP 送信確認ｺｰﾄﾞ格納用です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F4 メッセージのデコードを行います。
デコード結果は、ack に格納されます。

（5）例

①ｃ、C++

```
BYTE buff[100];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F4受信)
int msg_len = 3;                   // 受信したS7F4ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

int ack;
int ei;

ei = DSH_DecodeS7F4( buff, msg_len, &ack );
.
.
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S7F4受信)
int msg_len = 3;                   // 受信したS7F4ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

int ack =0 ;
IntPtr buff = Marshal. AllocCoTaskMem(1000);

int ei = DSH_DecodeS7F4( buff, msg_len, ref ack );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．33　S7F5 メッセージ　－　プロセスプログラム送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS7F5()	S7F5 をｴﾝｺｰﾄﾞします。	PP 要求ﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。
2	DSH_DecodeS7F5()	S7F5 をﾃﾞｺｰﾄﾞします。	PP 要求ﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS7F6()	S7F6 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	PPID, PPBODY をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS7F6()	S7F6 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	PPID, PPBODY 情報を取得します。

（2）S7F5 のユーザインタフェース情報
　　情報の引き渡しは、PPID です。

（3）S7F6 のユーザインタフェース情報
　　情報の引き渡しは PPID と PPBODY です。

３．２．33．１　DSH_EncodeS7F5() －　S7F5 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F5(
        BYTE *buffer,
        int buff_size,
        char *ppid,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F5(
        buffer As IntPtr,
        buff_size As Integer,
        ppid As String,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F5(
        IntPtr buffer,
        int buff_size,
        string ppid,
        ref int msg_len
       );
```

（2）引数

buffer	: S7F5 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ppid	: PPID(ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID)です。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F5 メッセージを作成します。
ppid をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

（5）例

①C/C++

```
char *ppid = “PPID1000”;

BYTE send_buffer[128];
int ei;
int msg_len;

ei = DSH_EncodeS7F5( send_buffer, 128, ppid, &msg_len );
.
.
```

②C#

```
string ppid =  “PPID1000”;

IntPtr send_buffer = Marshal. AllocCoTaskMem(128);
int  msg_len = 0;
int ei = DSH_EncodeS7F5( send_buffer, 128, ppid, ref msg_len );
.
.
Marshal.FreeCoTaskMem(send_buff);
```

３．２．33．２　DSH_DecodeS7F5() －　S7F5 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F5(
        BYTE *buffer,
        int msg_len,
        char *ppid
        );
```

[VB.Net]

```
Function DSH_DecodeS7F5(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ppid As String
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F5(
        IntPtr buffer,
        int msg_len,
        ref IntPtr ppid
        );
```

（2）引数

buffer	： S7F5 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S7F5 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ppid	： ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID 格納です。(PPID に必要な十分の領域を準備してください)

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F5 メッセージのデコードを行います。
デコードした PPID は ppid に格納します。

（5）例

①c、C++

```
BYTE buff[1000];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F5 受信)
int msg_len = 17;                  // 受信した S7F5 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
char ppid[128];
int ei;
ei = DSH_DecodeS7F5( buff, msg_len, ppid );
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(1000);
(S7F5 受信)
int msg_len = 17;                  // 受信した S7F5 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
IntPtr ppid = Marshal.AllocCoTasmMem(128);

int ei = DSH_DecodeS7F5( buff, msg_len, ppid );
string pp_id = Marshal. PtrToStringAnsi(ppid);
.
.
Marshal.FreeCoTaskMem(ppid);
Marshal.FreeCoTaskMem(buff);
```

３．２．33．３　DSH_EncodeS7F6() －　S7F6 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F6(
        BYTE *buffer,
        int buff_size,
        char *ppid,
        char *ppbody,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F6(
        buffer As IntPtr,
        buff_size As Integer,
        ppid As String,
        ppbody As String,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F6(
        IntPtr buffer,
        int buff_size,
        string ppid,
        string ppbody,
        ref int msg_len
        );
```

（2）引数

buffer	: S7F6 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ppid	: PPID(ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID)です。
ppbody	: PP の本体です。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F6 メッセージを作成します。
ppid, ppbody を S7F6 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。

作成したメッセージのバイトサイズが　buff_size 以内であれば、0 を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

（5）例

①C/C++

```
char *ppid = “PPID1000”;
char *ppbody = “PPBODY100200300”;
int    ei;
BYTE   buff[1000];
int    msg_len;

ei = DSH_EncodeS7F6( buff, 1000, ppid, ppbody, &msg_len );    //
.
.
```

②c#

```
string ppid = “PPID1000”;
string ppbody = “PPBODY100200300”;
int    ei;

IntPtr buff = Marshal. AllocCoTaskMem(1000);
int    msg_len = 0;

ei = DSH_EncodeS7F6( buff, 100, ppid, ppbody, ref msg_len ); //
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．33．４　DSH_DecodeS7F6() －　S7F6 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F6(
        BYTE *buffer,
        int msg_len,
        char *ppid,
        char *ppbody
        );
```

[VB.Net]

```
Function DSH_DecodeS7F6(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ppid As String,
        ByRef ppbody As String
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F6(
        IntPtr buffer,
        int msg_len,
        ref IntPtr ppid,
        ref IntPtr ppbody
        );
```

（2）引数

buffer	： S7F6 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	： S7F6 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ppid	： ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID 格納です。(PPID に必要な十分の領域を準備してください)
ppbody	： PP の本体情報を格納する領域のﾎﾟｲﾝﾀです。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F6 メッセージのデコードを行います。
デコードした PPID、PPBODY はそれぞれ、ppid, ppbody に格納します。

（5）例

①ｃ、C++

```
BYTE buff[100];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F6 受信)
int msg_len = 17;                  // 受信した S7F3 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
char ppid[128];
char ppbody[512];

int ei;

ei = DSH_DecodeS7F6( buff, msg_len, ppid, ppbody);
.
.
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S7F6 受信)
int msg_len = 3;                   // 受信した S7F6 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

int msg_len = 17;                  // 受信した S7F3 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
IntPtr ppid = Marshal.AllocCoTasmMem(128);
IntPtr ppbody = Marshal.AllocCoTasmMem(512);

IntPtr buff = Marshal. AllocCoTaskMem(1000);

int ei = DSH_DecodeS7F6( buff, msg_len, ppid, ppbody );
string pp_id = Marshal. PtrToStringAnsi(ppid);
string pp_body = Marshal. PtrToStringAnsi(ppbody);
.
.
Marshal.FreeCoTaskMem(ppid);
Marshal.FreeCoTaskMem(ppbody);
Marshal.FreeCoTaskMem(buff);
```

３．２．34　S7F17 メッセージ　－　プロセスプログラム削除支持

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS7F17()	S7F17 をｴﾝｺｰﾄﾞします。	PPID の削除ﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。
2	DSH_DecodeS7F17()	S7F17 をﾃﾞｺｰﾄﾞします。	PPID の削除ﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS7F18()	S7F18 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	ack をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS7F18()	S7F18 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	ack を取得します。

（2）S7F17 のユーザインタフェース情報

情報の引き渡しは、削除したい PPID が格納されている TPPID_LIST 構造体を使用します。

```
typedef struct{
        int         count;
        char        **ppid_list;
} TPPID_LIST;
```

（3）S7F18 のユーザインタフェース情報

ACKC7 です。

３．２．34．１　DSH_EncodeS7F17() －　S7F17 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F17(
        BYTE *buffer,
        int buff_size,
        TPPID_LIST *list,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F17(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef list As TPPID_LIST,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F17(
        IntPtr buffer,
        int buff_size,
        ref TPPID_LIST list,
        ref int msg_len
       );
```

（2）引数

buffer	：S7F17 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	：buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
list	：TPPID_LIST 構造体です。
msg_len	：ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F17 メッセージを作成します。
list に設定されている PPID をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
BYTE send_buffer[1000];
int ei;
int msg_len;

char* ppid_list[32] = { "PPID100", "PPID200", "PPID300" };

TPPID_LIST  list;

DshInitTPPID_LIST( &list, 3 );
DshPutTPPID_LIST( &list, ppid_list[0] );
DshPutTPPID_LIST( &list, ppid_list[1] );
DshPutTPPID_LIST( &list, ppid_list[2] );

ei = DSH_EncodeS7F17( send_buffer, 1000, &list, &msg_len );
.
.
DshFreeTPPID_LIST( &list );
```

②C#

```
TPPID_LIST  list = new TPPID_LIST();
string ppbody = "PPBODY100200300";

IntPtr send_buffer = Marshal. AllocCoTaskMem(1000);
int  msg_len = 0;

int PP_COUNT = 3;
string[] ppid_list = { "PPID100", "PPID200", "PPID300" };

DshInitTPPID_LIST(ref list, PP_COUNT);
for (int i = 0; i < PP_COUNT; i++)
{
    DshGemPro.LIB.DshPutTPPID_LIST(ref list, ppid_list[i]);
}
int ei = DSH_EncodeS7F17( send_buffer, 1000, ref list, ref msg_len );
.
.
DshFreeTPPID_LIST( ref list );
Marshal.FreeCoTaskMem(send_buff);
```

３．２．34．２　DSH_DecodeS7F17() －　S7F17 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F17(
        BYTE *buffer,
        int msg_len,
        TPPID_LIST *list
        );
```

[VB.Net]

```
Function DSH_DecodeS7F17(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef list As TPPID_LIST
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F17(
        IntPtr buffer,
        int msg_len,
        ref TPPID_LIST list
        );
```

（2）引数

buffer	：	S7F17 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S7F17 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
list	：	PPID を格納する構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F17 メッセージのデコードを行います。
デコードした PPID 情報は list に格納します。

（5）例

①c、C++

```
BYTE buff[1000];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F17 受信)
int msg_len = 17;                   // 受信した S7F17 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TPPID_LIST list;
int ei;
ei = DSH_DecodeS7F17( buff, msg_len, &list );
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(1000);
(S7F17 受信)
int msg_len = 17;                   // 受信した S7F17 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TPPID_LIST list = new TPPID_LIST();

int ei = DSH_DecodeS7F17( buff, msg_len, ref list );
string pp_id = Marshal.PtrToStringAnsi(ppid);
.
.
Marshal.FreeCoTaskMem(list);
Marshal.FreeCoTaskMem(buff);
```

３．２．34．３　DSH_EncodeS7F18() －　S7F18 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F18(
        BYTE *buffer,
        int buff_size,
        int ack,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F18(
        buffer As IntPtr,
        buff_size As Integer,
        ack As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F18(
        IntPtr buffer,
        int buff_size,
        int ack,
        ref int msg_len
       );
```

（2）引数

buffer	：S7F18 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	：buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ack	：ACKC7 です。
msg_len	：ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F18 メッセージを作成します。
S7F18 メッセージに ack をエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

３．２．34．４　DSH_DecodeS7F18() －　S7F18 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F18(
        BYTE *buffer,
        int msg_len,
        int *ack
        );
```

[VB.Net]

```
Function DSH_DecodeS7F18(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ack As Integer
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F18(
        IntPtr buffer,
       int msg_len,
       ref int ack
       );
```

（2）引数

buffer	: S7F18 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	: S7F18 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ack	: ACKC7 格納用です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F18 メッセージのデコードを行います。
デコードし、ACKC7 を ack に取得します。

３．２．35　S7F19 メッセージ　－　現在のプロセスプログラム要求

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS7F19()	S7F19 をｴﾝｺｰﾄﾞします。	現在の PPID(EPPD)要求をｴﾝｺｰﾄﾞします。(Header のみ)
2	DSH_DecodeS7F19()	S7F19 をﾃﾞｺｰﾄﾞします。	現在の PPID(EPPD)要求をﾃﾞｰﾄﾞします。(Header のみ)
3	DSH_EncodeS7F20()	S7F20 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	現在の PPID　ﾘｽﾄをｴﾝｺｰﾄﾞします。
4	DSH_DecodeS7F20()	S7F20 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	現在の PPID　ﾘｽﾄをﾃﾞｺｰﾄﾞします。

（2）S7F19 のユーザインタフェース情報

Header のみです。

（3）S7F20 のユーザインタフェース情報

TPPID_LIST 構造体を使用します。構造体内に PPID を格納します。

```
typedef struct{
        int         count;
        char        **ppid_list;
} TPPID_LIST;
```

３．２．35．１　DSH_EncodeS7F19() －　S7F19 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F19(
        BYTE *buffer,
        int buff_size,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F19(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F19(
        IntPtr buffer,
        int buff_size,
        ref int msg_len
       );
```

（2）引数

buffer	： S7F19 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F19 メッセージを作成します。(Header のみです)

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

３．２．35．２　DSH_DecodeS7F19() －　S7F19 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F19(
        BYTE *buffer,
        int msg_len,
        );
```

[VB.Net]

```
Function DSH_DecodeS7F19(
        buffer As IntPtr,
        msg_len As Integer,
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F19(
        IntPtr buffer,
        int msg_len,
        );
```

（2）引数

buffer　　：S7F19 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。

msg_len　　：S7F19 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。
(Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F19 メッセージのデコードを行います。
(Header のみです。)

３．２．35．３　DSH_EncodeS7F20() －　S7F20 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F20(
        BYTE *buffer,
        int buff_size,
        TPPID_LIST *list,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F20(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef list As TPPID_LIST,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F20(
        IntPtr buffer,
        int buff_size,
        ref TPPID_LIST list,
        ref int msg_len
       );
```

（2）引数

buffer	： S7F20 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
list	： TPPID_LIST 構造体です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F20 メッセージを作成します。
list に設定されている PPID をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
BYTE send_buffer[1000];
int ei;
int msg_len;

char* ppid_list[32] = { "PPID100", "PPID200", "PPID300" };

TPPID_LIST  list;

DshInitTPPID_LIST( &list, 3 );
DshPutTPPID_LIST( &list, ppid_list[0] );
DshPutTPPID_LIST( &list, ppid_list[1] );
DshPutTPPID_LIST( &list, ppid_list[2] );

ei = DSH_EncodeS7F20( send_buffer, 1000, &list, &msg_len );
.
.
DshFreeTPPID_LIST( &list );
```

②C#

```
TPPID_LIST  list = new TPPID_LIST();
string ppbody = "PPBODY100200300";

IntPtr send_buffer = Marshal. AllocCoTaskMem(1000);
int  msg_len = 0;

int PP_COUNT = 3;
string[] ppid_list = { "PPID100", "PPID200", "PPID300" };

DshInitTPPID_LIST(ref list, PP_COUNT);
for (int i = 0; i < PP_COUNT; i++)
{
    DshGemPro.LIB.DshPutTPPID_LIST(ref list, ppid_list[i]);
}
int ei = DSH_EncodeS7F20( send_buffer, 1000, ref list, ref msg_len );
.
.
DshFreeTPPID_LIST( ref list );
Marshal.FreeCoTaskMem(send_buff);
```

３．２．35．４　DSH_DecodeS7F20() －　S7F20 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F20(
        BYTE *buffer,
        int msg_len,
        TPPID_LIST *list
        );
```

[VB.Net]

```
Function DSH_DecodeS7F20(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef list As TPPID_LIST
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F20(
        IntPtr buffer,
        int msg_len,
        ref TPPID_LIST list
        );
```

（2）引数

buffer	：	S7F20 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S7F20 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
list	：	PPID を格納する構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F20 メッセージのデコードを行います。
デコードした PPID 情報は list に格納します。

３．２．36　S7F23 メッセージ　－　FPP 書式付プロセスプログラム送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS7F23()	S7F23 をｴﾝｺｰﾄﾞします。	FPP 送信ﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。
2	DSH_DecodeS7F23()	S7F23 をﾃﾞｺｰﾄﾞします。	FPP 送信ﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS7F24()	S7F24 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	確認情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS7F24()	S7F24 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	確認情報を取得します。

（2）S7F23 のユーザインタフェース情報
情報の引き渡しは、TS7F23_INFO 構造体を使って行います。

①FPP 情報を格納する構造体です。API の引数に使用します。

```
typedef struct{
        int         index;
        int         state;
        char        *ppid;
        char        *mdln;
        char        *softrev;
        int         ccode_count;        // # of process commands
        TFPP_CCODE  **ccode_list;
}TS7F23_INFO;                                   // formatted process program
```

②FPP に付属するセクション情報を格納する構造体です。

```
typedef struct{
        int         ccode_fmt;
        int         ccode_size;
        int         max_ccode_size;
        void        *ccode;                 // command code
        int         ppara_count;            // # of para count
        TFPP_PARA   **ppara_list;
} TFPP_CCODE
```

③FPP のセクション情報に付属する 1 個分のパラメータを格納する構造体です。

```
typedef struct{
        int         ppara_fmt;
        int         ppara_size;
        int         max_ppara_size;
        void        *ppara;
} TFPP_PARA;
```

（3）TS7F23_INFO 構造体への情報設定処理関連関数

C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。

.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTS7F23_INFO	TS7F23_INFO を初期設定する。
2	DshPutTS7F23_INFO	TS7F23_INFO に 1 個のﾊﾟﾗﾒｰﾀを加える。
3	DshFreeTS7F23_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。
4	DshInitTFPP_CCODE	TFPP_CCODE を初期設定する。
5	DshPutTFPP_CCODE	TFPP_CCODE に 1 個のﾊﾟﾗﾒｰﾀ情報を加える。
6	DshFreeTFPP_CCODE	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

（3）S7F24 のユーザインタフェース情報

情報の引き渡しはプロセスプログラム送信確認 ACK です。

３．２．36．１　DSH_EncodeS7F23() －　S7F23 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F23(
        BYTE *buffer,
        int buff_size,
        TS7F23_INFO *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F23(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef info As TS7F23_INFO,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F23(
        IntPtr buffer,
        int buff_size,
        ref TS7F23_INFO info,
        ref int msg_len
       );
```

（2）引数

buffer	：S7F23 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	：buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	：FPP 情報が格納されている構造体です。
msg_len	：ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F23 メッセージを作成します。
info に格納されている FPP 情報をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
char* PPID       = "PPID100";
char* MDLN       = "MDLN100";
char* SOFTREV    = "SOFT100";

char* CCODE1     = "CCODE_1";
char* PARA11     = "PARA_1";       // A
uint  PARA12     = 1234; // U4

char* CCODE2     = "CCODE_2";
char* PARA21     = "PARA_2";       // A

int   msg_len;
TS7F23_INFO info;
TFPP_CCODE  cinfo;

DshInitTS7F23_INFO( &info, PPID, MDLN, SOFTREV, 2 );

DshInitTFPP_CCODE( &cinfo, ICODE_A, CCODE1, 2 );
DshPutTFPP_CCODE( &cinfo, ICODE_A, (void*)PARA11 );
DshPutTFPP_CCODE( &cinfo, ICODE_U4, (void*)&PARA12 );
DshPutTS7F23_INFO( &info, &cinfo );
DshFreeTFPP_CCODE( &cinfo );

DshInitTFPP_CCODE( &cinfo, ICODE_A, CCODE2, 2 );
DshPutTFPP_CCODE( &cinfo, ICODE_A, (void*)PARA21 );
DshPutTFPP_CCODE( &cinfo, ICODE_U4, (void*)&PARA22 );
DshPutTS7F23_INFO( &info, &cinfo );
DshFreeTFPP_CCODE( &cinfo );

ei = DSH_EncodeS7F23( buff, SND_BUFF_SIZE, &info, &msg_len );
.
.

DshFreeTS7F23_INFO( &info );
```

②C#

```
string PPID = "PPID100";
string MDLN = "MDLN100";
string SOFTREV = "SOFT100";

string CCODE1 = "CCODE_1";
string PARA11 = "PARA_1";         // A
uint   PARA12 = 1234;             // U4

string CCODE2 = "CCODE_2";
string PARA21 = "PARA_2";         // A
```

```
uint   PARA22 = 2345;              // U4

int ei;
int msg_len = 0;
IntPtr buff = Marshal.AllocCoTaskMem(BUFF_SIZE);
TS7F23_INFO info = new TS7F23_INFO();
TFPP_CCODE cinfo = new TFPP_CCODE();

DshInitTS7F23_INFO( ref info, PPID, MDLN, SOFTREV, 2 );

DshInitTFPP_CCODE( ref cinfo, ICODE_A, CCODE1, 2 );
DshPutTFPP_CCODE(ref cinfo, ICODE_A, PARA11);
copy_int_to_ptr(int_ptr, (int)PARA12);
DshPutTFPP_CCODE(ref cinfo, ICODE_U4, int_ptr);
DshPutTS7F23_INFO( ref info, ref cinfo );
DshFreeTFPP_CCODE( ref cinfo );

DshInitTFPP_CCODE(ref cinfo, ICODE_A, CCODE2, 2);
DshPutTFPP_CCODE(ref cinfo, ICODE_A, PARA21);
copy_int_to_ptr(int_ptr, (int)PARA22);
DshPutTFPP_CCODE(ref cinfo, ICODE_U4, int_ptr);
DshPutTS7F23_INFO( ref info, ref cinfo );
DshFreeTFPP_CCODE( ref cinfo );

ei = DSH_EncodeS7F23(buff, BUFF_SIZE, ref info, ref msg_len);     // encode S7F23
.
.
DshFreeTS7F23_INFO(ref info);
Marshal.FreeCoTaskMem(buff);
```

(注) copy_int_to_ptr()については、3.2.2.3-(5)-②　を参照ください。

３．２．36．２　DSH_DecodeS7F23() －　S7F23 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F23(
        BYTE *buffer,
        int msg_len,
        TS7F23_INFO *info
        );
```

[VB.Net]

```
Function DSH_DecodeS7F23(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TS7F23_INFO,
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F23(
        IntPtr buffer,
        int msg_len,
        ref TS7F23_INFO info,
        );
```

（2）引数

buffer	：	S7F23 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S7F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
info	：	FPP 情報を格納する構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F23 メッセージのデコードを行います。
デコードした FPP 情報は　info　に格納します。

（5）例

①c、C++

```
BYTE buff[1000];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F23 受信)

int msg_len = 63;                   // 受信したS7F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TS7F23_INFO info;

int ei;
ei = DSH_DecodeS7F23( buff, msg_len, &info);
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(1000);
(S7F23 受信)

int msg_len = 63;                   // 受信したS7F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TS7F23_INFO info = new TS7F23_INFO();

int ei = DSH_DecodeS7F23( buff, msg_len, ref info);
.
.
DshFreeTS7F23_INFO( ref info );
Marshal.FreeCoTaskMem(buff);
```

３．２．36．３　DSH_EncodeS7F24() －　S7F24 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F24(
        BYTE *buffer,
        int buff_size,
        int ack,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F24(
        buffer As IntPtr,
        buff_size As Integer,
        ack As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F24(
        IntPtr buffer,
        int buff_size,
        int ack,
        ref int msg_len
       );
```

（2）引数

buffer	：S7F24 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	：buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ack	：PP 送信確認ｺｰﾄﾞです。
msg_len	：ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F24 メッセージを作成します。
ack を S7F24 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
int   ack = 0;
int   ei;
BYTE  buff[100;
int   msg_len;

ei = DSH_EncodeS7F24( buff, 100, ack, &msg_len );   //
.
.
```

②c#

```
int   ack = 0
int   ei;

IntPtr buff = Marshal. AllocCoTaskMem(1000);
int   msg_len = 0;

ei = DSH_EncodeS7F24( buff, 100, ack, ref msg_len );//
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．36．４　DSH_DecodeS7F24() －　S7F24 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F24(
        BYTE *buffer,
        int msg_len,
        int *ack
        );
```

[VB.Net]

```
Function DSH_DecodeS7F24(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ack As Integer
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F24(
        IntPtr buffer,
        int msg_len,
        ref int ack
        );
```

（2）引数

buffer	：	S7F24 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S7F24 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ack	：	PP 送信確認ｺｰﾄﾞ格納用です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F24 メッセージのデコードを行います。
デコード結果は、ack に格納されます。

３．２．37　S7F25 メッセージ　－　書式付プロセスプログラム送信

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS7F25()	S7F25 をｴﾝｺｰﾄﾞします。	FPP 要求ﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。 (PPID を指定)
2	DSH_DecodeS7F25()	S7F25 をﾃﾞｺｰﾄﾞします。	FPP 要求ﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS7F26()	S7F26 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	FPP 情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS7F26()	S7F26 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	FPP 情報を取得します。

（2）S7F25 のユーザインタフェース情報
情報の引き渡しは、PPID です。

（3）S7F26 のユーザインタフェース情報
情報の引き渡しは、TS7F23_INFO 構造体を使って行います。

①FPP 情報を格納する構造体です。API の引数に使用します。

```
typedef struct{
        int         index;
        int         state;
        char        *ppid;
        char        *mdln;
        char        *softrev;
        int         ccode_count;        // # of process commands
        TFPP_CCODE  **ccode_list;
}TS7F23_INFO;                                   // formatted process program
```

②FPP に付属するセクション情報を格納する構造体です。

```
typedef struct{
        int         ccode_fmt;
        int         ccode_size;
        int         max_ccode_size;
        void        *ccode;                 // command code
        int         ppara_count;            // # of para count
        TFPP_PARA   **ppara_list;
} TFPP_CCODE
```

③FPP のセクション情報に付属する 1 個分のパラメータを格納する構造体です。

```
typedef struct{
        int         ppara_fmt;
        int         ppara_size;
        int         max_ppara_size;
        void        *ppara;
} TFPP_PARA;
```

（4）TS7F23_INFO 構造体への情報設定処理関連関数

C/C++ 言語用ﾍﾀﾞｰﾌｧｲﾙは、DshGemProLib.h でﾌﾟﾛﾄﾀｲﾌﾟが定義されています。

.Net 言語では、DshGemProLib.cs, DshGemProLib.vb

番号	関数名	機能
1	DshInitTS7F23_INFO	TS7F23_INFO を初期設定する。
2	DshPutTS7F23_INFO	TS7F23_INFO に 1 個のﾊﾟﾗﾒｰﾀを加える。
3	DshFreeTS7F23_INFO	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。
4	DshInitTFPP_CCODE	TFPP_CCODE を初期設定する。
5	DshPutTFPP_CCODE	TFPP_CCODE に 1 個のﾊﾟﾗﾒｰﾀ情報を加える。
6	DshFreeTFPP_CCODE	使用後、構造体内で使用したﾋｰﾌﾟﾒﾓﾘを解放する。

３．２．37．１　DSH_EncodeS7F25() －　S7F25 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F25(
        BYTE *buffer,
        int buff_size,
        char *ppid,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F25(
        buffer As IntPtr,
        buff_size As Integer,
        ppid As String,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F25(
        IntPtr buffer,
        int buff_size,
        string ppid,
        ref int msg_len
       );
```

（2）引数

buffer	: S7F25 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	: buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ppid	: PPID(書式付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID)です。
msg_len	: ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F25 メッセージを作成します。
ppid をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
char *ppid = "PPID1000";

BYTE send_buffer[128];
int ei;
int msg_len;

ei = DSH_EncodeS7F25( send_buffer, 128, ppid, &msg_len );
.
.
```

②C#

```
string ppid =  "PPID1000";

IntPtr send_buffer = Marshal. AllocCoTaskMem(128);
int  msg_len = 0;
int ei = DSH_EncodeS7F25( send_buffer, 128, ppid, ref msg_len );
.
.
Marshal.FreeCoTaskMem(send_buff);
```

３．２．37．２　DSH_DecodeS7F25() －　S7F25 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F25(
        BYTE *buffer,
        int msg_len,
        char *ppid
        );
```

[VB.Net]

```
Function DSH_DecodeS7F25(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ppid As String
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F25(
        IntPtr buffer,
        int msg_len,
        ref IntPtr ppid
        );
```

（2）引数

buffer	: S7F25 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	: S7F25 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ppid	: ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID 格納です。(PPID に必要な十分の領域を準備してください)

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F25 メッセージのデコードを行います。
デコードした PPID は ppid に格納します。

（5）例

①c、C++

```
BYTE buff[1000];          // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F25 受信)
int msg_len = 17;                  // 受信した S7F25 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
char ppid[128];
int ei;
ei = DSH_DecodeS7F25( buff, msg_len, ppid );
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(1000);
(S7F25 受信)
int msg_len = 17;                  // 受信した S7F25 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
IntPtr ppid = Marshal.AllocCoTasmMem(128);

int ei = DSH_DecodeS7F25( buff, msg_len, ppid );
string pp_id = Marshal. PtrToStringAnsi(ppid);
.
.
Marshal.FreeCoTaskMem(ppid);
Marshal.FreeCoTaskMem(buff);
```

３．２．37．３　DSH_EncodeS7F26() －　S7F26 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F26(
        BYTE *buffer,
        int buff_size,
        TS7F23_INFO *info,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F26(
        buffer As IntPtr,
        buff_size As Integer,
        ByRef info As TS7F23_INFO,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F26(
        IntPtr buffer,
        int buff_size,
        ref TS7F23_INFO info,
        ref int msg_len
       );
```

（2）引数

buffer	：S7F26 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	：buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
info	：FPP 情報が格納されている構造体です。
msg_len	：ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F26 メッセージを作成します。
info に格納されている FPP 情報を S7F26 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
char* PPID       = "PPID100";
char* MDLN       = "MDLN100";
char* SOFTREV    = "SOFT100";

char* CCODE1     = "CCODE_1";
char* PARA11     = "PARA_1";       // A
uint  PARA12     = 1234; // U4

char* CCODE2     = "CCODE_2";
char* PARA21     = "PARA_2";       // A

int   msg_len;
TS7F23_INFO info;
TFPP_CCODE  cinfo;

DshInitTS7F23_INFO( &info, PPID, MDLN, SOFTREV, 2 );

DshInitTFPP_CCODE( &cinfo, ICODE_A, CCODE1, 2 );
DshPutTFPP_CCODE( &cinfo, ICODE_A, (void*)PARA11 );
DshPutTFPP_CCODE( &cinfo, ICODE_U4, (void*)&PARA12 );
DshPutTS7F23_INFO( &info, &cinfo );
DshFreeTFPP_CCODE( &cinfo );

DshInitTFPP_CCODE( &cinfo, ICODE_A, CCODE2, 2 );
DshPutTFPP_CCODE( &cinfo, ICODE_A, (void*)PARA21 );
DshPutTFPP_CCODE( &cinfo, ICODE_U4, (void*)&PARA22 );
DshPutTS7F23_INFO( &info, &cinfo );
DshFreeTFPP_CCODE( &cinfo );

ei = DSH_EncodeS7F26( buff, SND_BUFF_SIZE, &info, &msg_len );
.
.

DshFreeTS7F23_INFO( &info );
```

②c#

```
string PPID = "PPID100";
string MDLN = "MDLN100";
string SOFTREV = "SOFT100";

string CCODE1 = "CCODE_1";
string PARA11 = "PARA_1";         // A
uint   PARA12 = 1234;             // U4

string CCODE2 = "CCODE_2";
string PARA21 = "PARA_2";         // A
```

```
uint   PARA22 = 2345;              // U4

int ei;
int msg_len = 0;
IntPtr buff = Marshal.AllocCoTaskMem(BUFF_SIZE);
TS7F23_INFO info = new TS7F23_INFO();
TFPP_CCODE cinfo = new TFPP_CCODE();

DshInitTS7F23_INFO( ref info, PPID, MDLN, SOFTREV, 2 );

DshInitTFPP_CCODE( ref cinfo, ICODE_A, CCODE1, 2 );
DshPutTFPP_CCODE(ref cinfo, ICODE_A, PARA11);
copy_int_to_ptr(int_ptr, (int)PARA12);
DshPutTFPP_CCODE(ref cinfo, ICODE_U4, int_ptr);
DshPutTS7F23_INFO( ref info, ref cinfo );
DshFreeTFPP_CCODE( ref cinfo );

DshInitTFPP_CCODE(ref cinfo, ICODE_A, CCODE2, 2);
DshPutTFPP_CCODE(ref cinfo, ICODE_A, PARA21);
copy_int_to_ptr(int_ptr, (int)PARA22);
DshPutTFPP_CCODE(ref cinfo, ICODE_U4, int_ptr);
DshPutTS7F23_INFO( ref info, ref cinfo );
DshFreeTFPP_CCODE( ref cinfo );

ei = DSH_EncodeS7F26(buff, BUFF_SIZE, ref info, ref msg_len);     // encode S7F23
.
.
DshFreeTS7F23_INFO(ref info);
Marshal.FreeCoTaskMem(buff);
```

(注) copy_int_to_ptr()については、3.2.2.3-(5)-②　を参照ください。

３．２．37．４　DSH_DecodeS7F26() －　S7F26 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F26(
        BYTE *buffer,
        int msg_len,
        TS7F23_INFO *info
        );
```

[VB.Net]

```
Function DSH_DecodeS7F26(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef info As TS7F23_INFO
        ) As Integer
```

[C#]

```
int DSH_DecodeS7F26(
        IntPtr buffer,
        int msg_len,
        ref TS7F23_INFO info,
        ref IntPtr ppbody
        );
```

（2）引数

buffer　　：S7F26 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。

msg_len　　：S7F26 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。
(Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

info　　：FPP 情報を格納するための構造体です。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F26 メッセージのデコードを行います。
デコードした PPID、PPBODY はそれぞれ、ppid, ppbody に格納します。

（5）例

①c、C++

```
BYTE buff[1000];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F23 受信)

int msg_len = 63;                   // 受信したS7F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TS7F23_INFO info;

int ei;
ei = DSH_DecodeS7F23( buff, msg_len, &info);
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(1000);
(S7F23 受信)

int msg_len = 63;                   // 受信したS7F23 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
TS7F23_INFO info = new TS7F23_INFO();

int ei = DSH_DecodeS7F23( buff, msg_len, ref info);
.
.
DshFreeTS7F23_INFO( ref info );
Marshal.FreeCoTaskMem(buff);
```

３．２．38　S7F29 メッセージ　－　プロセスプログラム妥当性の問合せ

（1）下表に示す４種類の関数があります。

	関数名	機　能	備　考
1	DSH_EncodeS7F29()	S7F29 をｴﾝｺｰﾄﾞします。	PP 妥当性問合せﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。
2	DSH_DecodeS7F29()	S7F29 をﾃﾞｺｰﾄﾞします。	PP 妥当性問合せﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。
3	DSH_EncodeS7F30()	S7F30 のﾒｯｾｰｼﾞをｴﾝｺｰﾄﾞします。	妥当性許可情報をｴﾝｺｰﾄﾞします。
4	DSH_DecodeS7F30()	S7F30 のﾒｯｾｰｼﾞをﾃﾞｺｰﾄﾞします。	妥当性許可情報を取得します。

（2）S7F29 のユーザインタフェース情報
　　情報の引き渡しは、LENGTH 長さ情報です。

（3）S7F30 のユーザインタフェース情報
　　情報の引き渡しは PPGNT 妥当性コードです。(=ACK の意味)

３．２．38．１　DSH_EncodeS7F29() －　S7F29 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F29(
        BYTE *buffer,
        int buff_size,
        int length,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F29(
        buffer As IntPtr,
        buff_size As Integer,
        length As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F29(
        IntPtr buffer,
        int buff_size,
        int length,
        ref int msg_len
       );
```

（2）引数

buffer	： S7F29 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
length	： 妥当性確認 LENGTH です。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F29 メッセージを作成します。
length をメッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

（5）例

①C/C++

```
int length = 64;

BYTE buff[100];
int ei;
int msg_len;

ei = DSH_EncodeS7F29( buff, 100, length, &msg_len );
.
.
```

②C#

```
string ppid =  “PPID1000”;
int length = 64;

IntPtr buff = Marshal. AllocCoTaskMem(100);
int  msg_len = 0;
int ei = DSH_EncodeS7F29( buff, 100, length, ref msg_len );
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．38．２　DSH_DecodeS7F29() －　S7F29 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F29(
        BYTE *buffer,
        int msg_len,
        int *length
        );
```

[VB.Net]

```
Function DSH_DecodeS7F29(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef length As Integer
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F29(
        IntPtr buffer,
        int msg_len,
        ref int length
        );
```

（2）引数

buffer	：	S7F29 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	：	S7F29 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
length	：	妥当性を問い合わせる LENGTH を格納する領域のﾎﾟｲﾝﾀです。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F29 メッセージのデコードを行います。
デコードした LENGTH 値は length に格納します。

（5）例

①c、C++

```
BYTE buff[1000];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F29 受信)
int msg_len = 6;                    // 受信した S7F29 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
int length;
int ei;

ei = DSH_DecodeS7F29( buff, msg_len, ppid, &length );
.
.
```

②C#

```
IntPtr buff = Marshal. AllocCoTaskMem(1000);
(S7F29 受信)
int msg_len = 6;                    // 受信した S7F29 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ
uint length = 0;

int ei = DSH_DecodeS7F29( buff, msg_len, ref length);
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．38．３　DSH_EncodeS7F30() －　S7F30 のエンコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_EncodeS7F30(
        BYTE *buffer,
        int buff_size,
        int ppgnt,
        int *msg_len
        );
```

[VB.Net]

```
Function DSH_EncodeS7F30(
        buffer As IntPtr,
        buff_size As Integer,
        ppgnt As Integer,
        ByRef msg_len As Integer
        ) As Integer
```

[C#]

```
int DSH_EncodeS7F30(
        IntPtr buffer,
        int buff_size,
        int ppgnt,
        ref int msg_len
       );
```

（2）引数

buffer	： S7F30 ﾒｯｾｰｼﾞﾃﾞｰﾀ格納用ﾒﾓﾘのﾎﾟｲﾝﾀです。
buff_size	： buffer で示すﾒﾓﾘのﾊﾞｲﾄｻｲｽﾞを指定します。
ppgnt	： ﾛｰﾄﾞ許可ｺｰﾄﾞです。
msg_len	： ｴﾝｺｰﾄﾞしたﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞを格納します。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧのｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに S7F30 メッセージを作成します。
ppgnt を S7F30 メッセージにエンコードします。

作成したメッセージのバイトサイズを msg_len に設定し、返却します。
作成したメッセージのバイトサイズが　buff_size 以内であれば、０を返却します。
もし、メッセージが buff_size に入りきらなかった場合は、(-1)を返却します。

(5) 例

①C/C++

```
int   ppgnt = 0;
int   ei;
BYTE  buff[1000;
int   msg_len;

ei = DSH_EncodeS7F30( buff, 100, ppgnt, &msg_len ); //
.
.
```

②c#

```
int   ppgnt = 0
int   ei;
int   msg_len = 0;

ei = DSH_EncodeS7F30( buff, 1000, ppgnt, ref msg_len );      //
.
.
Marshal.FreeCoTaskMem(buff);
```

３．２．38．４　DSH_DecodeS7F30() －　S7F30 のデコード

（1）呼出書式

[C/C++]

```
API int  APIX DSH_DecodeS7F30(
        BYTE *buffer,
        int msg_len,
        int *ppgnt
        );
```

[VB.Net]

```
Function DSH_DecodeS7F30(
        buffer As IntPtr,
        msg_len As Integer,
        ByRef ppgnt As Integer
       ) As Integer
```

[C#]

```
int DSH_DecodeS7F30(
        IntPtr buffer,
        int msg_len,
        ref int ppgnt
        );
```

（2）引数

buffer	: S7F30 ﾒｯｾｰｼﾞﾃﾞｰﾀが格納されているﾒﾓﾘのﾎﾟｲﾝﾀです。
msg_len	: S7F30 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞです。 (Header を含む場合は Header + Text の合計ｻｲｽﾞになります。)
ppgnt	: ﾛｰﾄﾞ許可ｺｰﾄﾞです。

（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

（4）説明

buffer で指定されたバッファに格納されている S7F30 メッセージのデコードを行います。
デコード結果は、ppgnt に格納されます。

（5）例

①ｃ、C++

```
BYTE buffer[100];                    // ここにﾃﾞｺｰﾄﾞ対象のﾒｯｾｰｼﾞが格納されているとします。
(S7F30 受信)
int msg_len = 3;                     // 受信した S7F30 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

int ppgnt;
int ei;

ei = DSH_DecodeS7F30( buffer, msg_len, &ppgnt );
.
.
```

②c＃

```
IntPtr buff = Marshal. AllocCoTaskMem(2000);
(S7F30 受信)
int msg_len = 3;                     // 受信した S7F30 ﾒｯｾｰｼﾞのﾊﾞｲﾄｻｲｽﾞ

int ppgnt =0 ;

int ei = DSH_DecodeS7F30( buff, msg_len, ref ppgnt );
.
.
Marshal.FreeCoTaskMem(buff);
```