DSHGEM-LIB　通信エンジンライブラリ(GEM+GEM300)

ソフトウェア・パッケージ

APP インタフェース
ライブラリ関数説明書

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

VOL-６／１５

３．11　PP プロセスプログラム情報アクセス関数

３．12　書式付プロセスプログラム情報アクセス関数

２００９年6月

株式会社データマップ

文書番号　DSHGEM-LIB-09-30326-02

[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】

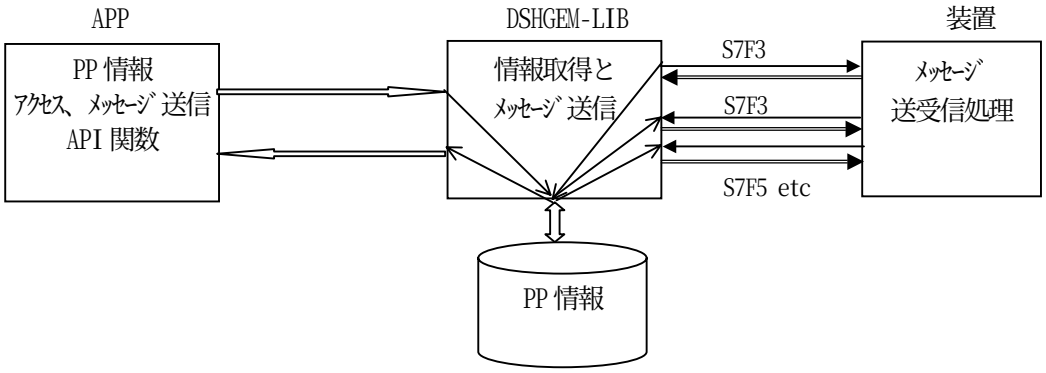
番号	改訂日付	項 目	概 略
1.	2009.6	改訂版	以前の DSHGEM-LIB-07-3032x-00 を全面改訂 .Net VB2008, C#2008 対応関数の説明を追加した。

目　次

(VOL－７に続く)

3．11　PP プロセスプログラム情報アクセスサービス関数

プロセスプログラム情報は、DSHGEM-LIB が管理します。従って、APP はこれらの情報をアクセスと関連メッセージを送信するために以下の DSHGEM-LIB API 関数を使用します。



（1）情報アクセスと送信 API 関数

プロセスプログラム情報のアクセスと装置へのメッセージ送信に関連するサービスのための API 関数名は一覧表のとおりです。

	API 関数名	機能
1	GemAllocPpInfo()	PP 情報領域を割当て登録します。
2	GemSetPpInfo()	PP ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を設定・変更します。
3	GemSetPpInfoX()	ｲﾝﾃﾞｸｽ指定ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を設定・変更します。
4	GemGetPpInfo()	PPID 指定でﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を取得します。
5	GemGetPpInfoX()	PPID 情報ｲﾝﾃﾞｸｽ指定でﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を取得します。
6	GemDelPpInfo()	PPID 指定でﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を削除します。
7	GemDelPpInfoX()	PPID 情報ｲﾝﾃﾞｸｽ指定でﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を削除します。
8	GemGetPpId()	指定した PPID 情報ｲﾝﾃﾞｸｽの PPID を取得します。
9	GemGetPpIdIndex()	指定した PPID の情報ｲﾝﾃﾞｸｽを取得します。
10	GemSetPpState()	PPID 指定で PP の状態を設定します。
11	GemSetPpStateX()	PP 情報ｲﾝﾃﾞｸｽ指定で PPID の状態を設定します。
12	GemGetPpState()	PP 指定で PP の状態を取得します。
13	GemGetPpStateX()	PP 情報ｲﾝﾃﾞｸｽ指定で PPID の状態を取得します。
14	GemGetPpList()	PPID の一覧ﾘｽﾄを取得します。
15	DshInitPpInfo	PP 情報構造体 TPP_INFO を初期設定します。
16	GemSendS7F1 ()	S7F1 PP 情報 LOAD 問合せを装置に送信します。
17	GemSendS7F3()	S7F3 PP 情報を装置に送信します。
18	GemSendS7F5()	S7F5 PP 情報を装置から取得します。
19	GemSendS7F17()	S7F17 PP 情報を装置から削除します。
20	GemSendS7F19()	S7F19 PP 情報一覧を装置から削除取得します。

PPID インデクスは、DSHGEM-LIB が管理する各 PPID の領域番号です。このインデクスの値は、GemAllocPpInfo()関数実行時に DSHGEM-LIB によって割当てられ、APP に渡されます。また、PPID 情報の取得時に、情報格納構造体のメンバーの index に設定されます。

（2）ライブラリ関数

他にAPPが使用できるプロセスプログラム情報処理用API関数として、以下の関数があります。

	API関数名	機能
1	DshDecodeS7F3()	S7F3のﾒｯｾｰｼﾞ内PP情報をTPP_INFO構造体にﾃﾞｺｰﾄﾞ収納するための関数です。
2	DshEncodeS7F3()	TPP_INFO構造体のﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報をS7F3ﾒｯｾｰｼﾞにｴﾝｺｰﾄﾞするための関数です。
3	DshFreeTPP_INFO()	PP情報が格納されているTPP_INFO構造体とその内部で使用されているﾒﾓﾘを開放するための関数です。
4	DshCopyTPP_INFO()	TPP_INFOのﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報をｺﾋﾟｰします。
5	DshInitPpInfo()	PP情報構造体TPP_INFOを初期設定します。
6	DshMakeS7F3Response()	S7F4応答ﾒｯｾｰｼﾞを生成し、送信するための関数。u_s7f3.cで使用します。
7	DshDecodeS7F17()	S7F17を削除ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ一覧情報構造体TPPID_LIST内にデコードします。
8	DshDecodeS7F20()	S7F20をﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ一覧情報構造体TPPID_LIST内にデコードします。
9	DshFreeTPPID_LIST()	TPPID_LIST構造体内で使用されているﾒﾓﾘを開放します。
10	DshDecodeS7F27()	S7F27ﾒｯｾｰｼﾞ内のﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ妥当性情報をTPP_PVS_LIST構造体内にﾃﾞｺｰﾄﾞします。
11	DshFreeTPP_PVS_LIST()	TPP_PVS_LIST構造体内で使用されているﾒﾓﾘを開放します。

（3）ユーザ作成ライブラリ関数

	ﾗｲﾌﾞﾗﾘ関数名	機能
1	DshResponseS7F2()	S7F2 プロセスプログラムロード問合せ応答
2	DshResponseS7F4()	S7F4 プロセスプログラム送信応答メッセージ
3	DshResponseS7F6()	S7F6 プロセスプログラム要求応答メッセージ
4	DshResponseS7F18()	S7F18 プロセスプログラム削除指示応答メッセージ

3．11．１　使用する情報格納構造体

（1）プロセスプログラム情報を操作する関数は、共通の情報格納のための TPP_INFO 構造体を使用します。

```
typedef struct{
        int     index;                  // 情報格納場所ｲﾝﾃﾞｸｽ
        int     state;                  // PP の状態(APP が任意に決める)
        char    *name;                  // pp 名(装置管理情報定義ﾌｧｲﾙで与えられた名前)
        int     ppid_fmt;               // ppid のｱｲﾃﾑﾌｫｰﾏｯﾄ( Format 10 ASCII が設定されます。)
        int     ppid_size;              // 同配列ｻｲｽﾞ(実際の ppid のﾊﾞｲﾄ長です。)
        int     max_ppid_size;          // 配列ｻｲｽﾞの最大値
        char    *ppid;                  // ppid 値格納ﾎﾟｲﾝﾀ
        int     ppbody_fmt;             // ppbody のｱｲﾃﾑﾌｫｰﾏｯﾄ( Format 10 ASCII が設定されます。)
        int     ppbody_size;            // 同配列ｻｲｽﾞ(実際の ppbody のﾊﾞｲﾄ長です。)
        }TPP_INFO;                      // formatted process program
```

（2）PPID(プロセスプログラム ID)リスト格納用構造体 － S7F19 メッセージの応答メッセージデコード情報用

```
typedef struct{
        int         count;              // ppid_list に含まれる PPID の数
        char        **ppid_list;        // ppid が格納されているﾘｽﾄ
        } TPPID_LIST;
```

（3）PP (プロセスプログラム)妥当性情報 － 装置からの S7F27 メッセージデコード情報用

```
typedef struct{
        char        *ppid;              // ppid
        int         err_count;          // 妥当性で検出されたｴﾗｰ情報の数
        TPP_PVS_INFO **err_list;        // ｴﾗｰ情報のﾘｽﾄ
        } TPP_PVS_LIST;

typedef struct{
        int         ackc7a;             // ack
        int         seqnum;             // 番号
        char        *errw7;             // ｴﾗｰ情報(文字列)
        } TPP_PVS_INFO;
```

3．11．２　PP プロセスプログラム情報アクセス関数

3．11．２．１　GemAllocPpInfo() - プロセスプログラムの登録

（1）呼出書式

[c,C++]

```
API int APIX GemAllocPpInfo(
        int   eqid,                 // 通信対象装置 ID(0,1,2,...)
        char *ppid,                 // PPID 格納領域のﾎﾟｲﾝﾀ
        int   *index                // 得られた情報領域ｲﾝﾃﾞｸｽ格納用ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemAllocPpInfo (
          ByVal eqid As Int32,
          ByVal ppid As String,
          ByRef index As Int32) As Int32
```

[.NET C#]

```
int GemAllocPpInfo(
          int eqid,
          byte[]   ppid,
          ref int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

登録したいプロセスプログラムの ID が格納されているポインタです。

index

登録された情報領域のインデクス値が格納される領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に登録できた。
1	指定された PPID は既に登録されていた。
(-1)	登録できなかった。

（4）説明

プロセスプログラムを新規にシステムに登録するための関数です。
引数 ppid で与えられるプロセスプログラム ID をシステムに登録します。

正常に登録できた場合は、index で指定される領域に登録された情報領域のインデクスが設定返却されます。

もし、ppid に指定されたプロセスプログラムが既に登録済みであった場合には関数の戻り値 1 を返却します。index には既に登録されている情報領域のインデクスが設定されます。
得られたインデクスを使って、情報の設定、取得、削除などのアクセスを行うことができます。

３．11．２．２　GemSetPpInfo()　- プロセスプログラム情報の設定
.　　GemSetPpInfoX() - インデクス指定によるプロセスプログラム情報の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetPpInfo(
        int       eqid,              // 通信対象装置 ID(0,1,2,...)
        TPP_INFO *pinfo              // PP 情報格納構造体のﾎﾟｲﾝﾀ
);

API int APIX GemSetPpInfoX(
        int       eqid,              // 通信対象装置 ID(0,1,2,...)
        int       index,             // GemAllocPpInfo()で得られたｲﾝﾃﾞｸｽ値
        TPP_INFO *pinfo              // PP 情報格納構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemSetPpInfo (
        ByVal eqid As Int32,
        ByRef pinfo As dsh_info.TPP_INFO) As Int32

Function GemSetPpInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef pinfo As dsh_info.TPP_INFO) As Int32
```

[.NET C#]

```
int GemSetPpInfo(
        int eqid,
        ref TPP_INFO pinfo );

int GemSetPpInfoX(
        int eqid,
        int index,
        ref TPP_INFO pinfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

pinfo

設定したいプロセスプログラム情報が格納されている格納構造体領域のポインタです。

index

PPID 情報のインデクスです。登録時に GemAllocPpInfo()関数によって与えられます。
インデクスは、PPID から GemGetPpIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に設定できた。

(-1)	設定できなかった。

（4）説明

本関数は、pinfo に格納されているプロセスプログラム情報の設定・変更に使用します。index 値の指定によっても設定できます。

引数 pinfo 内の ppid メンバーに指定されるプロセスプログラム ID の情報として設定されます。

pinfo 内には PPID の他、 PPBODY 情報が含まれます。
指定した PPID が既に登録済みである場合、pinfo 内の情報にすべて書き換えられます。

pinfo 内の PPID が未登録であった場合は、登録手続きをしてから PP 情報を設定します。
(GemAllocPpInfo()関数で行われる登録と同じ登録が行われます。)

３．11．２．３　GemGetPpInfo()　- プロセスプログラムの取得
.　　GemGetPpInfoX() - インデクス指定でのプロセスプログラムの取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetPpInfo(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        char        *ppid,                  // PPID が格納されている領域のポインタ
        TPP_INFO    *pinfo                  // PP 情報を格納する構造体ポインタ
);

API int APIX GemGetPpInfoX(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        int         index,                  // PPID 情報のインデクス
        TPP_INFO    *pinfo                  // PP 情報を格納する構造体ポインタ
);
```

[.NET VB]

```
Function GemGetPpInfo (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByRef pinfo As dsh_info.TPP_INFO) As Int32

Function GemGetPpInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef pinfo As dsh_info.TPP_INFO) As Int32
```

[.NET C#]

```
int GemGetPpInfo(
        int eqid,
        byte []ppid,
        ref TPP_INFO pinfo );

int GemGetPpInfoX(
        int eqid,
        int index,
        ref TPP_INFO pinfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

PPID が格納されている領域のポインタです。

pinfo

取得した PP 情報を格納する構造体領域のポインタです。

index

PPID 情報のインデクスです。登録時に GemAllocPpInfo()関数によって与えられます。

インデクスは、PPID から GemGetPpIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。(未登録であった)

（4）説明

ppid または index に指定されている PPID のプロセスプログラム情報を ppinfo で指定されたメモリ領域に取得格納します。

TPP_INFO 構造体の中に情報を格納するために必要なメモリは、DSHGEM-LIB が準備確保します。即ち、構造体のメンバーの中でポインタになっている情報の実体即ち、ppid,ppbody のためのメモリは DSHGEM-LIB が準備します。

これらのメモリは、使用後、ユーザが DSHGEM-LIB の API 関数を使って次のように開放してください。

```
TPP_INFO    *pinfo;

    if ( GemGetPpInfo( ppidinfo ) == 0 ){
        pinfo の処理
        処理終了後
        DshFreeTPP_INFO( pinfo );          // pinfo 内に使用されているﾒﾓﾘの開放
     }
```

３．11．２．４　GemDelPpInfo()　- プロセスプログラムの削除
.　　GemDelPpInfoX() - インデクスでのプロセスプログラムの削除

（1）呼出書式

[c,C++]

```
API int APIX GemDelPpInfo(
        int       eqid,              // 通信対象装置ID(0,1,2,...)
        char      *ppid;             // PPIDが格納されている領域のポインタ
);

API int APIX GemDelPpInfoX(
        int       eqid,              // 通信対象装置ID(0,1,2,...)
        int       index              // ｲﾝﾃﾞｸｽ(0,1,2....)
);
```

[.NET VB]

```
Function GemDelPpInfo (
        ByVal eqid As Int32,
        ByVal ppid As String) As Int32

Function GemDelPpInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32) As Int32
```

[.NET C#]

```
int GemDelPpInfo(
        int eqid,
        byte[] pppid );

int GemDelPpInfoX(
        int eqid,
        int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

ppid

PPIDが格納されている領域のポインタです。

index

PPID情報のインデクスです。登録時にGemAllocPpInfo()関数によって与えられます。
インデクスは、PPIDからGemGetPpIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に削除できた。
(-1)	未登録であった。

（4）説明

ppid または index に指定されている PPID または、index 指定されたプロセスプログラム情報をシステムの登録から削除します。

３．11．２．５　GemSetPpState() - プロセスプログラム状態の設定
.　　GemSetPpStateX() - インデクスでのプロセスプログラム状態の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetPpState(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *ppid,                // PPID が格納されている領域のポインタ
        int       state                 // 設定したい状態値
);

API int APIX GemSetPpStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // PPID 情報のインデクス
        int       state                 // 設定したい状態値
);
```

[.NET VB]

```
Function GemSetPpState (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByVal state As Int32) As Int32

Function GemSetPpStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal state As Int32) As Int32
```

[.NET C#]

```
int GemSetPpState(
        int eqid,
        byte[] ppid,
        int state);

int GemSetPpStateX(
        int eqid,
        int index,
        int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

PPID が格納されている領域のポインタです。

state

設定したい PPID の状態値です。
値=(-1)は使用不可を意味する値になります。

index

PPID 情報のインデクスです。登録時に GemAllocPpInfo()関数によって与えられます。
インデクスは、PPID から GemGetPpIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に設定できた。
(-1)	PPID が未登録であった。

（4）説明

プロセスプログラム情報の状態値を設定します。
状態値の意味合いは値=(-1)以外についてはユーザが定義して使用します。

３．11．２．６　GemGetPpState() － プロセスプログラム状態の取得
.　　GemGetPpStateX() － インデクスでのプロセスプログラム状態の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetPpState(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *ppid,                // PPID が格納されている領域のポインタ
        int       *state                // 取得した状態値格納用領域ポインタ
);

API int APIX GemGetPpStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // PPID 情報のインデクス
        int       *state                // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetPpState (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByRef state As Int32) As Int32

Function GemGetPpStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef state As Int32) As Int32
```

[.NET C#]

```
int GemGetPpState(
        int eqid,
        byte[] ppid,
        ref int state);

int GemGetPpStateX(
        int eqid,
        int index,
        ref int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

PPID が格納されている領域のポインタです。

state

取得した PPID の状態値を格納する領域のポインタです。

index

PPID 情報のインデクスです。登録時に GemAllocPpInfo()関数によって与えられます。

インデクスは、PPID から GemGetPpIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	PPID が未登録であった。

（4）説明

プロセスプログラム情報の状態値を取得します。

状態値の意味合いは値=(-1)以外についてはユーザが定義して使用します。

3．11．2．7　GemGetPpList() - 全登録 PPID 取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetPpList(
        int          eqid,                  // 通信対象装置 ID(0,1,2,...)
        TTEXT_DLIST **list                  // 取得ﾘｽﾄ格納ﾎﾟｲﾝﾀの格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetPpList (
        ByVal eqid As Int32,
        ByRef list As IntPtr) As Int32
```

[.NET C#]

```
int GemGetPpList(
        int eqid,
        IntPtr list );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

list

取得できた PPID が格納されている TTEXT_DLIST 構造体のポインタを格納する領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

（4）説明

システムに登録されている PPID とその名前を TTEXT_DLIST 構造体に取出すための関数です。
取出す名前は、装置管理情報定義ファイルで PPID 定義時に与えられた名前です。
取得した情報の処理が終了した後、DshFreeTTEXT_DLIST()関数で list 内部の情報格納用に使用されているメモリを開放してください。

TTEXT_DLIST 構造体は次のとおりです。

```
typedef struct{
        int         count;              // 取得できた ID 数
        char        **id_list;          // 取得できた ID 格納用配列
        char        **name_list;        // 取得できた名前格納ポインタ配列
        }TTEXT_DLIST;
```

3．11．2．8　GemGetPpId() - インデクスから PPID（プロセスプログラム ID）の取得

（1）呼出書式

[c,C++]

```
API int  APIX GemGetPpId(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        int     index,                  // PP 情報のインデクス
        char    *ppid                   // 取得した PPID を格納する領域のポインタ
);
```

[.NET VB]

```
Function GemGetPpId (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal ppid As String) As Int32
```

[.NET C#]

```
int GemGetPpId(
        int eqid,
        int index,
        byte[] ppid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

index

PP 情報のインデクスです。登録時に GemAllocPpInfo()関数によって与えられます。
インデクスは、PPID から GemGetPpIdIndex()関数で取得することができます。

ppid

PPID を格納するための領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。（未登録）

（4）説明

PP 情報のインデクスから PPID（プロセスプログラム ID）を取得し、ppid に格納します。
正常に取得できた場合は関数戻り値として 0 が返却されます。

3．11．２．９　GemGetPpIdIndex() -PPID（プロセスプログラム ID）からインデクスの取得

（1）呼出書式

[c,C++]

```
API int  APIX GemGetPpIdIndex(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        char    *ppid,                  // PPID が格納されている領域のポインタ
        int     *index                  // 取得したインデクスを格納するための領域のポインタ
);
```

[.NET VB]

```
Function GemGetPpIdIndex (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByRef index As Int32) As Int32
```

[.NET C#]

```
int GemGetPpIdIndex(
        int eqid,
        byte[] ppid,
        ref int index);
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

インデクスを取得したい対象の PPID が格納されている領域のポインタです。

index

取得したインデクスの値を格納するための領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。（未登録）

（4）説明

ppid に指定される PPID（プロセスプログラム ID）から PP 情報インデクスを取得するための関数です。
取得されたインデクスは index で指定された領域に格納されます。
正常に取得できた場合は関数戻り値として 0 が返却されます。

3．11．3　PP プロセスプログラム関連通信メッセージ送信要求関数

3．11．3．1　GemSendS7F1() - プロセスプログラムロード問合せメッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS7F1(
        int   eqid,                     // 通信対象装置 ID(0,1,2,...)
        char *ppid,                     // PPID 格納領域のﾎﾟｲﾝﾀ
        int   length,                   // ﾛｰﾄﾞ問合せするﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑのﾊﾞｲﾄ長
        int   *ppgnt,                   // S7F2 ﾒｯｾｰｼﾞの PPGNT 格納領域ﾎﾟｲﾝﾀ
        int   (WINAPI *callback)(),     // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                     // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS7F1 (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByVal length As Int32,
        ByRef ackc7 As Int32,
        ByVal callback As vcallback.callback_S7F1,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS7F1(
        int eqid,
        byte[] ppid,
        uint length,
        ref int ackc7,
        CallbackS7F1 callback,
        uint upara );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

ロードしたいプロセスプログラム ID が格納されているポインタです。

length

ロードしたいプロセスプログラムのバイト長です。

ppgnt

問合せ結果 PPGNT を格納する領域のポインタです。

callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバ

ックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。ppgnt に PPGNT 結果が返却される。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

（4）説明

装置にプロセスプログラム情報のロードを問い合わせます。
ppid に指定された PPID と length で与えられるプロセスプログラム長を S7F1 メッセージに設定し、装置に送信します。
通信が正常に行われた場合、受信した S7F2 応答メッセージ内の PPGNT を ppgnt に格納し、返却します。
送信要求から S7F2 応答メッセージ受信までの制御は引数の S7F1 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S7F1 送信後、応答ﾒｯｾｰｼﾞ S7F2 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、ppgnt に S7F2 ﾒｯｾｰｼﾞ内の PPGNT 値が渡されます。
あり	送信要求後、S7F1 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常であり、引数 ppgnt に S7F2 ﾒｯｾｰｼﾞ内の PPGNT 値が渡されます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

（5）コールバック関数

[c,C++]

```
API int  APIX callback(
        int    eqid,                  // 装置 ID
        int    end_status,            // 実行結果
        int    *ppgnt,                // PPGNT が格納されているﾒﾓﾘのﾎﾟｲﾝﾀ
        ULONG upara                   // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S7F1(ByVal eqid As Integer, ByVal end_status As Integer, ByRef ackc7 As Integer,
ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS7F1(int eqid, int status, int* ack, uint upara);
```

end_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。ppgnt に応答ﾒｯｾｰｼﾞの PPGNT 値が返却される。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．11．3．2　GemSendS7F3() - プロセスプログラム情報メッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS7F3(
        int   eqid,                         // 通信対象装置 ID(0,1,2,...)
        char *ppid,                         // PPID 格納領域のﾎﾟｲﾝﾀ
        void *ppbody,                       // ppbody ﾃﾞｰﾀ格納ﾎﾟｲﾝﾀ
        int   format,                       // ppbody ﾃﾞｰﾀのﾌｫｰﾏｯﾄ( ICODE_??)
        int   asize,                        // ppbody ﾃﾞｰﾀの配列ｻｲｽﾞ
        int   ackc7,                        // S7F4 応答 ACKC7
        int (WINAPI *s7f3Callback)(),       // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                         // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS7F3 (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByRef ackc7 As Int32,
        ByVal callback As vcallback.callback_S7F3,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS7F3(
        int eqid,
        byte[] ppid,
        ref int ackc7,
        CallbackS7F3 callback,
        uint upara );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

送信したいプロセスプログラム ID が格納されているポインタです。

ppbody

送信したいプロセスプログラム PPBODY データが格納されているポインタです。

format

ppbody データのフォーマットです。DSHDR2 ドライバーで使用するデータアイテムのフォーマットです。

asize

ppbody データの配列サイズです。(ICODE_A)の場合は文字列バイト長になります。）

ackc7

受信した応答メッセージ S7F4 の ACKC7 返却用メモリのポインタです。

s7f3Callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。ユーザは任意の関数名を指定できます。

コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。 ackc7 に ACKC7 が返却されます。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

（4）説明

プロセスプログラム情報の装置への送信要求を行います。S7F3 メッセージで送信します。
送信対象プロセスプログラム情報の指定はプロセスプログラム ID ならびに PPBODY データで行います。
PPBODY についてはフォーマットと配列サイズを指定する必要があります。

送信要求から S7F4 応答メッセージ受信までの制御は引数の S7F3 コールバック関数指定の有無によって次のようになります。

s7f3Callback の指定	制御の流れ
なし (=0)	S7F3 送信後、応答ﾒｯｾｰｼﾞ S7F4 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、ackc7 に ACKC7 が渡されます。
あり	送信要求後、S7F3 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で引数 ackc7 に ACKC7 が渡されます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

（5）コールバック関数

[c,C++]

```
API   int   APIX s7f3Callback(
        int   eqid,                 // 装置 ID
        int   end_status,           // 実行結果
        int   *ackc7,               // 応答 ACKC7 が格納されているﾎﾟｲﾝﾀ
        ULONG upara                 // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S7F3(ByVal eqid As Integer, ByVal end_status As Integer, ByRef ackc7 As Integer,
ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS7F3(int eqid, int end_status, int* ack, uint upara);
```

end_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。ackc7 に S7F4 の ACKC7 が設定されます。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．11．3．3　GemSendS7F5() - プロセスプログラム情報問合せメッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS7F5(
        int  eqid,                          // 通信対象装置 ID(0,1,2,...)
        char *ppid,                         // PPID 格納領域のﾎﾟｲﾝﾀ
        TPP_INFO *ppinfo,                   // S7F6 応答ﾒｯｾｰｼﾞのﾃﾞｺｰﾄﾞ結果格納ﾎﾟｲﾝﾀ
        int (WINAPI *s7f5Callback)(),       // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                         // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS7F5 (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByRef ppinfo As dsh_info.TPP_INFO,
        ByVal callback As vcallback.callback_S7F5,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS7F5(
        int eqid,
        byte[] ppid,
        ref TPP_INFO ppinfo,
        CallbackS7F5 callback,
        uint upara );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

装置に問合せたいプロセスプログラム ID が格納されているポインタです。

ppinfo

受信した応答メッセージ S7F6 をデコードした PP 情報を格納する TPP_INFO 構造体の指定します。

s7f5Callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。 ppinfo に S7F6 で得られた PP 情報が格納され、返却されます。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。

(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

（4）説明

S7F5 メッセージを使って装置にプロセスプログラム情報の問合せ要求を行います。問合せは ppid で指定されるプロセスプログラム ID で行います。要求を受けた DSHGEM-LIB は、S7F5 メッセージにエンコードし、装置に送信します。

送信要求から S7F6 応答メッセージ受信までの制御は引数の S7F5 コールバック関数指定の有無によって次のようになります。

s7f5Callback の指定	制御の流れ
なし (=0)	S7F5 送信後、応答ﾒｯｾｰｼﾞ S7F6 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、rmsg に S7F6 ﾒｯｾｰｼﾞ情報格納領域のﾎﾟｲﾝﾀが渡されます。
あり	送信要求後、S7F5 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で引数 rmsg に S7F6 ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀが与えられます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

DSHGEM-LIB は S7F6 で得られた情報を装置管理情報としての登録は行いません。必要に応じてユーザ側プログラムで登録処理をしてください。

正常に応答メッセージを受信した場合、PP 情報構造体のポインタが ppinfo に渡されますが、ユーザ側での処理を終えた後、その構造体に使用されているメモリを解放してください。

解放は次のように DshFreeTPP_INFO()関数を使って行ってください。

```
DshFreeTPP_INFO(ppinfo);
```

（5）コールバック関数

[c,C++]

```
API int  APIX s7f5Callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TPP_INFO *ppinfo,               // 問合せ結果の PP 情報が格納されている構造体のﾎﾟｲﾝﾀ
        ULONG upara                     // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S7F5(ByVal eqid As Integer, ByVal end_status As Integer, ByRef info As
dsh_info.TPP_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS7F5(int eqid, int end_status, ref TPP_INFO info, uint upara);
```

end_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。ppinfo の構造体に PP 情報が格納されます。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．11．3．4　GemSendS7F17() - プロセスプログラム削除メッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS7F17(
        int   eqid,                     // 通信対象装置 ID(0,1,2,...)
        char **ppid_list,               // 削除する PPID 格納領域のﾎﾟｲﾝﾀﾘｽﾄ
        int   count,                    // 削除する PPID の数
        int   *ackc7,                   // S7F18 応答 ACKC7 格納ﾎﾟｲﾝﾀ
        int   (WINAPI *callback)(),     // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                     // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS7F17 (
        ByVal eqid As Int32,
        ByRef ppid As IntPtr,
        ByVal count As Int32,
        ByRef ackc7 As Int32,
        ByVal callback As vcallback.callback_S7F17,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS7F17(
        int eqid,
        byte[] ppid,
        int count,
        ref int ackc7,
        CallbackS7F17 callback,
        uint upara );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid_list

削除したいプロセスプログラム ID が格納されているポインタのリストポインタです。
count=0 の場合は NULL でも構いません。

count

削除したいプロセスプログラム(PPID)の数です。count = 0 の場合、全プロセスプログラムの削除要求になります。

ackc7

削除結果 ACKC7 を格納する領域のポインタです。

callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバ

ックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。ackc7 に ACKC7 結果が返却される。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

（4）説明

装置にプロセスプログラム情報の削除を要求します。
ppid_list 内に指定された count 分の PPID を S7F17 メッセージに設定し、装置に送信します。
通信が正常に行われた場合、受信した S7F18 応答メッセージ内の ACKC7 を ackc7 に格納し、返却します。
送信要求から S7F18 応答メッセージ受信までの制御は引数の S7F17 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S7F17 送信後、応答ﾒｯｾｰｼﾞ S7F18 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、ackc7 に S7F18 ﾒｯｾｰｼﾞ内の ACKC7 値が渡されます。
あり	送信要求後、S7F17 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常であり、引数 ackc7 に S7F18 ﾒｯｾｰｼﾞ内の ACKC7 値が渡されます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

（5）コールバック関数

[c,C++]

```
API int  APIX callback(
        int    eqid,                  // 装置 ID
        int    end_status,            // 実行結果
        int    *ackc7,                // ACKC7 値が格納されているﾒﾓﾘのﾎﾟｲﾝﾀ
        ULONG upara                   // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S7F17(ByVal eqid As Integer, ByVal end_status As Integer, ByRef ackc7 As Integer,
ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS7F17(int eqid, int status, int* ack, uint upara)
```

end_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。ackc7 に応答ﾒｯｾｰｼﾞの ACKC7 値が返却される。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

３．11．３．５　GemSendS7F19() - プロセスプログラム一覧要求メッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS7F19(
        int   eqid,                     // 通信対象装置 ID(0,1,2,...)
        TPPID_LIST *list,               // 取得した PPID 一覧格納領域のﾎﾟｲﾝﾀ
        int   (WINAPI *callback)(),     // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                     // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS7F19 (
        ByVal eqid As Int32,
        ByRef list As dsh_info.TPPID_LIST,
        ByVal callback As vcallback.callback_S7F19,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS7F19(
        int eqid,
        ref TPPID_LIST list,
        CallbackS7F19 callback,
        uint upara );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

list

装置から S7F20 応答メッセージで取得した PPID 一覧を格納するための TPP_LIST 構造体のポインタです。

callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。ユーザは任意の関数名を指定できます。

コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。 list 内に取得した PPID ﾘｽﾄが格納され返却される。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

（4）説明

装置にプロセスプログラム情報の一覧要求を要求します。
通信が正常に行われた場合、受信した S7F20 応答メッセージ内の PPID の一覧を list 構造体に格納した上で APP に返却されます。取得した PPID の数は list 内に count メンバーに設定されます。
送信要求から S7F20 応答メッセージ受信までの制御は引数の S7F19 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S7F19 送信後、応答ﾒｯｾｰｼﾞ S7F20 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、list に S7F20 ﾒｯｾｰｼﾞ内の PPID 一覧が渡されます。
あり	送信要求後、S7F19 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常であり、list に S7F20 ﾒｯｾｰｼﾞ内の PPID 一覧が渡されます。ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

（5）コールバック関数

[c,C++]

```
API int  APIX callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TPP_LIST   *list,               // PPID 一覧が格納されている構造体のﾎﾟｲﾝﾀ
        ULONG upara                     // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S7F19(ByVal eqid As Integer, ByVal end_status As Integer, ByRef list As
dsh_info.TPPID_LIST, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS7F19(int eqid, int status, ref TPPID_LIST list, uint upara);
```

end_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。list に S7F20 ﾒｯｾｰｼﾞ内の PPID 一覧が渡されます。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．11．４　PP プロセスプログラム関連ライブラリ関数

3．11．４．１　DshDecodeS7F3() - S7F3 をプロセスプログラム情報にデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS7F3(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TPP_INFO *pinfo                 // ﾃﾞｺｰﾄﾞした情報を格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS7F3 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef info As dsh_info.TPP_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS7F3(
        ref DSHMSG smsg,
        ref TPP_INFO info );
```

（2）引数

smsg

S7F3 の SECS ﾒｯｾｰｼﾞ情報が格納されている構造体のポインタです。

pinfo

デコードしたプロセスプログラム情報を格納する構造体のポインタです。

（3）戻り値

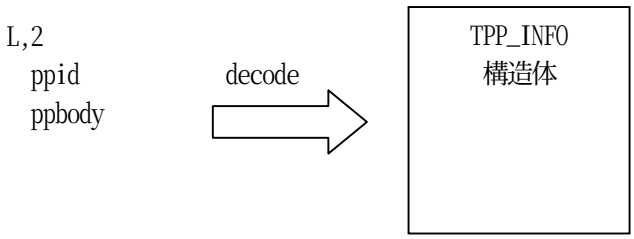
戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

S7F3 メッセージに含まれるプロセスプログラム情報を、ユーザプログラムが処理しやすい TPP_INFO 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTPP_INFO()関数を使って開放してください。(DshResponseS7F4()を使う場合、DshResonseS7F4()が開放してくれます。)

smsg S7F3



3．11．４．２　DshEncodeS7F3() - プロセスプログラム情報を S7F3 へエンコード

（1）呼出書式

[c,C++]

```
API int  APIX DshEncodeS7F3(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        BYTE   *buffer,                 // S7F3 を格納するﾊﾞｯﾌｧﾎﾟｲﾝﾀ
        int    bufflen,                 // buffer のﾊﾞｲﾄｻｲｽﾞ
        TPP_INFO *pinfo                 // ｴﾝｺｰﾄﾞしたい PP 情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshEncodeS7F3 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef buff As Byte,
        ByVal bufflen As Int32,
        ByRef info As dsh_info.TPP_INFO) As Int32
```

[.NET C#]

```
int DshEncodeS7F3(
        ref DSHMSG smsg,
        byte[] buff,
        int bufflen,
        ref TPP_INFO info );
```

（2）引数

smsg

エンコードした S7F3 メッセージを格納するメッセージ情報構造体のポインタです。

buffer

エンコードした S7F3 のテキストを格納するためのバッファポインタです。

bufflen

buffer のバイトサイズです。

pinfo

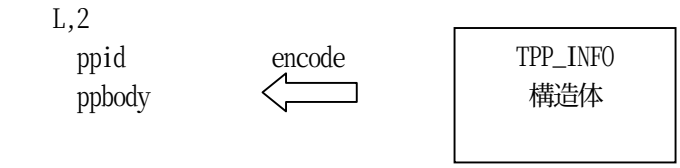
エンコードしたいプロセスプログラム情報が格納されている構造体のポインタです。

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	smsg を正しくｴﾝｺｰﾄﾞできなかった。

（4）説明

TPP_INFO 構造体に格納されているプロセスプログラム情報を S7F3 の SECS メッセージにエンコードします。
buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。



３．11．４．３　DshFreeTPP_INFO() - プロセスプログラム情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPP_INFO(
      TPP_INFO *pinfo                    // ﾒﾓﾘを開放したい PP 情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTPP_INFO (
        ByRef info As dsh_info.TPP_INFO)
```

[.NET C#]

```
void DshFreeTPP_INFO_ALL(
        ref TPP_INFO info );
```

（2）引数

pinfo

メモリを解放したいプロセスプログラム情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPP_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPP_INFO の内容を 全て 0 で初期設定します。
pinfo が NULL ならば、何も処理しません。

3．11．４．４　DshCopyTPP_INFO() - プロセスプログラム情報構造体メモリのコピー

（1）呼出書式

[c,C++]

```
API int APIX DshCopyTPP_INFO(
        TPP_INFO *dinfo,                    // ｺﾋﾟｰ先のﾎﾟｲﾝﾀ
        TPP_INFO *sinfo                     // ｺﾋﾟｰ元のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshCopyTPP_INFO (
          ByRef dinfo As dsh_info.TPP_INFO,
          ByRef sinfo As dsh_info.TPP_INFO) As Int32
```

[.NET C#]

```
int DshCopyTPP_INFO(
          ref TPP_INFO dinfo,
          ref TPP_INFO sinfo );
```

（2）引数

dinfo

コピー先構造体メモリのポインタです。

sinfo

コピー元のプロセスプログラム情報が格納されている構造体メモリのポインタです。

（3）戻り値

戻り値	意味
0	正常にｺﾋﾟｰできた。
(-1)	sinfo またはdinfo の値がNULL だったのでｺﾋﾟｰできなかった。

（4）説明

sinfo が指すTPP_INFO 構造体内に格納されているプロセスプログラム情報をdinfo で指定されたTPP_INFO 構造体メモリにコピーします。

dinfo 内のメンバーで新しいメモリが必要なものは本関数が取得します。

dinfo 内メンバーで確保されたメモリは、使用後、DshFreeTPP_INFO()関数を使って開放してください。

3．11．４．５　DshInitPpInfo – プロセスプログラム TPP_INFO の初期設定

（1）呼出書式

[c,C++]

```
API int  APIX DshInitPpInfo(
        TPP_INFO     *info,              // TPP_INFO 構造体のﾎﾟｲﾝﾀ
        int          ppid_fmt,           // ppid のﾌｫｰﾏｯﾄ
        int          ppid_size,          // ppid の配列ｻｲｽﾞ
        char         *ppid,              // ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID
        int          ppbody_fmt,         // ppbody のﾌｫｰﾏｯﾄ
        int          ppbody_size,        // ppbody の配列ｻｲｽﾞ
        char         *ppbody             // ppbody 値
);
```

[.NET VB]

```
Sub DshInitPpInfo (
        ByRef info As dsh_info.TPP_INFO,
        ByVal ppid_fmt As Int32,
        ByVal ppid_size As Int32,
        ByVal ppid As String,
        ByVal ppbody_fmt As Int32,
        ByVal ppbody_size As Int32,
        ByVal ppbody As String)
```

[.NET C#]

```
void DshInitPpInfo(
        ref TPP_INFO info,
        int ppid_fmt,
        int ppid_size,
        byte[] ppid,
        int ppbody_fmt,
        int ppbody_size,
        byte[] ppbody );
```

（2）引数

info

プロセスプログラム TPP_INFO 構造体のポインタです。このメンバーを初期設定します。

ppid_fmt

設定するプロセスプログラム ID のフォーマットです。(ICODE_A を指定してください。)

ppid_size

設定するプロセスプログラム ID の配列サイズです。

ppid

設定するプロセスプログラム ID です。

ppbody_fmt

設定するプロセスプログラムの PPBODY フォーマットです。(ICODE_A を指定してください。)

ppbody_size

設定するプロセスプログラムの PPBODY 配列サイズです。

ppbody_fmt

設定するプロセスプログラム PPBODY 値です。

（3）戻り値

なし。

（4）説明

本関数は APP が OFFLINE でﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を生成する際に使用することができます。
最初に info 内をクリアします。そして、引数で指定された情報を info 内に設定します。
メモリが必要なメンバーについてはメモリを確保し情報をコピーします。

TPP_INFO 構造体の使用後は DshFreeTPP_INFO()関数を使って構造体内部で使用したメモリを開放してください。

3．11．４．６　DshMakeS7F3Response() - S7F3 の応答メッセージの生成

（1）呼出書式

[c,C++]

```
API int APIX DshMakeS7F3Response(
        BYTE ackc7,                     // S7F4 に設定する ACKC7 です。
        DSHMSG *smsg,                   // S7F4 ﾒｯｾｰｼﾞを格納するﾒｯｾｰｼﾞ構造体のﾎﾟｲﾝﾀ
        BYTE   *buff,                   // S7F4 のﾃｷｽﾄ格納ﾊﾞｯﾌｧﾎﾟｲﾝﾀ
        int    buff_size                // buff のﾊﾞｲﾄｻｲｽﾞ
);
```

[.NET VB]

```
Function DshMakeS7F3Response (
          ByVal ackc7 As Byte,
          ByRef smsg As dshdr2.DSHMSG,
          ByRef buff As Byte,
          ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS7F3Response(
          byte ackc7,
          ref DSHMSG smsg,
          byte[] buff,
          int buff_size );
```

（2）引数

ackc7

S7F4 メッセージの ACKC7 です。

msg

S7F4 応答メッセージ情報を格納するためのメッセージ構造体のポインタです。

buff

S7F4 応答メッセージのテキストを格納するためのバッファポインタです。

buff_size

buff のバイトサイズです。

（3）戻り値

戻り値	意味
0	正常に生成できた。
(-1)	生成できなかった。(buff 領域不足)

（4）説明

S7F4 応答メッセージを msg 内に生成し、ackc7 を S7F4 の ACKC7 として設定します。
ackc7 の設定はユーザが PP 情報を評価した結果です。

3．11．４．７　DshDecodeS7F17() - S7F17 デコード削除プロセスプログラム一覧

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS7F17(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TPPID_LIST *list                // ﾃﾞｺｰﾄﾞした PPID 一覧情報を格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS7F17 ( _
        ByRef smsg As dshdr2.DSHMSG, _
        ByRef list As dsh_info.TPPID_LIST) As Int32
```

[.NET C#]

```
int DshDecodeS7F17(
          ref DSHMSG smsg,
          ref TPPID_LIST list );
```

（2）引数

smsg

S7F17 の SECS ﾒｯｾｰｼﾞ情報が格納されている構造体のポインタです。

list

デコードした PPID 一覧情報を格納する構造体のポインタです。

（3）戻り値

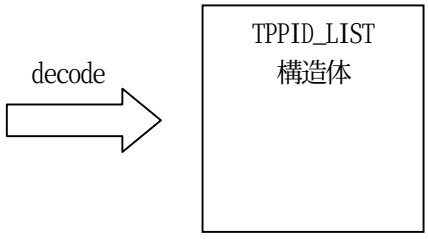
戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

S7F17 メッセージに含まれるプロセスプログラム ID 情報を、ユーザプログラムが処理しやすい TPPID_LIST 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTPPID_LIST()関数を使って開放してください。

smsg S7F17



3．11．４．８　DshDecodeS7F20() - S7F20 デコード - プロセスプログラム一覧情報

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS7F20(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TPPID_LIST *list                // ﾃﾞｺｰﾄﾞした PPID 一覧情報を格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS7F20 (
          ByRef smsg As dshdr2.DSHMSG,
          ByRef list As dsh_info.TPPID_LIST) As Int32
```

[.NET C#]

```
int DshDecodeS7F20(
          ref DSHMSG smsg,
          ref TPPID_LIST list );
```

（2）引数

smsg

S7F20 の SECS ﾒｯｾｰｼﾞ情報が格納されている構造体のポインタです。

list

デコードした PPID 一覧情報を格納する構造体のポインタです。

（3）戻り値

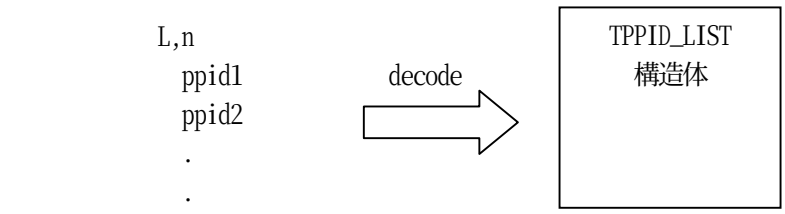
戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

S7F20 メッセージに含まれるプロセスプログラム ID 情報を、ユーザプログラムが処理しやすい TPPID_LIST 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTPPID_LIST()関数を使って開放してください。

smsg S7F20



3．11．４．９　DshFreeTPPID_LIST() - プロセスプログラム ID 一覧情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPPID_LIST(
      TPPID_LIST *list                  // ﾒﾓﾘを開放したい PPID 情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTPPID_LIST (
        ByRef list As dsh_info.TPPID_LIST)
```

[.NET C#]

```
void DshFreeTPPID_LIST(
        ref TPPID_LIST list );
```

（2）引数

list

メモリを解放したいプロセスプログラム ID 一覧情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPPID_LIST 構造体内で PPID 情報格納用に使用されているメモリを全て解放します。
開放した後、TPPID_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

3．11．４．10　DshDecodeS7F27() - S7F27 をプロセスプログラム妥当性情報にデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS7F27(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TPP_PVS_LIST *list              // ﾃﾞｺｰﾄﾞした PP 妥当性ﾁｪｯｸ情報を格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS7F27 (
          ByRef smsg As dshdr2.DSHMSG,
          ByRef list As dsh_info.TPP_PVS_LIST) As Int32
```

[.NET C#]

```
int DshDecodeS7F27(
          ref DSHMSG smsg,
          ref TPP_PVS_LIST list );
```

（2）引数

smsg

S7F27 の SECS ﾒｯｾｰｼﾞ情報が格納されている構造体のポインタです。

list

デコードした PP 妥当性チェック情報を格納する構造体のポインタです。

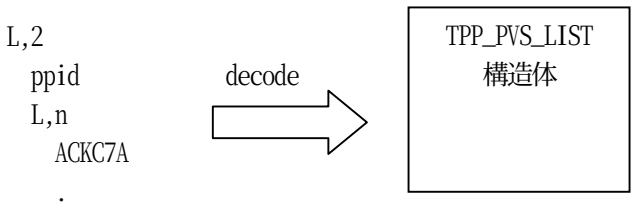
（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

S7F27 メッセージに含まれるプロセスプログラム ID 情報を、ユーザプログラムが処理しやすい TPP_PVS_LIST 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTPP_PVS_LIST()関数を使って開放してください。



3．11．４．11　DshFreeTPP_PVS_LIST() - プロセスプログラム妥当性情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPP_PVS_LIST(
      TPP_PVS_LIST *list              // ﾒﾓﾘを開放したい妥当性情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTPP_PVS_LIST (
        ByRef list As dsh_info.TPP_PVS_LIST)
```

[.NET C#]

```
void DshFreeTPP_PVS_LIST(
        ref TPP_PVS_LIST list );
```

（2）引数

list

メモリを解放したいプロセスプログラム妥当性情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPP_PVS_LIST 構造体内で妥当性情報格納用に使用されているメモリを全て解放します。
開放した後、TPP_PVS_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

3．11．４　ユーザ作成ライブラリ関数

3．11．４．１　DshResponseS7F2() - S7F2 PP ロード問合せ応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS7F2(
        int     eqid,                   // 通信対象装置 ID(0,16,...)
        ID_TR   trid,                   // DSHDR2 のﾄﾗﾝｻﾞｸｼｮﾝ ID
        TPPINQ_INFO *info,              // PP ﾛｰﾄﾞ問合せﾒｯｾｰｼﾞ情報格納領域のﾎﾟｲﾝﾀ
        int     ppgnt                   // S7F2 応答 ACK
);
```

[.NET VB]

```
Function DshResponseS7F2 (
          ByVal eqid As Int32,
          ByVal trid As Int32,
          ByRef info As dsh_info.TPPINQ_INFO,
          ByVal ppgnt As Int32) As Int32
```

[.NET C#]

```
int DshResponseS7F2(
          int eqid,
          uint trid,
          ref TPPINQ_INFO info,
          int ppgnt );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

trid

S7F1 信時に DSHGEMLIB から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

PP ロード問合せ情報が格納されている構造体のポインタです。

ppgnt

送信する応答メッセージ S7F2 の許可情報

（3）戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

（4）説明

PP ロード問合せメッセージ S7F1 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL(dshgemulib.dll)に含まれる関数ですが、ここでは DSHGEMLIB パッケー

ジに標準的な関数として付属されているものです。（ユーザ独自による作成も可能です）

引数に指定されている ppgnt 情報から S7F2 メッセージを組み立て、その後、S7F2 メッセージを送信します。

なお、S7F2 メッセージの組み立てに、DshMakeS7F2Response()関数を使用できます。

３．11．４．２　DshResponseS7F4() – S7F4 PP送信応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS7F4(
        int     eqid,                   // 通信対象装置ID(0,16,...)
        ID_TR   trid,                   // DSHDR2のﾄﾗﾝｻﾞｸｼｮﾝID
        TPP_INFO *info,                 // PP送信ﾒｯｾｰｼﾞ情報格納領域のﾎﾟｲﾝﾀ
        int     ackc7                   // S7F4応答ACK
);
```

[.NET VB]

```
Function DshResponseS7F4 (
        ByVal eqid As Int32,
        ByVal trid As Int32,
        ByRef info As dsh_info.TPP_INFO,
        ByVal ackc7 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS7F4(
        int eqid,
        uint trid,
        ref TPP_INFO info,
        int ackc7 );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

trid

S7F3信時にDSHGEMLIBから渡されるDSHDR2通信ドライバーのトランザクション管理のためのIDです。

info

PP送信情報が格納されている構造体のポインタです。

ackc7

送信する応答メッセージS7F4の許可情報

（3）戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

（4）説明

PP送信メッセージS7F3に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリDLL(dshgemulib.dll)に含まれる関数ですが、ここではDSHGEMLIBパッケージに標準的な関数として付属されているものです。（ユーザ独自による作成も可能です）

引数に指定されているackc7情報からS7F4メッセージを組み立て、その後、S7F4メッセージを送信します。

なお、S7F4メッセージの組み立てに、DshMakeS7F4Response()関数を使用できます。

3．11．４．３　DshResponseS7F6() - S7F6 PP要求応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS7F6(
        int     eqid,                   // 通信対象装置ID(0,16,...)
        ID_TR   trid,                   // DSHDR2のﾄﾗﾝｻﾞｸｼｮﾝID
        TPP_INFO *info,                 // PP要求ﾒｯｾｰｼﾞ情報格納領域のﾎﾟｲﾝﾀ
        int     ackc7                   // S7F6応答ACK
);
```

[.NET VB]

```
Function DshResponseS7F6 (
        ByVal eqid As Int32,
        ByVal trid As Int32,
        ByRef info As dsh_info.TPP_INFO,
        ByVal ackc7 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS7F6(
        int eqid,
        uint trid,
        ref TPP_INFO info,
        int ackc7 );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

trid

S7F5信時にDSHGEMLIBから渡されるDSHDR2通信ドライバーのトランザクション管理のためのIDです。

info

PP要求情報が格納されている構造体のポインタです。この内容が応答されます。

ackc7

送信する応答メッセージS7F6の許可情報

（3）戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

（4）説明

PP要求メッセージS7F5に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリDLL(dshgemulib.dll)に含まれる関数ですが、ここではDSHGEMLIBパッケージに標準的な関数として付属されているものです。（ユーザ独自による作成も可能です）

引数に指定されている info と ackc7 情報から S7F6 メッセージを組み立て、その後、S7F6 メッセージを送信します。

ackc7 が 0 以外の値（エラー）であれば、List=0 のメッセージを応答します。

なお、S7F6 メッセージの組み立てに、DshEncodePp ()関数を使用できます。

送信後は、info の内部で使用されているメモリを DshFreeTPP_INFO()で開放します。

３．11．４．４　DshResponseS7F18() – S7F18 PP 削除指示応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS7F18(
        int     eqid,                   // 通信対象装置 ID(0,16,...)
        ID_TR   trid,                   // DSHDR2 のﾄﾗﾝｻﾞｸｼｮﾝ ID
        int     ackc7                   // S7F18 応答 ACK
);
```

[.NET VB]

```
Function DshResponseS7F18 (
        ByVal eqid As Int32,
        ByVal trid As Int32,
        ByVal ackc7 As Long) As Int32
```

[.NET C#]

```
int DshResponseS7F18(
        int eqid,
        ID_TR trid,
        int ackc7);
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

trid

S7F17 信時に DSHGEMLIB から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

ackc7

送信する応答メッセージ S7F18 の許可情報

（3）戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

（4）説明

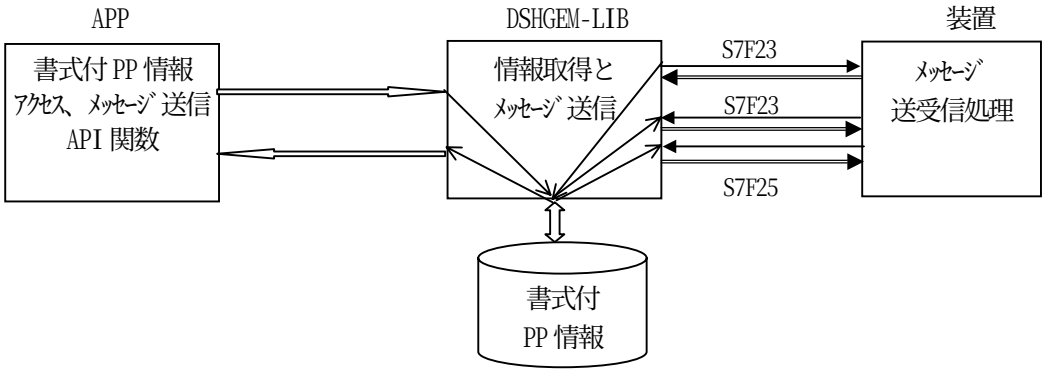
PP 削除指示メッセージ S7F17 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL(dshgemulib.dll)に含まれる関数ですが、ここでは DSHGEMLIB パッケージに標準的な関数として付属されているものです。（ユーザ独自による作成も可能です）

引数に指定されている ackc7 情報から S7F18 メッセージを組み立て、その後、S7F18 メッセージを送信します。

3．12　FPP 書式付プロセスプログラム情報アクセスサービス関数

ここで述べる書式付プロセスプログラム情報は、DSHGEM-LIB が管理します。従って、APP はこれらの情報をアクセスと関連メッセージを送信するために以下の DSHGEM-LIB API 関数を使用します。



（1）情報アクセスと送信 API 関数

プロセスプログラム情報のアクセスと装置へのメッセージ送信に関連するサービスのための API 関数名は一覧表のとおりです。

	API 関数名	機能
1	GemAllocFppInfo()	PPID 情報領域を割当て登録します。
2	GemSetFppInfo()	書式付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を設定・変更します。
3	GemGetFppInfo()	PPID 指定で書式付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を取得します。
4	GemGetFppInfoX()	PPID 情報ｲﾝﾃﾞｸｽ指定で書式付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を取得します。
5	GemDelFppInfo()	PPID 指定で書式付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を削除します。
6	GemDelFppInfoX()	PPID 情報ｲﾝﾃﾞｸｽ指定で書式付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を削除します。
7	GemSetFppState()	PPID 指定で PPID の状態を設定します。
8	GemSetFppStateX()	PPID 情報ｲﾝﾃﾞｸｽ指定で PPID の状態を設定します。
9	GemGetFppState()	PPID 指定で PPID の状態を取得します。
10	GemGetFppStateX()	PPID 情報ｲﾝﾃﾞｸｽ指定で PPID の状態を取得します。
11	GemGetFppId()	指定した PPID 情報ｲﾝﾃﾞｸｽの PPID を取得します。
12	GemGetFppIdIndex()	指定した PPID の情報ｲﾝﾃﾞｸｽを取得します。
13	GemGetFppList()	PPID の一覧ﾘｽﾄを取得します。
14	GemSendS7F23()	S7F23 ﾒｯｾｰｼﾞで PPID 情報を送信します。
15	GemSendS7F25()	S7F25 ﾒｯｾｰｼﾞを使って PPID 情報を取得します。

PPID インデクスは、DSHGEM-LIB が管理する各 PPID 領域の番号です。このインデクスの値は、GemAllocFppInfo()関数実行時に DSHGEM-LIB によって割当てられ、APP に渡されます。また、PPID 情報の取得時に、情報格納構造体メンバーの index に設定されます。

（2）ライブラリ関数

他にAPPが使用できる書式付プロセスプログラム情報処理用API関数として、以下の関数があります。

	API関数名	機能
1	DshDecodeS7F23()	S7F23のﾒｯｾｰｼﾞ内の書式付PP情報をTFPP_INFO構造体にﾃﾞｺｰﾄﾞ収納するための関数です。
2	DshEncodeS7F23()	TFPP_INFO構造体のﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報をS7F23ﾒｯｾｰｼﾞにｴﾝｺｰﾄﾞするための関数です。
3	DshFreeTFPP_INFO()	書式付PP情報が格納されているTFPP_INFO構造体とその内部で使用されているﾒﾓﾘを開放するための関数です。
4	DshCopyTFPP_INFO()	TFPP_INFOの書式付PP情報をｺﾋﾟｰします。
5	DshInitFppInfo()	書式付PP情報、TFPP_INFO構造体の初期設定を行います。
6	DshPutFppCCode()	書式付PP情報にｺﾏﾝﾄﾞｺｰﾄﾞ情報を設定します。
7	DshPutFppPPara()	書式付PP情報のｺﾏﾝﾄﾞｺｰﾄﾞ情報にﾊﾟﾗﾒｰﾀ情報を設定します。
8	DshPutFppPParaInfo()	書式付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑﾊﾟﾗﾒｰﾀを配列位置指定で追加します。
9	DshMakeS7F23Response()	S7F24応答ﾒｯｾｰｼﾞを生成し、送信するための関数。u_s7f23.cで使用します。

（3）ユーザ作成ライブラリ関数

	ﾗｲﾌﾞﾗﾘ関数名	機能
1	DshResponseS7F24()	S7F24 書式付きプロセスプログラム送信応答メッセージ
2	DshResponseS7F26()	S7F26 書式付きプロセスプログラム要求応答メッセージ

３．12．１　使用する情報格納構造体

書式付プロセスプログラム情報を操作する関数は、共通の情報格納のための TFPP_INFO 構造体を使用します。
TFPP_INFO と、その内部で使用する他の構造体は下記のとおりです。

（1）TFPP_INFO - Formatted Process Program Information

```
typedef struct{
        int     index;                  // 情報格納場所ｲﾝﾃﾞｸｽ
        int     state;                  // PP の状態
        char    *name;                  // pp 名(装置管理情報定義ﾌｧｲﾙで与えられた名前)
        int     ppid_fmt;               // ppid のｱｲﾃﾑﾌｫｰﾏｯﾄ( Format 10 ASCII が設定されます。)
        int     ppid_size;              // 同配列ｻｲｽﾞ(実際の ppid のﾊﾞｲﾄ長です。)
        int     max_ppid_size;          // 配列ｻｲｽﾞの最大値
        char    *ppid;                  // ppid 値格納ﾎﾟｲﾝﾀ
        char    *mdln;                  // mdln 格納 ﾎﾟｲﾝﾀ
        char    *softrev;               // softrev 格納ﾎﾟｲﾝﾀ
        int     ccode_count;            // # of process commands
        TFPP_CCODE  **ccode_list;       // ｺﾏﾝﾄﾞｺｰﾄﾞ情報ﾎﾟｲﾝﾀﾘｽﾄ
        }TFPP_INFO;                     // formatted process program
```

(注) ccode_count = 0 の場合、ｺﾏﾝﾄﾞｺｰﾄﾞがないことを意味します。

（2）TFPP_CCODE - Process Commmand Information

```
typedef struct{
        int     ccode_fmt;              // ccode のｱｲﾃﾑﾌｫｰﾏｯﾄ
        int     ccode_size;             // 同配列ｻｲｽﾞ
        void    *ccode;                 // command code 値格納ﾎﾟｲﾝﾀ
        int     ppara_count;            // # of para count
        TFPP_PARA **ppara_list;         // ﾊﾟﾗﾒｰﾀ情報ﾎﾟｲﾝﾀﾘｽﾄ
        } TFPP_CCODE;
```

(注) ppara_count = 0 の場合、ﾊﾟﾗﾒｰﾀがないことを意味します。

（3）Formatted Program Parameter

```
typedef struct{
        int     ppara_fmt;              // ppara のｱｲﾃﾑﾌｫｰﾏｯﾄ
        int     ppara_size;             // 同配列ｻｲｽﾞ
        void    *ppara;                 // ﾊﾟﾗﾒｰﾀ値格納ﾎﾟｲﾝﾀ
        } TFPP_PARA;
```

3．12．２　FPP 書式付プロセスプログラム情報アクセス関数

3．12．２．１　GemAllocFppInfo() - 書式付プロセスプログラムの登録

（1）呼出書式

[c,C++]

```
API int APIX GemAllocFppInfo(
        int   eqid,                 // 通信対象装置 ID(0,1,2,...)
        char *ppid,                 // 書式付 PPID 格納領域のﾎﾟｲﾝﾀ
        int  *index                 // 得られた情報領域ｲﾝﾃﾞｸｽ格納用ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemAllocFppInfo (
          ByVal eqid As Int32,
          ByVal ppid As String,
          ByRef index As Int32) As Int32
```

[.NET C#]

```
int GemAllocFppInfo(
          int eqid,
          byte *ppid,
          ref int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

登録したい書式付プロセスプログラム ID が格納されているポインタです。

index

登録された情報領域のインデクス値を格納する領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に登録できた。
1	指定された PPID は既に登録されていた。
(-1)	登録できなかった。

（4）説明

書式付プロセスプログラムを新規にシステムに登録するための関数です。
引数 ppid で与えられる書式付プロセスプログラム ID をシステムに登録します。

正常に登録できた場合は、index で指定された領域に登録された情報領域のインデクスが設定返却されます。
もし、ppid に指定された書式付プロセスプログラムが既に登録済みであった場合には関数の戻り値 =1 を返却します。index には既に登録されている情報領域のインデクスが設定されます。

得られたインデクスを使って、情報の設定、取得、削除などのアクセスを行うことができます。

３．12．２．２　GemSetFppInfo()　- 書式付プロセスプログラム情報の設定
.　　GemSetFppInfoX() – インデクス指定による書式付プロセスプログラム情報の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetFppInfo(
        int  eqid,                  // 通信対象装置 ID(0,1,2,...)
        TFPP_INFO *pinfo            // 書式付 PP 情報格納構造体のﾎﾟｲﾝﾀ
);

API int APIX GemSetFppInfoX(
        int  eqid,                  // 通信対象装置 ID(0,1,2,...)
        int  index,                 // GemAllocFppInfo()で得られたｲﾝﾃﾞｸｽ値
        TFPP_INFO *pinfo            // 書式付 PP 情報格納構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemSetFppInfo (
        ByVal eqid As Int32,
        ByRef pinfo As dsh_info.TFPP_INFO) As Int32

Function GemSetFppInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef pinfo As dsh_info.TFPP_INFO) As Int32
```

[.NET C#]

```
int GemSetFppInfo(
        int eqid,
        ref TFPP_INFO pinfo );

int GemSetFppInfoX(
        int eqid,
        int index,
        ref TFPP_INFO pinfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

pinfo

設定したい書式付プロセスプログラム情報が格納されている格納構造体領域のポインタです。

index

書式付 PPID 情報のインデクスです。登録時に GemAllocFppInfo()関数によって与えられます。
インデクスは、PPID から GemGetFppIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に設定できた。

(-1)	設定できなかった。

（4）説明

本関数は、pinfo に格納されている書式付プロセスプログラム情報の設定・変更に使用します。

引数 pinfo 内の ppid メンバーに指定される書式付プロセスプログラム ID の情報として情報が設定されます。

pinfo 内には PPID の他、 コマンドコード、パラメータなどの情報が含まれます。
指定した PPID が既に登録済みである場合、pinfo 内の内容にすべて書き換えられます。

pinfo 内の PPID が未登録であった場合は、登録手続きをしてから PP 情報を設定します。
(GemAllocFppInfo()関数で行われる登録と同じ登録が行われます。)

３．12．２．３　GemGetFppInfo() - 書式付プロセスプログラムの取得
.　　GemGetFppInfoX() - インデクス指定での書式付プロセスプログラムの取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetFppInfo(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        char        *ppid;                  // PPID が格納されている領域のﾎﾟｲﾝﾀ
        TFPP_INFO *pinfo                    // 書式付 PP 情報を格納する構造体ﾎﾟｲﾝﾀ
);

API int APIX GemGetFppInfoX(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        int         index,                  // PPID 情報のインデクス
        TFPP_INFO *pinfo                    // 書式付 PP 情報を格納する構造体ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetFppInfo (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByRef pinfo As dsh_info.TFPP_INFO) As Int32

Function GemGetFppInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef pinfo As dsh_info.TFPP_INFO) As Int32
```

[.NET C#]

```
int GemGetFppInfo(
        int eqid,
        byte *ppid,
        ref TFPP_INFO pinfo );

int GemGetFppInfoX(
        int eqid,
        int index,
        ref TFPP_INFO pinfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

書式付 PPID が格納されている領域のポインタです。

pinfo

取得した書式付 PP 情報を格納するための構造体領域のポインタです。

index

書式付 PPID 情報のインデクスです。登録時に GemAllocFppInfo()関数によって与えられます。

インデクスは、PPID から GemGetFppIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。(PPID が未登録であった。)

（4）説明

ppid または index に指定されている PPID のプロセスプログラム情報を pinfo が指す構造体領域に取得します。

TFPP_INFO 構造体の中に情報を格納するために必要なメモリは、DSHGEM-LIB が準備確保します。即ち、構造体のメンバーの中でポインタになっている情報の実体即ち、ppid, mdln などのためのメモリは DSHGEM-LIB が準備します。
これらのメモリは、使用後、ユーザが DSHGEM-LIB の API 関数を使って次のように開放してください。

```
TFPP_INFO   *pinfo;

if ( GemGetFppInfo( ppidinfo ) == 0 ){
    pinfo の処理
    処理終了後
    DshFreeTFPP_INFO( pinfo );          // pinfo 内に使用されているﾒﾓﾘの開放
 }
```

３．12．２．４　GemDelFppInfo()　- 書式付プロセスプログラムの削除
.　　GemDelFppInfoX() - インデクスでの書式付プロセスプログラムの削除

（1）呼出書式

[c,C++]

```
API int APIX GemDelFppInfo(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        char        *ppid;                  // PPID が格納されている領域のポインタ
);

API int APIX GemDelFppInfoX(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        int         index;                  // ｲﾝﾃﾞｸｽ(0,1,2....)
);
```

[.NET VB]

```
Function GemDelFppInfo (
        ByVal eqid As Int32,
        ByVal ppid As String) As Int32

Function GemDelFppInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32) As Int32
```

[.NET C#]

```
int GemDelFppInfo(
        int eqid,
        byte *ppid );

int GemDelFppInfoX(
        int eqid,
        int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

書式付 PPID が格納されている領域のポインタです。

index

書式付 PPID 情報のインデクスです。登録時に GemAllocFppInfo()関数によって与えられます。
インデクスは、PPID から GemGetFppIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に削除できた。
(-1)	PPID が未登録であった。

（4）説明

ppid または index に指定されている書式付 PPID のプロセスプログラム情報をシステムの登録から削除します。

３．12．２．５　GemSetFppState() － 書式付プロセスプログラム状態の設定
.　　GemSetFppStateX() － インデクスでの書式付プロセスプログラム状態の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetFppState(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *ppid,                // PPID が格納されている領域のポインタ
        int       state                 // 設定したい状態値
);

API int APIX GemSetFppStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // PPID 情報のインデクス
        int       state                 // 設定したい状態値
);
```

[.NET VB]

```
Function GemSetFppState (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByVal state As Int32) As Int32

Function GemSetFppStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal state As Int32) As Int32
```

[.NET C#]

```
int GemSetFppState(
        int eqid,
        byte *ppid,
        int state );

int GemSetFppStateX(
        int eqid,
        int index,
        int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

書式付 PPID が格納されている領域のポインタです。

state

設定したい PPID の状態値です。
値=(-1)は使用不可を意味する値になります。

index

書式付 PPID 情報のインデクスです。登録時に GemAllocFppInfo()関数によって与えられます。
インデクスは、PPID から GemGetFppIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に設定できた。
(-1)	PPID が未登録であった。

（4）説明

書式付プロセスプログラム情報の状態値を設定します。
状態値の意味合いは値=(-1)以外についてはユーザが定義して使用します。

３．12．２．６　GemGetFppState() － 書式付プロセスプログラム状態の取得
.　　GemGetFppStateX() － インデクスでの書式付プロセスプログラム状態の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetFppState(
        int       eqid,              // 通信対象装置 ID(0,1,2,...)
        char      *ppid,             // PPID が格納されている領域のポインタ
        int       *state             // 取得した状態値格納用領域ポインタ
);

API int APIX GemGetFppStateX(
        int       eqid,              // 通信対象装置 ID(0,1,2,...)
        int       index,             // PPID 情報のインデクス
        int       *state             // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetFppState (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByRef state As Int32) As Int32

Function GemGetFppStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef state As Int32) As Int32
```

[.NET C#]

```
int GemGetFppState(
        int eqid,
        byte *ppid,
        ref int state );

int GemGetFppStateX(
        int eqid,
        int index,
        ref int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

書式付 PPID が格納されている領域のポインタです。

state

取得した PPID の状態値を格納する領域のポインタです。

index

書式付 PPID 情報のインデクスです。登録時に GemAllocFppInfo()関数によって与えられます。

インデクスは、PPID から GemGetFppIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	PPID が未登録であった。

（4）説明

書式付プロセスプログラム情報の状態値を取得します。
状態値の意味合いは値=(-1)以外についてはユーザが定義して使用します。

3．12．2．7　GemGetFppList() - 全登録 FPPID 取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetFppList(
        int          eqid,                  // 通信対象装置 ID(0,1,2,...)
        TTEXT_DLIST **list                  // 取得ﾘｽﾄ格納ﾎﾟｲﾝﾀの格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetFppList (
        ByVal eqid As Int32,
        ByRef list As IntPtr) As Int32
```

[.NET C#]

```
int GemGetFppList(
        int eqid,
        IntPtr list );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

list

取得できた FPPID が格納されている TTEXT_DLIST 構造体のポインタを格納する領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

（4）説明

システムに登録されている FPPID(書式付 PPID)とその名前を TTEXT_DLIST 構造体に取出すための関数です。
取出す名前は、装置管理情報定義ファイルで FPPID 定義時に与えられた名前です。
取得した情報の処理が終了した後、DshFreeTText_DLIST()関数で list 内部の情報格納用に使用されているメモリを開放してください。

TTEXT_DLIST 構造体は次のとおりです。

```
typedef struct{
        int         count;              // 取得できた ID 数
        char        **id_list;          // 取得できた ID 格納用配列
        char        **name_list;        // 取得できた名前格納ポインタ配列
        }TTEXT_DLIST;
```

3．12．２．８　GemGetFppId() - インデクスから FPPID（書式付プロセスプログラム ID）の取得

（1）呼出書式

[c,C++]

```
API int  APIX GemGetFppId(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        int     index,                  // FPP 情報のインデクス
        char    *ppid                   // 取得した PPID を格納する領域のポインタ
);
```

[.NET VB]

```
Function GemGetFppId (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal ppid As String) As Int32
```

[.NET C#]

```
int GemGetFppId(
        int eqid,
        int index,
        byte *ppid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

index

FPP 情報のインデクスです。登録時に GemAllocFppInfo()関数によって与えられます。
インデクスは、PPID から GemGetFppIdIndex()関数で取得することができます。

ppid

FPPID を格納するための領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。（未登録）

（4）説明

FPP 情報のインデクスから PPID（書式付プロセスプログラム ID）を取得し、ppid に格納します。
正常に取得できた場合は関数戻り値として 0 が返却されます。

3．12．２．９　GemGetFppIdIndex() -FPPID（書式付プロセスプログラム ID）からインデクスの取得

（1）呼出書式

[c,C++]

```
API int  APIX EgnGetFppIdIndex(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        char    *ppid,                  // PPID が格納されている領域のポインタ
        int     *index                  // 取得したインデクスを格納するための領域のポインタ
);
```

[.NET VB]

```
Function GemGetFppIdIndex (
         ByVal eqid As Int32,
         ByVal ppid As String,
         ByRef index As Int32) As Int32
```

[.NET C#]

```
int GemGetFppIdIndex(
         int eqid,
         byte *ppid,
         ref int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

インデクスを取得したい対象の PPID が格納されている領域のポインタです。

index

取得したインデクスの値を格納するための領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。（未登録）

（4）説明

ppid に指定される PPID（書式付プロセスプログラム ID）からインデクスを取得するための関数です。
取得されたインデクスは index で指定された領域に格納されます。
正常に取得できた場合は関数戻り値として 0 が返却されます。

3．12．3　FPP 書式付プロセスプログラム関連通信メッセージ送信要求関数

3．12．3．1　GemSendS7F23() - 書式付プロセスプログラム情報送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS7F23(
        int   eqid,                         // 通信対象装置 ID(0,1,2,...)
        TFPP_INFO *info,                    // FPP 情報が格納されている構造体ﾎﾟｲﾝﾀ
        int   *ackc7,                       // S7F24 応答 ACKC7 格納ﾎﾟｲﾝﾀ
        int (WINAPI *s7f23Callback)(),      // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                         // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS7F23 (
          ByVal eqid As Int32,
          ByVal ppid As String,
          ByRef ackc7 As Int32,
          ByVal callback As vcallback.callback_S7F23,
          ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS7F23(
          int eqid,
          byte[] ppid,
          ref int ackc7,
          CallbackS7F23 callback,
          uint upara );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

info

送信したい書式付プロセスプログラム情報が格納されている構造体のポインタです。

ackc7

受信した応答メッセージの ACKC7 を格納する領域のポインタです。

s7f23Callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。

	ackc7 に S7F24 の ACKC7 が格納されます。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

（4）説明

装置に info 内に格納されている書式付プロセスプログラム情報の送信を行います。S7F23 メッセージで送信します。

本関数は info の内容を S7F23 メッセージにエンコードし、装置に送信します。

送信要求から S7F24 応答メッセージ受信までの制御は引数の S7F23 コールバック関数指定の有無によって次のようになります。

s7f23Callback の指定	制御の流れ
なし (=0)	S7F23 送信後、応答ﾒｯｾｰｼﾞ S7F24 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、ackc7 に S7F24 の ACKC7 が格納されます。
あり	送信要求後、S7F23 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で、ackc7 に S7F24 の ACKC7 が格納されます。ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

正常に応答メッセージを受信した場合、ackc7 に S7F24 の ACKC7 の値が格納されます。

（5）コールバック関数

[c,C++]

```
API int  APIX s7f23Callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        int    *ackc7,                  // S7F24 の ACKC 7 の値が格納されているメモリポインタ
        ULONG upara                     // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S7F23(ByVal eqid As Integer, ByVal end_status As Integer, ByRef ackc7 As Integer,
ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS7F23(int eqid, int end_status, int *ackc7, uint upara);
```

end_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。ackc7 に S7F24 の ACKC7 が格納されます。。
9	ｽﾌﾟｰﾙされた。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

３．12．３．２　GemSendS7F25() - 書式付プロセスプログラム情報問合せ関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS7F25(
        int  eqid,                          // 通信対象装置 ID(0,1,2,...)
        char *ppid,                         // 書式付 PPID 格納領域のﾎﾟｲﾝﾀ
        TFPP_INFO *info,                    // S7F26 で取得した FPP 情報格納構造体のﾎﾟｲﾝﾀ
        int (WINAPI *s7f25Callback)(),      // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                         // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS7F25 (
        ByVal eqid As Int32,
        ByVal ppid As String,
        ByRef ppinfo As dsh_info.TFPP_INFO,
        ByVal callback As vcallback.callback_S7F25,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS7F25(
        int eqid,
        byte[] ppid,
        ref TFPP_INFO ppinfo,
        CallbackS7F25 callback,
        uint upara );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

ppid

装置に問合せたい書式付プロセスプログラム ID が格納されているポインタです。

info

受信した応答メッセージ S7F26 の FPP 情報を格納する構造体のポインタを指定します。

s7f25Callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。 info 内に FPP 情報が格納されます。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

（4）説明

装置に書式付プロセスプログラム情報の問合せ要求を行います。S7F25 メッセージを使います。
問合せ指定は書式付プロセスプログラム ID で行います。
要求を受けた DSHGEM-LIB は、S7F25 メッセージにエンコードし、装置に送信します。
送信要求から S7F26 応答メッセージ受信までの制御は引数の S7F25 コールバック関数指定の有無によって次のようになります。

s7f25Callback の指定	制御の流れ
なし (=0)	S7F25 送信後、応答ﾒｯｾｰｼﾞ S7F26 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、info 内に FPP 情報が格納されます。
あり	送信要求後、S7F25 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で引数 info 内に FPP 情報が格納されます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

DSHGEM-LIB は S7F26 で得られた情報の装置管理情報としての登録は行いません。必要に応じてユーザ側プログラムで登録のための処理をしてください。

正常に応答メッセージを受信した場合、info 内の情報の処理を終えた後、その構造体に使用されているメモリを解放してください。
解放は次のように DshFreeTFPP_INFO()関数を使って行ってください。

```
DshFreeTFPP_INFO(info);
```

（5）コールバック関数

```
API int  APIX s7f25Callback(
        int    eqid,                  // 装置 ID
        int    end_status,            // 実行結果
        TFPP_INFO *info,              // 応答ﾒｯｾｰｼﾞ S7F26 の FPP 情報が格納されている構造体のﾎﾟｲﾝﾀ
        ULONG upara                   // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S7F25(ByVal eqid As Integer, ByVal end_status As Integer, ByRef info As
dsh_info.TFPP_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS7F25(int eqid, int status, ref TFPP_INFO *info, uint upara);
```

end_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。info 内に FPP 情報が格納されます。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．12．４　FPP 書式付プロセスプログラム関連ライブラリ関数

3．12．４．１　DshDecodeS7F23() - S7F23 を書式付プロセスプログラム情報にデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS7F23(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TFPP_INFO *pinfo                // ﾃﾞｺｰﾄﾞした情報を格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS7F23 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef info As dsh_info.TFPP_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS7F23(
        ref DSHMSG smsg,
        ref TFPP_INFO info );
```

（2）引数

smsg

S7F23 の SECS メッセージ情報が格納されている構造体のポインタです。

pinfo

デコードした書式付プロセスプログラム情報を格納する構造体のポインタです。

（3）戻り値

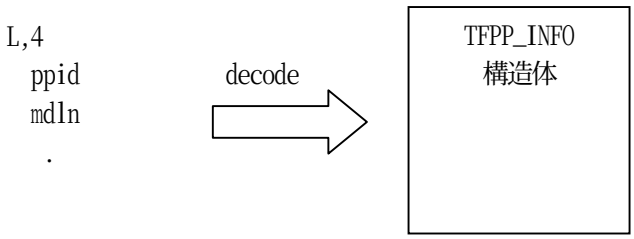
戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

S7F23 メッセージに含まれる書式付プロセスプログラム情報を、ユーザプログラムが処理しやすい TFPP_INFO 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTFPP_INFO()関数を使って開放してください。(DshResponseS7F24()を使う場合、DshResonseS7F24()が開放してくれます。)

smsg S7F23



3．12．４．２　DshEncodeS7F23() - 書式付プロセスプログラム情報を S7F23 へエンコード

（1）呼出書式

[c,C++]

```
API int  APIX DshEncodeS7F23(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        BYTE   *buffer,                 // S7F23 を格納するﾊﾞｯﾌｧﾎﾟｲﾝﾀ
        int    bufflen,                 // buffer のﾊﾞｲﾄｻｲｽﾞ
        TFPP_INFO *pinfo                // ｴﾝｺｰﾄﾞしたい PP 情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshEncodeS7F23 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef buff As Byte,
        ByVal bufflen As Int32,
        ByRef info As dsh_info.TFPP_INFO) As Int32
```

[.NET C#]

```
int DshEncodeS7F23(
        ref DSHMSG smsg,
        byte[] buff,
        int bufflen,
        ref TFPP_INFO info );
```

（2）引数

smsg

エンコードした S7F23 メッセージを格納するメッセージ情報構造体のポインタです。

buffer

エンコードした S7F23 のテキストを格納するためのバッファポインタです。

bufflen

buffer のバイトサイズです。

pinfo

エンコードしたい書式付プロセスプログラム情報が格納されている構造体のポインタです。

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	smsg を正しくｴﾝｺｰﾄﾞできなかった。

（4）説明

TFPP_INFO 構造体に格納されている書式付プロセスプログラム情報を、S7F23 の SECS メッセージにエンコードします。

buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。

３．12．４．３　DshFreeTFPP_INFO() - 書式付プロセスプログラム情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTFPP_INFO(
      TFPP_INFO *pinfo                    // ﾒﾓﾘを開放したいPP情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTFPP_INFO (
        ByRef info As dsh_info.TFPP_INFO)
```

[.NET C#]

```
void DshFreeTFPP_INFO(
        ref TFPP_INFO info );
```

（2）引数

pinfo

メモリを解放したい書式付プロセスプログラム情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TFPP_INFO構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TFPP_INFOの内容を 全て0 で初期設定します。
pinfoがNULLならば、何も処理しません。

3．12．４．４　DshCopyTFPP_INFO() - 書式付プロセスプログラム情報構造体メモリのコピー

（1）呼出書式

[c,C++]

```
API int APIX DshCopyTFPP_INFO(
        TFPP_INFO *dinfo,                   // ｺﾋﾟｰ先のﾎﾟｲﾝﾀ
        TFPP_INFO *sinfo                    // ｺﾋﾟｰ元のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshCopyTFPP_INFO (
          ByRef dinfo As dsh_info.TFPP_INFO,
          ByRef sinfo As dsh_info.TFPP_INFO) As Int32
```

[.NET C#]

```
int DshCopyTFPP_INFO(
          ref TFPP_INFO dinfo,
          ref TFPP_INFO sinfo );
```

（2）引数

dinfo

コピー先の書式付プロセスプログラム情報を格納する構造体メモリのポインタです。

sinfo

コピー元の書式付プロセスプログラム情報が格納されている構造体メモリのポインタです。

（3）戻り値

戻り値	意味
0	正常にｺﾋﾟｰできた。
(-1)	sinfo またはdinfo の値がNULL だったのでｺﾋﾟｰできなかった。

（4）説明

sinfo が指すTFPP_INFO 構造体内に格納されている書式付プロセスプログラム情報をdinfo で指定されたTFPP_INFO 構造体にコピーします。

dinfo 内のメンバーで新しいメモリが必要なものは本関数が取得します。

dinfo 内メンバーで確保されたメモリは,使用後、DshFreeTFPP_INFO()関数を使って開放してください。

3．12．４．５　DshInitFppInfo - 書式付プロセスプログラム TFPP_INFO の初期設定

（1）呼出書式

[c,C++]

```
API int  APIX DshInitFppInfo(
        TFPP_INFO    *info,              // TFPP_INFO 構造体のﾎﾟｲﾝﾀ
        char         *ppid,              // 書式付きﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID
        char         *mdln,              // model 名
        char         *softrev,           // soft revision
        int          ccode_count         // ｺﾏﾝﾄﾞｺｰﾄﾞ情報ﾘｽﾄのｻｲｽﾞ
);
```

[.NET VB]

```
Sub DshInitFppInfo (
        ByRef info As dsh_info.TFPP_INFO,
        ByVal ppid As String,
        ByVal mdln As String,
        ByVal softrev As String,
        ByVal ccode_count As Int32)
```

[.NET C#]

```
void DshInitFppInfo(
        ref TFPP_INFO info,
        byte[] ppid,
        byte[] mdln,
        byte[] softrev,
        int ccode_count );
```

（2）引数

info

書式付プロセスプログラム TFPP_INFO 構造体のポインタです。このメンバーを初期設定します。

ppid

設定する書式付きプロセスプログラム ID（文字列）です。

mdln

書式付きプロセスプログラムのモデル名です。

softrev

書式付きプロセスプログラムの Software Revision です。

ccode_count

TFPP_INFO に含むコマンドコードの最大数です。

（3）戻り値

なし。

（4）説明

本関数は APP が OFFLINE で書式付きﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を生成する際に使用することができます。
最初に info 内をクリアします。そして、引数で指定された情報を info 内に設定します。
メモリが必要なメンバーについてはメモリを確保し情報をコピーします。

また、書式付きプロセスプログラムのコマンドコード情報の設定については次のDshPutFppCCode()関数を使ってください。

TFPP_INFO構造体の使用後はDshFreeTFPP_INFO()関数を使って構造体内部で使用したメモリを開放してください。

3．12．４．６　DshPutFppCCode() -書式付プロセスプログラムコマンドコード情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutFppCCode(
        TFPP_INFO   *info,              // 書式付きﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報構造体のﾎﾟｲﾝﾀ
        int         ccode_fmt,          // ccode 値の format
        int         ccode_size,         // ccode 値の配列ｻｲｽﾞ
        int         max_ccode_size,     // ccode 値の最大配列ｻｲｽﾞ
        void        *ccode,             // ccode 値が格納されている領域ﾎﾟｲﾝﾀ
        int         ppara_count         // 付属ﾊﾟﾗﾒｰﾀの数
);
```

[.NET VB]

```
Function DshPutFppCCode (
        ByRef info As dsh_info.TFPP_INFO,
        ByVal ccode_fmt As Int32,
        ByVal ccode_size As Int32,
        ByVal max_ccode_size As Int32,
        ByVal ccode As String,
        ByVal ppara_count As Int32) As Int32
```

[.NET C#]

```
int DshPutFppCCode(
        ref TFPP_INFO info,
        int ccode_fmt,
        int ccode_size,
        int max_ccode_size,
        byte[] ccode,
        int ppara_count );
```

（2）引数

info

書式付きプロセスプログラム情報構造体のポインタです。

ccode_fmt

コマンドコード値のフォーマットです。

ccode_size

コマンドコード値の配列サイズです。

max_ccode_size

コマンドコード値の最大配列サイズです。

ccode

コマンドコード値が格納されている領域のポインタです。

ppara_count

コマンドコード情報に付属するパラメータの数です。

（3）戻り値

戻り値	意味
0	正常に追加できた。

(-1)	ｺﾏﾝﾄﾞｺｰﾄﾞの数が指定数を超えている。

（4）説明

先にDshInitFppInfo()で初期設定された書式付きプロセスプログラム情報構造体内のccode_list リストにコマンドコード情報 TFPP_CCODE を１個加えます。

ccode_list 内のコマンドコード情報リスト上に、本関数が実行される順に情報を追加していきます。

設定後 0 を返却します。

もし、info 内の ccode_count で指定された分のコマンドコードが既に設定済みであった場合は、(-1)を返却します。

TFPP_CCODE 内へのパラメータ情報の設定は DshPutFppPPara()関数を使って行ってください。

3．12．４．７　DshPutFppPPara() -書式付プロセスプログラムパラメータ情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutFppPPara(
        TFPP_CCODE  *info,              // 書式付きﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑｺﾏﾝﾄﾞｺｰﾄﾞ情報構造体のﾎﾟｲﾝﾀ
        int         ppara_fmt,          // ppara 値の format
        int         ppara_size,         // ppara 値の配列ｻｲｽﾞ
        int         max_ppara_size,     // ppara 値の最大配列ｻｲｽﾞ
        void        *ppara,             // ppara 値が格納されている領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshPutFppPPara (
        ByRef info As dsh_info.TFPP_CCODE,
        ByVal ppara_fmt As Int32,
        ByVal ppara_size As Int32,
        ByVal max_ppara_size As Int32,
        ByVal ppara As IntPtr) As Int32

    Function DshPutFppPParaInfo (
        ByRef info As dsh_info.TFPP_INFO,
        ByVal order As Int32,
        ByVal ppara_fmt As Int32,
        ByVal ppara_size As Int32,
        ByVal max_ppara_size As Int32,
        ByVal ppara As String) As Int32

    Function DshPutFppPParaInfo (
        ByRef info As dsh_info.TFPP_INFO,
        ByVal order As Int32,
        ByVal ppara_fmt As Int32,
        ByVal ppara_size As Int32,
        ByVal max_ppara_size As Int32,
        ByVal ppara As IntPtr) As Int32
```

[.NET C#]

```
int DshPutFppPPara(
        ref TFPP_CCODE info,
        int ppara_fmt,
        int ppara_size,
        int max_ppara_size,
        byte[] ppara );
```

（2）引数

info

書式付きプロセスプログラムコマンドコード情報構造体のポインタです。

ppara_fmt

パラメータ値のフォーマットです。

ppara_size

パラメータ値の配列サイズです。

max_ppara_size

パラメータ値の最大配列サイズです。

ppara

パラメータ値が格納されている領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	ﾊﾟﾗﾒｰﾀの数が指定数を超えている。

（4）説明

先に DshPutFppCCode()で設定された書式付きプロセスプログラムのコマンドコード情報構造体リスト ppara_list にパラメータ情報 TFPP_PPARA を１個加えます。

ppara_list 内のパラメータ情報リスト上に、本関数が実行される順に情報を追加していきます。

設定後 0 を返却します。

もし、info 内の ppara_count で指定された分のパラメータが既に設定済みであった場合は、(-1)を返却します。

3．12．４．８　DshPutFppPParaInfo() -書式付プロセスプログラムパラメータの配列指定追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutFppPParaInfo(
        TFPP_INFO  *info,                  // 書式付きﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報構造体のﾎﾟｲﾝﾀ
        int         index,                 // ccode_list[]配列ｲﾝﾃﾞｸｽ
        int         ppara_fmt,             // ppara 値の format
        int         ppara_size,            // ppara 値の配列ｻｲｽﾞ
        int         max_ppara_size,        // ppara 値の最大配列ｻｲｽﾞ
        void        *ppara,                // ppara 値が格納されている領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshPutFppPParaInfo (
        ByRef info As dsh_info.TFPP_INFO,
        ByVal order As Int32,
        ByVal ppara_fmt As Int32,
        ByVal ppara_size As Int32,
        ByVal max_ppara_size As Int32,
        ByVal ppara As String) As Int32

Function DshPutFppPParaInfo (
        ByRef info As dsh_info.TFPP_INFO,
        ByVal order As Int32,
        ByVal ppara_fmt As Int32,
        ByVal ppara_size As Int32,
        ByVal max_ppara_size As Int32,
        ByVal ppara As IntPtr) As Int32
```

[.NET C#]

```
int DshPutFppPParaInfo(
        ref TFPP_INFO pinfo,
        int order,
        int ppara_fmt,
        int ppara_size,
        int max_ppara_size,
        byte[] ppara );
```

（2）引数

info

書式付きプロセスプログラム情報構造体のポインタです。

index

ccode_list[]配列の位置(0,1,2...)

ppara_fmt

パラメータ値のフォーマットです。

ppara_size

パラメータ値の配列サイズです。

max_ppara_size

パラメータ値の最大配列サイズです。

ppara

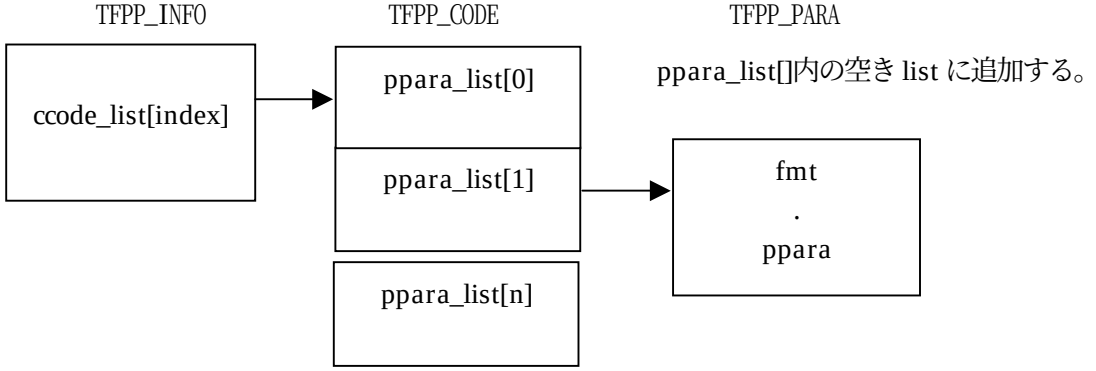
パラメータ値が格納されている領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	ﾊﾟﾗﾒｰﾀの数が指定数を超えている。

（4）説明

先に DshPutFppCCode()で設定された書式付きプロセスプログラムのコマンドコード情報構造体リスト ppara_list の index で指定された配列位置にパラメータ情報 TFPP_PPARA を１個加えます。
ppara_list 内のパラメータ情報リスト上に、本関数が実行される順に情報を追加していきます。
設定後 0 を返却します。
もし、index の値が ppara_list の配列サイズを超えている場合、または、ppara_list の ppara_count で指定された分のパラメータが既に設定済みであった場合は、(-1)を返却します。



３．12．４．９　DshMakeS7F23Response() - S7F23 の応答メッセージの生成

（1）呼出書式

[c,C++]

```
API int APIX DshMakeS7F23Response(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        BYTE    ackc7,                  // S7F24 に設定する ACKC7 です。
        DSHMSG *smsg,                   // S7F24 ﾒｯｾｰｼﾞを格納するﾒｯｾｰｼﾞ構造体のﾎﾟｲﾝﾀ
        BYTE   *buff,                   // S7F24 のﾃｷｽﾄ格納ﾊﾞｯﾌｧﾎﾟｲﾝﾀ
        int     buff_size               // buff のﾊﾞｲﾄｻｲｽﾞ
);
```

[.NET VB]

```
Function DshMakeS7F23Response (
          ByVal ackc7 As Int32,
          ByRef info As dsh_info.TFPP_INFO,
          ByRef smsg As dshdr2.DSHMSG,
          ByRef buff As Byte,
          ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS7F23Response(
          int ackc7,
          ref TFPP_INFO info,
          ref DSHMSG smsg,
          byte[] buff,
          int buff_size );
```

（2）引数

ackc7

S7F24 メッセージの ACKC7 です。

msg

S7F24 応答メッセージ情報を格納するためのメッセージ構造体のポインタです。

buff

S7F24 応答メッセージのテキストを格納するためのバッファポインタです。

buff_size

buff のバイトサイズです。

（3）戻り値

戻り値	意味
0	正常に生成できた。
(-1)	生成できなかった。(buff 領域不足)

（4）説明

S7F23 に対する S7F24 応答メッセージを msg, buff 内に作成します。
応答情報は、ackc7 を S7F24 の ACKC7 として設定します。
ackc7 の設定はユーザが書式付 PP 情報を評価した結果です。

3．12．４　ユーザ作成ライブラリ関数

3．12．４．１　DshResponseS7F24() – S7F24 書式付き PP 送信応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS7F24(
        int     eqid,                   // 通信対象装置 ID(0,16,...)
        ID_TR   trid,                   // DSHDR2 のﾄﾗﾝｻﾞｸｼｮﾝ ID
        TFPP_INFO *info,                //書式付き PP 送信ﾒｯｾｰｼﾞ情報格納領域のﾎﾟｲﾝﾀ
        int     ackc7                   // S7F24 応答 ACK
);
```

[.NET VB]

```
Function DshResponseS7F24 (
        ByVal eqid As Int32,
        ByVal trid As Int32,
        ByRef info As dsh_info.TFPP_INFO,
        ByVal ackc7 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS7F24(
        int eqid,
        uint trid,
        ref TFPP_INFO info,
        int ackc7 );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

trid

S7F23 信時に DSHGEMLIB から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

書式付き PP 送信情報が格納されている構造体のポインタです。

ackc7

送信する応答メッセージ S7F24 の許可情報

（3）戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

（4）説明

書式付き PP 送信メッセージ S7F23 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL(dshgemulib.dll)に含まれる関数ですが、ここでは DSHGEMLIB パッケージに標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている ackc7 情報から S7F24 メッセージを組み立て、その後、S7F24 メッセージを送信します。

なお、S7F24 メッセージの組み立てに、DshMakeS7F24Response()関数を使用できます。

3．12．４．２　DshResponseS726() – S7F26 PP要求応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS726(
        int     eqid,                   // 通信対象装置ID(0,16,...)
        ID_TR   trid,                   // DSHDR2のﾄﾗﾝｻﾞｸｼｮﾝID
        TFPP_INFO *info,                //書式付きPP要求ﾒｯｾｰｼﾞ情報格納領域のﾎﾟｲﾝﾀ
        int     ackc7                   // S726応答ACK
);
```

[.NET VB]

```
Function DshResponseS7F26 (
        ByVal eqid As Int32,
        ByVal trid As Int32,
        ByRef info As dsh_info.TFPP_INFO,
        ByVal ackc7 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS7F26(
        int eqid,
        uint trid,
        ref TFPP_INFO info,
        int ackc7 );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

trid

S7F25信時にDSHGEMLIBから渡されるDSHDR2通信ドライバーのトランザクション管理のためのIDです。

info

書式付きPP要求情報が格納されている構造体のポインタです。この内容が応答されます。

ackc7

送信する応答メッセージS726の許可情報

（3）戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

（4）説明

書式付きPP要求メッセージS7F25に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリDLL(dshgemulib.dll)に含まれる関数ですが、ここではDSHGEMLIBパッケージに標準的な関数として付属されているものです。（ユーザ独自による作成も可能です）

引数に指定されている info と ackc7 情報から S7F26 メッセージを組み立て、その後、S726 メッセージを送信します。

ackc7 が 0 以外の値（エラー）であれば、List=0 のメッセージを応答します。

なお、S7F26 メッセージの組み立てに、DshEncodeFpp ()関数を使用できます。

送信後は、info の内部で使用されているメモリを DshFreeTFPP_INFO()で開放します