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-５／１５

３．９．　Spool スプール関連関数

３．10.　端末表示関連関数

２００９年6月

株式会社データマップ

文書番号　DSHGEM-LIB-09-30325-00

[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】

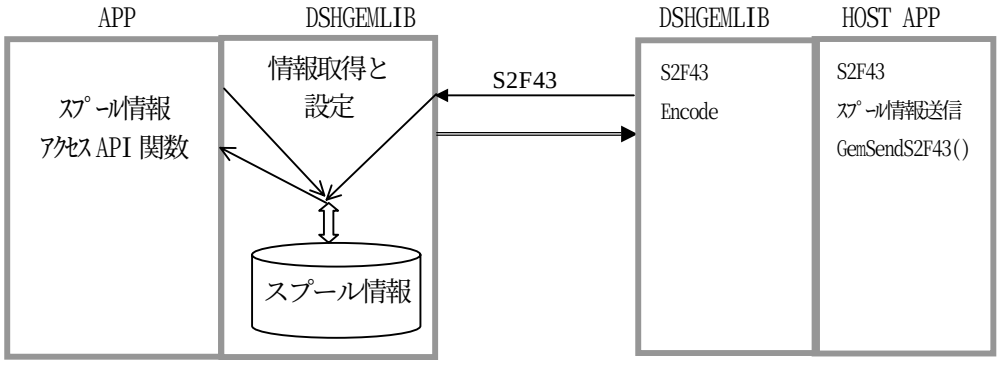
番号	改訂日付	項 目	概 略
1.	2009.6	改訂版	以前の DSHGEM-LIB-07-3032x-00 を全面改訂 .Net VB2008, C#2008 対応関数の説明を追加した。

目　次

(VOL－6 に続く)

３．９　Spool スプール関連関数

ここで述べるスプール関連情報は、DSHGEM-LIB が管理します。従って、APP はこれらの情報をアクセスするために以下の DSHGEM-LIB ライブラリ関数を使用します。



（1）情報アクセスと送信 API 関数

スプール情報アクセス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	GemPutSpoolInfo()	指定した STREAM のスプール情報を登録または変更します。
2	GemGetSpoolInfo()	指定した STREAM のスプール情報を取得します。
3	GemPutAllSpoolInfo()	全 STREAM のスプール情報を登録または変更します。
4	GemGetAllSpoolInfo()	登録済みの全 STREAM のスプール情報を取得します。
5	GemResetSpoolInfo()	指定した STREAM のスプール登録を解除します。(ｽﾌﾟｰﾙ不許可にします。)
6	GemResetAllSpoolInfo()	全 STREAM のスプール登録を解除します。(ｽﾌﾟｰﾙを全て不許可にします。)
7	GemSendS2F43()	S2F43 ｽﾌﾟｰﾙの設定ﾒｯｾｰｼﾞを送信します。
8	GemSendS6F23()	S6F23 ｽﾌﾟｰﾙﾃﾞｰﾀ要求ﾒｯｾｰｼﾞを送信します。

（2）ライブラリ関数

他に APP が使用できるスプール情報処理用 API 関数として、以下の関数があります。

	API 関数名	機能
1	DshEncodeS2F43()	TSPOOL_INFO 構造体のｽﾌﾟｰﾙ情報を S2F43 ﾒｯｾｰｼﾞにｴﾝｺｰﾄﾞするための関数です。
2	DshDecodeS2F43()	S2F43 に含まれるｽﾌﾟｰﾙ情報を TSPOOL_INFO 構造体内にﾃﾞｺｰﾄﾞします。
3	DshFreeTSPOOL_INFO()	スプール情報が格納されている TSPOOL_INFO 構造体と内部で使用されている全ﾒﾓﾘを開放するための関数です。
4	DshCopyTSPOOL_INFO()	TSPOOL_INFO のｽﾌﾟｰﾙ情報をｺﾋﾟｰします。
5	DshDecodeS2F44()	S2F44 ﾒｯｾｰｼﾞの応答情報を TSPOOL_ERR_INFO 構造体に取得するための関数です。
6	DshInitTSPOOL_INFO()	ｽﾌﾟｰﾙ情報構造体 TSPOOL_INFO を初期設定します。
7	DshPutTSPOOL_INFO()	TSPOOL_INFO 構造体内に１個の Stream 情報を設定します。
8	DshInitTSTRE_INFO()	１個のｽﾄﾘｰﾑ情報構造体を初期設定します。
9	DshPutTSTRE_INFO()	ｽﾄﾘｰﾑ情報構造体に１個の Function を設定します。
10	DshInitTSPOOL_ERR_INFO()	S2F44 ﾒｯｾｰｼﾞのｴﾝｺｰﾄﾞに使用される応答情報を TSPOOL_ERR_INFO 構造体の初期設定を行います。
11	DshPutSpool_ERR_INFO()	TSPOOL_ERR_INFO 構造体内にｴﾗｰ情報を設定します。
12	DshFreeTSPOOL_ERR_INFO()	TSPOOL_ERR_INFO 構造体内に使用されたﾒﾓﾘを開放します。
13	DshMakeS2F44Response()	S2F43 に対する応答ﾒｯｾｰｼﾞを TSPOOL_ERR_INFO 構造体内の情報に基づいて生成します。

（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	DshResponseS2F44()	S2F43 スプール（Stream,Function）設定応答

３．９．１　使用する情報格納構造体

スプール情報を操作する関数は、情報格納のための共通の TSPOOL_INFO 構造体を使用します。
TSPOOL_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）**TSPOOL_INFO** - Spool Information

```
typedef struct{
        int         s_count;            // # of streams
        TSTRE_INFO  **stre_list;        // stream info list
        } TSPOOL_INFO;
```

① s_count = 0 の場合、ｽﾌﾟｰﾙ対象 Stream が１つもないことを意味します。
② s_count > 0 の場合、stre_list[]に stream と function 情報が格納されている TSTER_INFO のﾎﾟｲﾝﾀが格納されています。

（2）**TSTRE_INFO** - Spool Stream Information

```
typedef struct{
        int     stream;             // stream
        int     f_count;            // # of functions
        int     *func_list;         // fuction list
        } TSTRE_INFO;
```

① stream はｽﾌﾟｰﾙ指定された stream 値が格納されます。
② f_count には、stream に属するﾒｯｾｰｼﾞで、ｽﾌﾟｰﾙ対象となる function の数になります。
=0 の場合、当該 stream についてｽﾌﾟｰﾙ function がないことを意味します。
③ func_list[]には f_count の数だけの function 値がﾘｽﾄで格納されます。

（3）TSPOOL_ERR_INFO – S2F44 応答情報

```
typedef struct{
        int         rsack;              // ack for s2f43
        int         err_count;          // # of streams
        TSTRE_ERR_INFO  **stre_list;    // err stream info list
        } TSPOOL_ERR_INFO;
```

DSHGEM-LIB はこの情報から S2F44 ﾒｯｾｰｼﾞを組立てﾎｽﾄに送信します。
① rsack は S2F44 の RSACK の値です。
S2F43 の中に指定されたｽﾄﾘｰﾑ、ﾌｧﾝｸｼｮﾝについて全て OK の場合は、=0 が設定されます。
１個以上のｴﾗｰが見つかった場合、rsack には 0 以外の値が設定され、そのｽﾄﾘｰﾑとﾌｧﾝｸｼｮﾝが stre_list に設定されます。
② err_count は rsack != 0 の時のｴﾗｰ補足情報の数です。
この数はｴﾗｰが見つからなくてもそのｽﾄﾘｰﾑとﾌｧﾝｸｼｮﾝを含めることができます。この場合、(4)で述べる func_err[i]の中に=0 を設定してください。func_err[i]=0 の場合、正常扱いになります。
③ stre_list[]は、ｴﾗｰ情報ﾎﾟｲﾝﾀのﾘｽﾄです。err_count 分のﾘｽﾄです。

（4）TSTRE_ERR_INFO – ｴﾗｰ補足情報

```
typedef struct{                         // S2F43 からの取得情報
        int     strack;                 // ack for stream
        int     stream;                 // stream
        int     f_count;                // # of functions
        int     *func_list;             // fuction list
        int     *func_err;              // func err( 0/1 )
        } TSTRE_ERR_INFO;
```

１個のｽﾄﾘｰﾑについて、どのﾌｧﾝｸｼｮﾝがどのｴﾗｰであったかを報せます。

① strack はそのｽﾄﾘｰﾑに対する STRACK です。
② stream はｽﾄﾘｰﾑです。
③ f_count は stream の中のﾌｧﾝｸｼｮﾝの数です。
④ func_list はﾌｧﾝｸｼｮﾝの値のﾘｽﾄです。f_count 分の配列です。
⑤ func_err は func_list の配列に対応していて、ｴﾗｰであったかどうかを示します。
=0 の場合はｴﾗｰがなかったことを、!=0 の場合ｴﾗｰがあったことを意味します。

３．９．２　Spool スプール情報アクセス関数

３．９．２．１　GemSetSpoolInfo() - ストリーム指定スプール情報の登録

（1）呼出書式

[c,C++]

```
API int APIX GemSetSpoolInfo(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        BYTE    stream,                 // ｽﾌﾟｰﾙ登録したい stream
        int     f_count,                // 指定 function の数
        BYTE    *func_list              // f_count 分の function のﾘｽﾄ
);
```

[.NET VB]

```
Function GemSetSpoolInfo (
        ByVal eqid As Int32,
        ByVal stream As Int32,
        ByVal f_count As Int32,
        ByRef func_list As Int32) As Int32
```

[.NET C#]

```
int GemSetSpoolInfo(
        int eqid,
        int stream,
        int f_count,
        int []func_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。

stream

登録したい stream(Sx)の値(2～127)です。

f_count

stream に属する登録したい function(Fy)の数です。

func_list

f_count 分の function の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	正常に登録できた。
(-1)	登録できなかった。

（4）説明

ストリーム単位でそのストリームに属するスプール対象にしたいファンクションを指定して DSHGEM-LIB に登録します。

指定ストリームについて既に登録されている情報は一旦解除され新しく設定しなおします。

（5）コーディング例

S6F11，S6F13 の２個をスプール対象にしたい場合、例えば次のようにコーディングします。

```
BYTE     func_list[2];
int      end_status;
func_list[0]=11;     func_list[1]=13;
end_status = GemPutSpoolInfo( eqid, 6, 2, func_list );
.
```

３．９．２．２　GemSetAllSpoolInfo() - 全スプール情報の登録

（1）呼出書式

[c,C++]

```
API int APIX GemSetAllSpoolInfo(
        int          eqid,                  // 通信対象装置 ID(0,1,2,...)
        TSPOOL_INFO *psinfo                 // ｽﾌﾟｰﾙ情報格納構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemSetAllSpoolInfo (
        ByVal eqid As Int32,
        ByRef ppinfo As dsh_info.TSPOOL_INFO) As Int32
```

[.NET C#]

```
int GemSetAllSpoolInfo(
        int eqid,
        ref TSPOOL_INFO ppinfo );
```

（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。

psinfo

登録したいスプール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	正常に登録できた。
(-1)	登録できなかった。

（4）説明

psinfo に格納されているスプール情報を DSHGEM-LIB に登録します。
既に登録されているスプール情報は全て解除された上で新しく情報を設定します。
psinfo 内には、スプールしたい STREAM とその FUNCTION リスト情報を格納することができます。

（5）コーディング例

S5F1, S6F11, S6F13 を登録する場合、例えば、以下のようにコーディングします。

```
TSPOOL_INFO *sinfo;
sinfo = calloc( sizeof(TSPOOL_INFO), 1 );
sinfo->s_count = 2;                 // S5 と S6
sinfo->stre_list = calloc( sizeof(TSTRE_INFO*) * 2, 1 );  // stre_list ﾎﾟｲﾝﾀﾘｽﾄ用
sinfo->stre_list[0] = calloc( sizeof(TSTRE_INFO), 1 );    // S5 用
sinfo->stre_list[0]->stream = 5;                          // S5
sinfo->stre_list[0]->f_count = 1;                          // 1 個
sinfo->stre_list[0]->func_list = malloc( 1 );
sinfo->stre_list[0]->func_list[0] = 1;                     // F1
```

```
sinfo->stre_list[1] = calloc( sizeof(TSTRE_INFO), 1 );     // S6用
sinfo->stre_list[1]->stream = 6;                           // S6
sinfo->stre_list[1]->f_count = 2;                          // 2個
sinfo->stre_list[1]->func_list = malloc( 2 );
sinfo->stre_list[1]->func_list[0] = 11;                    // F11
sinfo->stre_list[1]->func_list[1] = 13;                    // F13
end_status = GemPutAllSpoolInfo( sinfo );                  // 登録
DshFreeTSPOOL_INFO( sinfo );                               // sinfoの全ﾒﾓﾘを解放
.
.
```

（6）補足事項

受信したS2F43メッセージから、それに含まれているスプール情報をTSPOOL_INFO構造体に取得するための関数　DshDecodeS4F23()が準備されています。

３．９．２．３　GemGetSpoolInfo() - ストリームのスプール登録情報取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetSpoolInfo(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        BYTE        stream;                 // 取得対象 stream
        BYTE        *func_list              // ｽﾌﾟｰﾙ対象 function の格納ﾘｽﾄのﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetSpoolInfo (
        ByVal eqid As Int32,
        ByVal stream As Int32,
        ByRef func_list As Int32) As Int32
```

[.NET C#]

```
int GemGetSpoolInfo(
        int eqid,
        int stream,
        int []func_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。

stream

取得する対象の stream を指定します。

func_list

取得した function を格納するリストのポインタです。登録されている可能性のある最大数の function を格納するための領域を確保しておく必要があります。

（3）戻り値

戻り値	意味
>= 0	取得できた function の数
(-1)	stream が正しくなかった。

（4）説明

stream に指定されたストリームについてスプール登録されている function 情報を取得し、function リストに格納します。

戻り値として 登録されている function 数が返却され、func_list には登録されている function の値を順に格納されます。

３．９．２．４　GemGetAllSpoolInfo() - 全スプール情報の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetAllSpoolInfo(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        TSPOOL_INFO *sinfo                  // ｽﾌﾟｰﾙ情報を格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetAllSpoolInfo (
        ByVal eqid As Int32,
        ByRef ppinfo As dsh_info.TSPOOL_INFO) As Int32
```

[.NET C#]

```
int GemSetAllSpoolInfo(
        int eqid,
        ref TSPOOL_INFO ppinfo );
```

（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。

sinfo

取得したスプール情報を格納するための構造体のポインタです。

（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

DSHGEM-LIB に登録されている全スプール情報を取得し、sinfo 構造体領域に情報を格納します。
構造体メンバー内の情報格納用メモリは DSHGEM-LIB 側で取得します。

スプール情報のために割付けられたメモリと、その構成メンバーデータ用にメモリが獲得されています。
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。

```
        TSPOOL_INFO         sinfo;

            if ( GemGetAllSpoolInfo( eqid, &sinfo ) == 0 ){
                sinfo の処理
                処理終了後
                DshFreeTSPOOL_INFO( sinfo );        // sinfo と内部に使用されているﾒﾓﾘの開放
             }
```

sinfo->s_count に、登録されている stream の数が設定されます。この値が =0 の場合は、スプール対象メッセージがないことを意味します。

３．９．２．５　GemResetSpoolInfo() - ストリーム指定のスプール登録解除

（1）呼出書式

[c,C++]

```
API int APIX GemResetSpoolInfo(
      int       eqid,                  // 通信対象装置 ID(0,1,2,...)
      BYTE      stream;                // stream (  1 < stream < 128 )
);
```

[.NET VB]

```
Function GemResetSpoolInfo (
        ByVal eqid As Int32,
        ByVal stream As Int32) As Int32
```

[.NET C#]

```
int GemResetSpoolInfo(
        int eqid,
        int stream );
```

（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。

stream

スプール解除する対象の stream を指定します。

（3）戻り値

戻り値	意味
0	正常に解除できた。
(-1)	stream の値が間違っている。

（4）説明

DSHGEM-LIB 内部に登録されている stream で指定されたスプール情報を全て解除します。

３．９．２．６　GemResetAllSpoolInfo() - 全スプール登録の削除

（1）呼出書式

[c,C++]

```
API int APIX GemResetAllSpoolInfo(
      int       eqid                  // 通信対象装置 ID(0,1,2,...)
);
```

[.NET VB]

```
Function GemResetSpoolInfo (
       ByVal eqid As Int32, ) As Int32
```

[.NET C#]

```
int GemResetSpoolInfo(
       int eqid  );
```

（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。

（3）戻り値

戻り値	意味
0	正常に削除できた。

（4）説明

DSHGEM-LIB 内に登録されている全てのスプール情報を解除します。

３．９．２．７　GemSendS2F43() - スプール設定情報メッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS2F43(
        int  eqid,                          // 通信対象装置 ID(0,1,2,...)
        TSPOOL_INFO *info,                  // ｽﾌﾟｰﾙ設定情報格納構造体のﾎﾟｲﾝﾀ
        TSPOOL_ERR_INFO *erinfo,            // S2F44 応答情報格納用構造体のﾎﾟｲﾝﾀ
        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 GemSendS2F43 (
        ByVal eqid As Int32,
        ByRef info As dsh_info.TSPOOL_INFO,
        ByRef erinfo As dsh_info.TSPOOL_ERR_INFO,
        ByVal callback As vcallback.callback_S2F43,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS2F43(
        int eqid,
        ref TSPOOL_INFO info,
        ref TSPOOL_ERR_INFO erinfo,
        CallbackS2F43 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

送信したいスプール設定情報が格納されている構造体のポインタです。
スプールの対象したいメッセージの Stream, Function の情報が含まれています。

erinfo

S2F43 に対する応答メッセージ S2F44 の応答情報をデコードし、格納するための構造体ポインタです。

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。 erinfo に応答情報が返却されます。

	(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

スプール設定情報を装置に送信するために S2F43 メッセージの送信要求用関数です。
info にスプール設定情報を設定した上で呼出ます。TSPOOL_INFO 構造体 info へのスプール情報の設定には以下のライブラリ関数を使用することができます。

DshInitTSPOOL_INFO(), DshPutTSPOOL_INFO(), DshInitTSTRE_INFO(), DshPutTSTRE_INFO()

erinfo には S2F43 の応答メッセージ S2F44 の応答情報が格納されます。

送信要求から S2F44 応答メッセージ受信までの制御は引数の S2F43 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)	S2F43 送信後、応答ﾒｯｾｰｼﾞ S2F44 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、erinfo に応答情報が渡されます。
あり	送信要求後、S2F43 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 erinfo に応答情報が渡されます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

正常に応答メッセージを受信した場合、erinfo 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 DshFreeTSPOOL_ERR_INFO()関数を使って行ってください。

DshFreeTSPOOL_ERR_INFO(erinfo);

（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
        TSPOOL_ERINFO   *erinfo,     // S2F44 応答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_S2F43(ByVal eqid As Integer, ByVal end_status As Integer, ByRef erinfo As
dsh_info.TSPOOL_ERR_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS2F43(int eqid, int end_status, ref TSPOOL_ERR_INFO erinfo, 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。erinfo 内に S2F44 の応答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。

３．９．２．８　GemSendS6F23() - スプールデータ要求メッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS6F23(
        int   eqid,                     // 通信対象装置 ID(0,1,2,...)
        int   rsdc,                     // ｽﾌﾟｰﾙ要求ｺｰﾄﾞ ( 0 or 1 )
        int   *rsda,                    // ｽﾌﾟｰﾙ要求応答ﾃﾞｰﾀ RSDA 格納用ﾎﾟｲﾝﾀ
        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 GemSendS6F23 (
        ByVal eqid As Int32,
        ByVal rsdc As Int32,
        ByRef rsda As Int32,
        ByVal callback As vcallback.callback_S6F23,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS6F23(
        int eqid,
        int rsdc,
        ref int rsda,
        CallbackAck 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。

rsdc

スプールデータ要求内容を指定するコードです。
値 = 0 ならばスプールされたデータの転送要求します。
　= 1 ならばスプールされたデータの破棄を要求します。

rsda

S6F24 応答メッセージで返却される RSDA 確認データをするための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。

	rsda に応答 RSDA 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

装置がスプールしているスプールデータに対する要求を行う S6F23 メッセージの送信要求用関数です。
rsdc には要求コードを設定します。

値 = 0 ならばスプールされたデータの転送要求します。
= 1 ならばスプールされたデータの破棄を要求します。

正常に S6F24 メッセージを受信できた場合、rsda に S6F24 の RSDA が格納返却されます。

送信要求から S6F24 応答メッセージ受信までの制御は引数の S6F23 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)	S6F23 送信後、応答ﾒｯｾｰｼﾞ S6F24 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、rsda に応答 RSDA が渡されます。
あり	送信要求後、S6F23 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 rsda に応答 RSDA 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    *rsda,                 // S6F24 の RSDA が格納するための領域のﾎﾟｲﾝﾀ
         ULONG upara                   // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S6F23(ByVal eqid As Integer, ByVal end_status As Integer, ByRef rsda As Integer,
ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackAck(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。rsda に応答 RSDA 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。

３．９．３　Spool スプール関連ライブラリ関数

３．９．３．１　DshEncodeS2F43() - スプール情報を S2F43 へエンコード

（1）呼出書式

[c,C++]

```
API int  APIX DshEncodeS2F43(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        BYTE   *buffer,                 // S2F43 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
        TSPOOL_INFO *sinfo              // ｴﾝｺｰﾄﾞしたいｽﾌﾟｰﾙ情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshEncodeS2F43 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef buff As Byte,
        ByVal bufflen As Int32,
        ByRef info As dsh_info.TSPOOL_INFO) As Int32
```

[.NET C#]

```
int DshEncodeS2F43(
        ref DSHMSG smsg,
        byte[] buff,
        int bufflen,
        ref TSPOOL_INFO info );
```

（2）引数

smsg

エンコードした S2F43 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 S2F43 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。

sinfo

エンコードしたいスプール情報が格納されている構造体のポインタです。

（3）戻り値

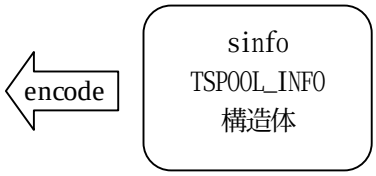
戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	smsg を正しくｴﾝｺｰﾄﾞできなかった。

（4）説明

TSPOOL_INFO 構造体に格納されているスプール情報を、S2F43 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。

```
smsg S2F43
  L,m
    L,2
      stream
        L,n
          func1,
          func2,

          funcn
    L,2
      .
      .
```



３．９．３．２　DshDecodeS2F43( ) - S2F43 をスプール情報にデコード

（1）呼出書式

[c,C++]

```
API int APIX DshDecodeS2F43(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TSPOOL_INFO *sinfo              // 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 DshDecodeS2F43 (
          ByRef smsg As dshdr2.DSHMSG,
          ByRef info As dsh_info.TSPOOL_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS2F43(
          ref DSHMSG smsg,
          ref TSPOOL_INFO info );
```

（2）引数

smsg

S2F43 の SECS メッセージ情報構造体のポインタです。

sinfo

デコードした結果のスプール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

S2F43 メッセージに含まれるスプール情報を、ユーザプログラムが処理しやすい TSPOOL_INFO 構造体の中にデコードします。

スプール構造体のメンバーに使用するメモリは全て DSHGEMLIB 側が準備します。

従って、情報処理終了後は、sinfo のメモリを DshFreeTSPOOL_INFO( )関数を使って解放してください。

（DshResponseS7F4( )を使う場合、DshResonseS7F4( )が開放してくれます。）

```
smsg S2F43
  L,m
    L,2
      stream
        L,n
          func1,
          func2,

          funcn
    L,2
     .
```

.

（5）関連関数

DshEncodeSpoolMSG( ) ： TSPOOL_INFO の内容を S2F43 メッセージにエンコードします。
DshFreeTSPOOL_INFO( ) ： TSPOOL_INFO のメモリを解放します。
DshCopyTSPOOL_INFO( ) ： TSPOOL_INFO の内容を別の構造体にコピーします。
DshMakeS2F43Response( ): TSPOOL_INFO の ACK 情報から S2F44 応答メッセージを作成します。

３．９．３．３　DshFreeTSPOOL_INFO() - スプール情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTSPOOL_INFO(
      TSPOOL_INFO *sinfo                    // ﾒﾓﾘを開放したいｽﾌﾟｰﾙ情報構造体の格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTSPOOL_INFO (
        ByRef info As dsh_info.TSPOOL_INFO)
```

[.NET C#]

```
void DshFreeTSPOOL_INFO(
        ref TSPOOL_INFO info );
```

（2）引数

sinfo

メモリを解放したいスプール情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TSPOOL_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、TSPOOL_INFO の内容を 全て 0 で初期設定します。
sinfo が NULL ならば、何も処理しません。

３．９．３．４　dTSPOOL_INFO() - スプール情報構造体メモリのコピー

（1）呼出書式

[c,C++]

```
int     DshCopyTSPOOL_INFO(
        TSPOOL_INFO *dinfo,                 // ｺﾋﾟｰ先のﾎﾟｲﾝﾀ
        TSPOOL_INFO *sinfo                  // ｺﾋﾟｰ元のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshCopyTSPOOL_INFO (
          ByRef dinfo As dsh_info.TSPOOL_INFO,
          ByRef sinfo As dsh_info.TSPOOL_INFO) As Int32
```

[.NET C#]

```
int DshCopyTSPOOL_INFO(
          ref TSPOOL_INFO dinfo,
          ref TSPOOL_INFO sinfo );
```

（2）引数

dinfo

スプール情報のコピー先構造体メモリのポインタを格納するポインタです。
構造体用メモリは本関数が取得します。

sinfo

コピー元のスプール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

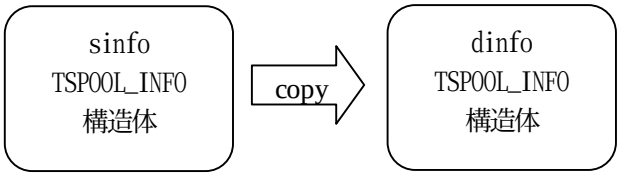
戻り値	意味
0	正常にｺﾋﾟｰできた。
(-1)	sinfo または dinfo の値が NULL だったのでｺﾋﾟｰできなかった。

（4）説明

sinfo が指す TSPOOL_INFO 構造体内に格納されているスプール情報を dinfo で指定された TSPOOL_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 にコピーされた構造体のメモリはユーザが使用後、DSHFreeTSPOOL_INFO()関数で解放してください。



３．９．３．５　DshDecodeS2F44() - S2F44 をスプール設定応答情報にデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS2F44(
        DSHMSG *smsg,                    // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TSPOOL_ERR_INFO *erinfo          // ﾃﾞｺｰﾄﾞしたｽﾌﾟｰﾙ設定応答情報格納構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
int DshDecodeS2F44(
        ref DSHMSG smsg,
        ref TSPOOL_ERR_INFO erinfo );
```

[.NET C#]

```
Function DshDecodeS2F44 (
        ByRef smsg As dshdr2.DSHMSG, _
        ByRef erinfo As dsh_info.TSPOOL_ERR_INFO) As integer
```

（2）引数

smsg

S2F44 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。

erinfo

デコードしたスプール設定応答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。

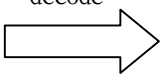
（4）説明

S2F44 メッセージに含まれるスプール設定応答情報を、ユーザプログラムが処理しやすい TSPOOL_ERR_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 DshFreeTSPOOL_ERR_INFO()関数を使って開放してください。

smsg S2F44

```
L,2
  rsack
  L,m
    L,3
    .
```



TSPOOL_ERR_INFO
構造体

３．９．３．６　DshInitTSPOOL_INFO() - スプール S2F43 情報の初期設定

（1）呼出書式

[c,C++]

```
API void APIX DshInitTSPOOL_INFO(
      TSPOOL_INFO *info,                    // ｽﾌﾟｰﾙ情報構造体の格納ﾎﾟｲﾝﾀ
      int     s_count                       // ｽﾌﾟｰﾙ情報内に設定する TSTRE_INFO 構造体の数
);
```

[.NET VB]

```
Sub DshInitTSPOOL_INFO (
        ByRef info As dsh_info.TSPOOL_INFO,
        ByVal s_count As Int32)
```

[.NET C#]

```
void DshInitTSPOOL_INFO(
        ref TSPOOL_INFO info,
        int s_count );
```

（2）引数

info

初期設定する TSPOOL_INFO スプール情報構造体のポインタです。

s_count

スプール情報構造体に後で設定する Stream 情報の数です。

（3）戻り値

なし。

（4）説明

本関数は、TSPOOL_INFO スプール情報構造体の初期設定を行います。
info で指定された構造体のメンバーs_count の値を設定します。

もし、s_count > 0 の場合は、stre_list に s_count だけの TSTRE_INFO 情報構造体のポインタリストを設けます。

実際に stre_list へのスプール情報（TSTRE_INFO 構造体への）設定は DshPutTSTRE_INFO()関数を使って行ってください。s_count 分だけ順に設定できます。

TSOOL_INFO info 使用が済んだら、情報内に確保して使用したメモリは DshFreeTSPOOL_INFO ()関数を使って開放することができます。

３．９．３．７　DshPutTSPOOL_INFO () - スプール情報の設定

（1）呼出書式

[c,C++]

```
API void APIX DshPutTSPOOL_INFO (
        TSPOOL_INFO  *info,             // ｽﾌﾟｰﾙ情報格納構造体のﾎﾟｲﾝﾀ
        TSTRE_INFO   *sinfo             // １個の Stream の情報のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshPutTSPOOL_INFO (
          ByRef info As dsh_info.TSPOOL_INFO,
          ByRef sinfo As dsh_info.TSTRE_INFO) As Int32
```

[.NET C#]

```
int DshPutTSPOOL_INFO(
          ref TSPOOL_INFO info,
          ref TSTRE_INFO sinfo );
```

（2）引数

info

TSPOOL_INFO スプール情報構造体のポインタです。

sinfo

Stream １個分のストリーム情報が格納されている TSTRE_INFO 構造体のポインタです。

（3）戻り値

戻り値	意味
0	正常に設定できた。
(-1)	ﾘｽﾄが満杯で設定できなかった。

（4）説明

本関数は、info 内の stre_list リストの先頭から空きリストを探します。
もし、空きリストがなければ、(-1)を返却します。
もし、空きリストがあれば、その空きリストに sinfo を設定し 0 を返却します。
STRE_INFO 構造体への１個のストリーム情報の設定には DshInitTSTRE_INFO()、DshPutTSTRE_INFO()関数を使ってください。

３．９．３．８　DshInitTSTRE_INFO() - スプール・ストリーム情報の初期設定

（1）呼出書式

[c,C++]

```
API void APIX DshInitTSTRE_INFO(
      TSTRE_INFO **info,                  // ｽﾄﾘｰﾑ情報構造体ﾎﾟｲﾝﾀの格納ﾎﾟｲﾝﾀ
      int     stream,                     // Stream ｺｰﾄﾞ (Sx)
      int     f_count                     // 設定する Function(Fy)の数
);
```

[.NET VB]

```
Sub DshInitTSTRE_INFO (
        ByRef pinfo As IntPtr,
        ByVal stream As Int32,
        ByVal f_count As Int32)
```

[.NET C#]

```
void DshInitTSTRE_INFO(
        ref IntPtr pinfo,
        int stream,
        int f_count );
```

（2）引数

info

初期設定するスプールストリーム情報構造体のポインタです。

stream

TSTRE_INFO に設定する Steam の値です。

f_count

stream に含まれる function の数です。

（3）戻り値

なし。

（4）説明

本関数は、TSPOOL_INFO 内に設定するストリーム１個分の情報を格納するための TSTRE_INFO 構造体の初期設定を行います。

TSTRE_INFO 構造体のメモリは本関数が確保し、そのポインタを info に設定し返します。
そして、構造体のメンバーstream，f_count の値を設定します。

もし、f_count > 0 の場合は、func_list に f_count だけの Function コードを格納するための領域を設けます。

f_list への Function の設定は DshPutTSTRE_INFO()関数を使って行ってください。f_count 分だけ順に設定できます。

ここで確保されたメモリは DshFreeTSPOOL_INFO()関数実行時に開放されます。

３．９．３．９　DshPutTSTRE_INFO () - ストリーム情報の設定

（1）呼出書式

[c,C++]

```
API void APIX DshPutTSTRE_INFO (
        TSTRE_INFO  *info,              // ｽﾄﾘｰﾑ情報格納構造体のﾎﾟｲﾝﾀ
        int         f_code              // Function ｺｰﾄﾞ
);
```

[.NET VB]

```
Function DshPutTSTRE_INFO (
          ByRef info As dsh_info.TSTRE_INFO,
          ByVal f_code As Int32) As Int32
```

[.NET C#]

```
int DshPutTSTRE_INFO(
          ref TSTRE_INFO info,
          int f_code );
```

（2）引数

info

Stream と Function 情報を格納するための TSTRE_INFO 情報構造体のポインタです。

f_code

info 内の func_list に追加する Function コードです。

（3）戻り値

戻り値	意味
0	正常に設定できた。
(-1)	ﾘｽﾄが満杯で設定できなかった。

（4）説明

本関数は、info 内の func_list リストの先頭から空きリストを探します。
もし、空きリストがなければ、(-1)を返却します。
もし、空きリストがあれば、その空きリストに f_code を設定し 0 を返却します。

３．９．３．10　DshInitTSPOOL_ERR_INFO( ) - スプール S2F43 応答情報の初期設定

（1）呼出書式

[c,C++]

```
API void APIX DshInitTSPOOL_ERR_INFO(
      TSPOOL_ERR_INFO *erinfo,                  // 応答情報構造体の格納ﾎﾟｲﾝﾀ
      int     rsack,                            // S2F44 に設定する rsack
      int     err_count                         // ｴﾗｰ情報 TSTRE_ERR_INFO 構造体の数
);
```

[.NET VB]

```
Sub DshInitTSPOOL_ERR_INFO (
        ByRef info As dsh_info.TSPOOL_ERR_INFO,
        ByVal rsack As Int32,
        ByVal err_count As Int32)
```

[.NET C#]

```
void DshInitTSPOOL_ERR_INFO(
        ref TSPOOL_ERR_INFO info,
        int rsack,
        int err_count );
```

（2）引数

erinfo

初期設定する S2F43 スプール応答情報構造体のポインタです。

rsack

S2F44 の rsack に設定する値です。

err_count

スプール応答情報構造体に後で設定するエラー情報の数です。

（3）戻り値

なし。

（4）説明

本関数は、S2F43 スプールリセットの応答メッセージのための TSPOOL_ERR_INFO 構造体の初期設定を行います。

erinfo で指定された構造体のメンバー rsack, err_count の値を設定します。

もし、err_count > 0 の場合は、err_list に err_count だけの TSTRE_ERR_INFO エラー情報構造体のポインタを設けます。

実際に err_list へのエラー情報（TSTRE_ERR_INFO 構造体）への設定は DshPutTSTRE_ERR_INFO( )関数を使って行ってください。err_count 分だけ順に設定できます。

処理が終了したあと、応答情報内に確保して使用したメモリは DshFreeTSPOOL_ERR_INFO ( )関数を使って開放することができます。

３．９．３．11　DshPutTSPOOL_ERR_INFO ( ) - スプールリセットエラー情報の設定

（1）呼出書式

[c,C++]

```
API void APIX DshPutTSPOOL_ERR_INFO (
        TSPOOL_ERR_INFO   *errinfo,          // ｴﾗｰ情報格納構造体のﾎﾟｲﾝﾀ
        TSTRE_ERR_INFO    *serrinfo          // １個の Stream のｴﾗｰ情報のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshPutTSPOOL_ERR_INFO (
          ByRef info As dsh_info.TSPOOL_ERR_INFO,
          ByRef errinfo As dsh_info.TSTRE_ERR_INFO) As Int3
```

[.NET C#]

```
int DshPutTSPOOL_ERR_INFO(
          ref TSPOOL_ERR_INFO info,
          ref TSTRE_ERR_INFO errinfo );
```

（2）引数

errinfo

S2F43 スプールリセット応答メッセージのための TSPOOL_ERR_INFO 情報構造体のポインタです。

serrinfo

Stream １個分のストリームエラー情報が格納されている TSTRE_ERR_INFO 構造体のポインタです。

（3）戻り値

戻り値	意味
0	正常に設定できた。
(-1)	ﾘｽﾄが満杯で設定できなかった。

（4）説明

本関数は、errinfo 内の stre_list リストの先頭から空きリストを探します。
もし、空きリストがなければ、(-1)を返却します。
もし、空きリストがあれば、その空きリストに serr_info を設定し 0 を返却します。
STRE_ERR_INFO 構造体への１個分のストリームエラー情報の設定には DshInitTSTRE_ERR_INFO( )、
DshPutTSTRE_ERR_INFO( )関数を使ってください。

３．９．３．12　DshFreeTSPOOL_ERR_INFO() - スプール S2F43 応答情報メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTSPOOL_ERR_INFO(
      TSPOOL_ERR_INFO *info                    // ﾒﾓﾘを開放したい応答情報構造体の格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTSPOOL_ERR_INFO (
        ByRef info As dsh_info.TSPOOL_ERR_INFO)
```

[.NET C#]

```
void DshFreeTSPOOL_ERR_INFO(
        ref TSPOOL_ERR_INFO info );
```

（2）引数

info

メモリを解放したい S2F43 スプール応答情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TSPOOL_ERR_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。

３．９．３．13　DshMakeS2F43Response( ) - S2F43 の応答メッセージの生成

（1）呼出書式

[c,C++]

```
API int APIX DshMakeS2F43Response(
        TSPOOL_INFO *info,              // ｽﾌﾟｰﾙ情報が格納されている構造体のﾎﾟｲﾝﾀ
        TSPOOL_ERR_INFO *erinfo,        // S2F44 応答情報が格納されている構造体ﾎﾟｲﾝﾀ
        DSHMSG *smsg,                   // S2F44 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,                  // S2F44 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 DshMakeS2F43Response (
          ByRef info As dsh_info.TSPOOL_INFO,
          ByRef erinfo As dsh_info.TSPOOL_ERR_INFO,
          ByRef smsg As dshdr2.DSHMSG,
          ByRef buff As Byte,
          ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS2F43Response(
          ref TSPOOL_INFO info,
          ref TSPOOL_ERR_INFO erinfo,
          ref DSHMSG smsg,
          byte[] buff,
          int buff_size );
```

（2）引数

info

スプール情報が格納されている構造体のポインタです。

erinfo

S2F44 応答情報が格納されている構造体のポインタです。

msg

S2F44 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

S2F44 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

S2F43 に対する S2F44 応答メッセージを erinfo に含まれるスプール応答情報に従って作成します。

① erinfo 内の strack - S2F44 の STRACK として設定します。

② 以下、erinfo 内の err_count 分のエラー情報をメッセージ内にエンコードします。
エラー情報は、stream，strack と function になります。

３．９．４　ユーザ作成ライブラリ関数

３．９．４．１　DshResponseS2F44() – S2F44 スプール設定応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS2F44(
        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
        TSPOOL_INFO *info,              // ｽﾌﾟｰﾙ設定ﾒｯｾｰｼﾞ情報格納領域のﾎﾟｲﾝﾀ
        TSPOOL_ERR_INFO *erinfo         // S2F44 応答情報格納用構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshResponseS2F44 (
          ByVal eqid As Int32,
          ByVal trid As Int32,
          ByRef info As dsh_info.TSPOOL_INFO,
          ByRef erinfo As dsh_info.TSPOOL_ERR_INFO) As Int32
```

[.NET C#]

```
int DshResponseS2F44(
          int eqid,
          uint trid,
          ref TSPOOL_INFO info,
          ref TSPOOL_ERR_INFO erinfo );
```

（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

S2F43 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

スプール設定モード情報が格納されている構造体のポインタです。

erinfo

送信する応答メッセージ S2F44 に含まれる情報を格納するための構造体領域のポインタを指定します。

（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

スプール設定メッセージ S2F43 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(dsh_ulib.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)

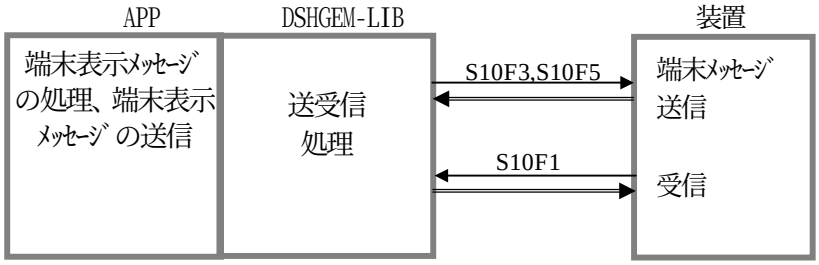
引数に指定されている TSPOOL_ERR_INFO 構造体に含まれている情報から S2F44 メッセージを組み立て、その後、S2F44 メッセージを送信します。

送信が終ったら、TSPOOL_ERR_INFO の構造体に使用されたメモリを DshFreeTASCCESS_ERR_INFO ()関数を使って開放します。

なお、S2F44 メッセージの組み立てに、DshMakeS2F44Response()関数を使用できます。

３．10　端末サービス情報関連関数

端末サービス関連メッセージとして、S10F1, S10F3, S10F5 があります。APP プログラムが関連する処理は、装置から送信されてくる端末表示メッセージの処理ならびに装置への端末表示要求になります。



（1）端末メッセージ送信 API 関数

端末サービスのための API 関数名は次の一覧表のとおりです。

	API 関数名	機能
1	GemSendS10F1()	指定された端末要求を S10F1 で送信する。　HOST<-EQ
2	GemSendS10F3()	指定された端末ﾒｯｾｰｼﾞ(SingleBlock)を S10F3 で送信する。HOST->EQ
3	GemSendS10F5()	指定された端末ﾒｯｾｰｼﾞ(MultiBlock)を S10F5 で送信する。 HOST->EQ

（2）ライブラリ関数

S1F10，S10F3,S10F5 受信時、S10F5 送信時に使用するための関数です。

	API 関数名	機能
1	DshDecodeS10F1()	S10F1 ﾒｯｾｰｼﾞ内の端末ﾃｷｽﾄ情報をﾃﾞｺｰﾄﾞするための関数です。
2	DshDecodeS10F3()	S10F3 ﾒｯｾｰｼﾞ内の端末ﾃｷｽﾄ情報をﾃﾞｺｰﾄﾞするための関数です。
3	DshEncodeS10F5()	TTERMTEXT_INFO の端末情報を S10F5 ﾒｯｾｰｼﾞにｴﾝｺｰﾄﾞします。
4	DshDecodeS10F5()	S10F5 ﾒｯｾｰｼﾞ内の端末情報を TTERMTEXT_INFO 構造体にﾃﾞｺｰﾄﾞします。
5	DshFreeTTERMTEXT_INFO()	S10F5 でﾃﾞｺｰﾄﾞした情報格納に使用されたﾒﾓﾘを解放します。
6	DshCopyTTERMTEXT_INFO()	TTERMTEXT_INFO 構造体を別の構造体にｺﾋﾟｰします。
7	DshInitTTERMTEXT_INFO()	S10F5 用 TTERMTEXT_INO 構造体の初期設定を行います。
8	DshAddTTERMTEXT_INFO()	TTERMTEXT_INFO にﾃｷｽﾄを１行追加します。

（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	DshResponseS10F2()	S10F1 端末要求応答
2	DshResponseS10F4()	S10F3 端末表示応答
3	DshResponseS10F6()	S10F5 端末表示(マルチ)応答

3．10．1　使用する情報格納構造体

S10F5 に含まれる複数ブロック端末表示情報をユーザが処理しやすい以下の TTERMTEXT_INFO 構造体の中にデコードします。

TTERMTEXT_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。

```
typedef struct{
        int     tid;                    // Terminal ID
        int     text_count;             // # of text line
        char    **text_list;            // text line list
        int     ackc10;                 // ackc10 用(Work 用)
}TTERMTEXT_INFO;                        // terminal text
```

3．10．２　端末メッセージ送信関数

3．10．２．１　GemSendS10F1() – 端末要求メッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS10F1(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        int     tid,                    // 端末 ID
        char    *text,                  // 表示ﾃｷｽﾄが格納されているﾎﾟｲﾝﾀ
        int     *ackc10                 // S10F4 内の ACKC10 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 GemSendS10F1 (
        ByVal eqid As Int32,
        ByVal tid As Int32,
        ByVal text As String,
        ByRef ackc10 As Int32,
        ByVal callback As vcallback.Callback_S10F1,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS10F1(
        int eqid,
        int tid,
        byte[] text,
        ref int ackc10,
        CallbackS10F1 TerminalCallback,
        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。

tid

表示する装置の端末 ID です。

text

表示テキストが格納されている領域のポインタです。

ackc10

応答メッセージ S10F4 で得られた ACKC10 を格納するための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。 ackc10 に応答結果が返却される。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
9	ｽﾌﾟｰﾙされた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。
その他 >0	ﾌﾞﾛｯｸﾓｰﾄﾞの場合、ackc10 の値が返却されます。

（4）説明

装置側からホストに対し、端末からのテキスト情報を装置へ送信要求します。S10F1 メッセージで送信します。

端末 ID とテキストメッセージをそれぞれ引数 tid, text で指定します。

要求を受けた DSHGEM-LIB は、端末 ID と表示テキストを S10F1 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 S10F1 応答メッセージ受信までの制御は引数の S10F1 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	S10F1 送信後、応答ﾒｯｾｰｼﾞ S10F2 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、ackc10 に応答結果が返却される。
あり	送信要求後、S10F1 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 ackc10 に応答結果が返却される。 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    *ackc10,                 // ACKC10 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_S10F1(ByVal eqid As Integer, ByVal end_status As Integer, ByRef ack10 As Integer,
ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS10F1(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。ackc10 に S10F4 の応答結果が返却される。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

end_status = 0 のときのみ ackc10 の内容が有効です。

3．10．２．２　GemSendS10F3() - 端末テキスト送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS10F3(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        int     tid,                    // 端末 ID
        char    *text,                  // 表示ﾃｷｽﾄが格納されているﾎﾟｲﾝﾀ
        int     *ackc10                 // S10F4 内の ACKC10 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 GemSendS10F3 (
          ByVal eqid As Int32,
          ByVal tid As Int32,
          ByVal text As String,
          ByRef ackc10 As Int32,
          ByVal callback As vcallback.callback_S10F3,
          ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS10F3(
          int eqid,
          int tid,
          byte[] text,
          ref int ackc10,
          CallbackS10F3 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。

tid

表示する装置の端末 ID です。

text

表示テキストが格納されている領域のポインタです。

ackc10

応答メッセージ S10F4 で得られた ACKC10 を格納するための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。 ackc10 に応答結果が返却される。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
9	ｽﾌﾟｰﾙされた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。
その他 >0	ﾌﾞﾛｯｸﾓｰﾄﾞの場合、ackc10 の値が返却されます。

（4）説明

ホストから装置へ端末からのテキスト情報を装置へ送信要求します。S10F3 メッセージで送信します。
端末 ID とテキストメッセージをそれぞれ引数 tid, text で指定します。
要求を受けた DSHGEM-LIB は、端末 ID と表示テキストを S10F3 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 S10F3 応答メッセージ受信までの制御は引数の S10F3 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	S10F1 送信後、応答ﾒｯｾｰｼﾞ S10F2 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、ackc10 に応答結果が返却される。
あり	送信要求後、S10F1 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 ackc10 に応答結果が返却される。 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    *ackc10,               // ACKC10 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_S10F3(ByVal eqid As Integer, ByVal end_status As Integer, ByRef ack10 As Integer,
ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS10F3(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。ackc10 に S10F4 の応答結果が返却される。

(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

end_status = 0 のときのみ ackc10 の内容が有効です。

3．10．２．３　GemSendS10F5() – 端末表示マルチブロックメッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS10F5(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        TTERMTEXT_INFO   *info,         // 表示ﾃｷｽﾄが格納されている構造体ﾎﾟｲﾝﾀ
        int     *ackc10,                // S10F6 の ACKC10 の格納ﾎﾟｲﾝﾀ
        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 GemSendS10F5 (
        ByVal eqid As Int32,
        ByRef info As dsh_info.TTERMTEXT_INFO,
        ByRef ackc10 As Int32,
        ByVal callback As vcallback.callback_S10F5,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS10F5(
        int eqid,
        ref TTERMTEXT_INFO info,
        ref int ackc10,
        CallbackS10F5 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

マルチブロック端末表示情報が格納されている領域のポインタです。

ackc10

応答メッセージ S10F6 で得られた ACKC10 を格納するための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。ackc10 に応答結果が返却される。 (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

端末からのマルチブロックの端末表示情報を装置へ送信要求します。S10F5 メッセージで送信します。info が示す構造体に設定された端末 ID tid と複数行の表示テキストが格納されています。

要求を受けた DSHGEM-LIB は、info 内の端末 ID と表示テキスト行を S10F5 メッセージにエンコードし、装置に送信します。

TTERMTEXT_INFO 内への表示テキストの設定は、DshInitTTERMTEXT_INFO(),DshAddTTERMTEXT_INFO()関数を使って行うことができます。

送信要求から S10F5 応答メッセージ受信までの制御は引数の S10F5 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	S10F5 送信後、応答ﾒｯｾｰｼﾞ S10F6 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、ackc10 に S10F6 の応答結果が返却される。
あり	送信要求後、S10F5 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 ackc10 に S10F6 の応答結果が返却される。 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    *ackc10,               // ACKC10 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_S10F5(ByVal eqid As Integer, ByVal end_status As Integer, ByRef ack10 As Integer,
ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS10F5(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。ackc10 に S10F6 の応答結果が返却される。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

end_status = 0 のときのみ ackc10 の内容が有効です。

3．10．3　端末表示関連ライブラリ関数

3．10．3．1　DshDecodeS10F1() - S10F1 端末メッセージのデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS10F1(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        int    *tid,                    // 端末 ID を格納する領域のﾎﾟｲﾝﾀ
        char   *text                    // 端末ﾃｷｽﾄ格納用ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS10F1 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef tid As Int32,
        ByVal text As String) As Int32
```

[.NET C#]

```
int DshDecodeS10F1(
        ref DSHMSG smsg,
        ref int tid,
        byte[] text );
```

（2）引数

smsg

S10F1 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。

tid

S10F1 端末メッセージ情報に含まれる端末 ID を格納するための領域ポインタです。

text

S10F1 端末メッセージに含まれる 1 行分のテキストを格納するための領域ポインタです。
81 バイト以上の領域を指定してください。

（3）戻り値

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

（4）説明

S10F1 メッセージに含まれる端末メッセージ情報である端末 ID を tid に、そしてテキストを text で指定された領域にデコードして取り込みます。

３．10．３．２　DshDecodeS10F3() - S10F3 端末メッセージのデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS10F3(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        int    *tid,                    // 端末 ID を格納する領域のﾎﾟｲﾝﾀ
        char   *text                    // 端末ﾃｷｽﾄ格納用ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS10F3 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef tid As Int32,
        ByVal text As String) As Int32
```

[.NET C#]

```
int DshDecodeS10F3(
        ref DSHMSG smsg,
        ref int tid,
        byte[] text );
```

（2）引数

smsg

S10F3 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。

tid

S10F3 端末メッセージ情報に含まれる端末 ID を格納するための領域ポインタです。

text

S10F3 端末メッセージに含まれる 1 行分のテキストを格納するための領域ポインタです。
81 バイト以上の領域を指定してください。

（3）戻り値

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

（4）説明

S10F3 メッセージに含まれる端末メッセージ情報である端末 ID を tid に、そしてテキストを text で指定された領域にデコードして取り込みます。

3．10．3．3　DshEncodeS10F5() - 端末メッセージ情報を S10F5 へエンコード

（1）呼出書式

[c,C++]

```
API int  APIX DshEncodeS10F5(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        BYTE   *buffer,                 // S10F5 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
        TTERMTEXT_INFO *info            // ｴﾝｺｰﾄﾞしたい端末表示情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshEncodeS10F5(
        ByRef smsg As dshdr2.DSHMSG,
        ByRef buff As Byte,
        ByVal bufflen As Integer,
        ByRef info As dsh_info.TTERMTEXT_INFO) As Int32
```

[.NET C#]

```
int DshEncodeS10F5( ref DSHMSG smsg,
        byte[] buff,
        int bufflen,
        ref TTERMTEXT_INFO info );
```

（2）引数

smsg

エンコードした S10F5 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 S10F5 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。

info

エンコードしたい端末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。

（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

TTERMTEXT_INFO 構造体に格納されている端末メッセージ情報を、S10F5 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。

```
smsg S10F5
  L,2
    tid
    L,n
      A,text1
      A,text2
```

encode ⇐

TTERMTEXT_INFO 構造体

3．10．3．4　DshDecodeS10F5() - S10F5 端末情報を TTERMTEXT_INFO 構造体にデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS10F5(
        DSHMSG *smsg,                    // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TTERMTEXT_INFO *info             // ﾃﾞｺｰﾄﾞしたい端末表示情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS10F5 (
          ByRef smsg As dshdr2.DSHMSG,
          ByRef pinfo As dsh_info.TTERMTEXT_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS10F5(
          ref DSHMSG smsg,
          ref TTERMTEXT_INFO pinfo );
```

（2）引数

smsg

デコードする S10F5 メッセージが格納されているメッセージ情報構造体のポインタです。

info

デコードした端末メッセージ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

smsg に格納されている S10F5 メッセージをデコードし、info で指定された TTERMTEXT_INFO 構造体領域に端末メッセージ情報を格納します。。

smsg S10F5

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

decode ⇨

TTERMTEXT_INFO
構造体

デコード後、info の処理が終了したら、DshFreeTTERMTEXT_INFO()関数を使って構造体内で使用されたメモリを開放してください。

３．10．３．５　DshFreeTTERMTEXT_INFO() - 端末表示情報構造体メモリの開放

（１）呼出書式

[c,C++]

```
API void APIX DshFreeTTERMTEXT_INFO(
      TTERMTEXT_INFO *info       // ﾒﾓﾘを開放したい端末表示情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTTERMTEXT_INFO (
        ByRef info As dsh_info.TTERMTEXT_INFO)
```

[.NET C#]

```
void DshFreeTTERMTEXT_INFO(
        ref TTERMTEXT_INFO info );
```

（２）引数

pinfo

メモリを解放したい端末表示情報構造体のポインタです。

（３）戻り値

なし。

（４）説明

TTERMTEXT_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、TTERMTEXT_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

3．10．3．6　DshCopyTTERMTEXT_INFO() - 端末表示情報構造体メモリのコピー

（1）呼出書式

[c,C++]

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

[.NET VB]

```
Sub DshCopyTTERMTEXT_INFO (
          ByRef dinfo As dsh_info.TTERMTEXT_INFO,
          ByRef sinfo As dsh_info.TTERMTEXT_INFO)
```

[.NET C#]

```
void DshCopyTTERMTEXT_INFO(
          ref TTERMTEXT_INFO dinfo,
          ref TTERMTEXT_INFO sinfo );
```

（2）引数

dinfo

端末表示情報のコピー先構造体メモリのポインタです。

sinfo

コピー元の端末表示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TTERMTEXT_INFO構造体内に格納されている端末表示情報をdinfoに指定されたTERMTEXT_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、DshFreeTTERM_TEXT_INFO()関数を使って開放してください。

３．10．３．７　DshInitTTERMTEXT_INFO – 端末表示情報 TTERMTEXT_INFO の初期設定

（1）呼出書式

[c,C++]

```
API void  APIX DshInitTTERMTEXT_INFO(
      TTERMTEXT_INFO *info,          // TTERMTEXT_INFO 構造体のﾎﾟｲﾝﾀ
      int          tid,              // 端末 ID
      int          count             // 表示ﾃｷｽﾄ行数
);
```

[.NET VB]

```
Sub DshInitTTERMTEXT_INFO (
       ByRef info As dsh_info.TTERMTEXT_INFO,
       ByVal tid As Int32,
       ByVal count As Int32)
```

[.NET C#]

```
void DshInitTTERMTEXT_INFO(
       ref TTERMTEXT_INFO info,
       int tid,
       int count );
```

（2）引数

info

端末表示情報 TTERMTEXT_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。

tid

表示端末 ID です。

count

info に設定する表示テキストの行数です。

（3）戻り値

なし。

（4）説明

本関数は S10F5 メッセージを送信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。
info 内に count で指定された分のテキスト行格納用ポインタリストを設けます。

テキスト行を１行づつ info 内に順番に設定する関数として DshAddTERMTEXT_INFO()が準備されています。

TTERMTEXT_INFO 構造体の使用後は DshFreeTTERMTEXT_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。

３．10．３．８　DshAddTTERMTEXT_INFO() -端末表示テキストの追加

（1）呼出書式

[c,C++]

```
API int APIX DshAddTTERMTEXT_INFO(
        TTERMTEXT_INFO    *info,            // TTERMTEXT_INFO 構造体のﾎﾟｲﾝﾀ
        char              *text             // info 内に加える表示ﾃｷｽﾄ格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshAddTTERMTEXT_INFO (
          ByRef info As dsh_info.TTERMTEXT_INFO,
          ByVal text As String) As Int32
```

[.NET C#]

```
int DshAddTTERMTEXT_INFO(
          ref TTERMTEXT_INFO info,
          byte[] text );
```

（2）引数

info

表示テキストを加える対象の端末表示情報構造体のポインタです。

text

加えるテキスト文字列です。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	ﾃｷｽﾄ行数数が既に指定数を超えている。

（4）説明

先に DshInitTTERMTEXT_INFO()で初期設定された端末表示情報構造体内のテキストリストに text で指定されたテキスト行を１個加えます。

info 内のテキストリスト上に、本関数が実行される順に引数 text で与えられたテキストを追加していきます。

設定後 0 を返却します。

もし、info 内の text_count メンバーで指定された行分のテキストが既に設定済みであった場合は、(-1)を返却します。

3．10．４　ユーザ作成ライブラリ関数

3．10．４．１　DshResponseS10F2() – S10F2 端末要求応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS10F2(
        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
        char    *text,                  // 端末表示ﾒｯｾｰｼﾞ格納ﾎﾟｲﾝﾀ
        int     ackc10                  // S10F2 応答情報格納用構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshResponseS10F2 (
          ByVal eqid As Int32,
          ByVal trid As Int32,
          ByVal text As String,
          ByVal ackc10 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS10F2(
          int eqid,
          uint trid,
          byte[] text,
          int ackc10 );
```

（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

S10F1 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。

text

端末要求の表示メッセージ（文字列)が格納されているポインタです。

ackc10

送信する応答メッセージ ack の値です。

（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

端末要求メッセージ S10F1 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(dsh_ulib.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 ackc10 情報から S10F2 メッセージを組み立て、その後、S10F2 メッセージを送信します。

3．10．４．２　DshResponseS10F4() – S10F4 端末表示応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS10F4(
        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
        char    *text,                  // 端末表示ﾒｯｾｰｼﾞ格納ﾎﾟｲﾝﾀ
        int     ackc10                  // S10F4 応答情報格納用構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshResponseS10F4 (
          ByVal eqid As Int32,
          ByVal trid As Int32,
          ByVal text As String,
          ByVal ackc7 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS10F4(
          int eqid,
          uint trid,
          byte[] text,
          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

S10F3 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。

text

端末表示の表示メッセージ（文字列)が格納されているポインタです。

ackc10

送信する応答メッセージ ack の値です。

（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

端末表示メッセージ S10F3 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(dsh_ulib.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 ackc10 情報から S10F4 メッセージを組み立て、その後、S10F4 メッセージを送信します。

3．10．４．３　DshResponseS10F6() - S10F6 端末表示 (Multi-Block)応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS10F6(
        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
        TTERMTEXT_INFO *info,           // 端末表示 (Multi-Block)ﾒｯｾｰｼﾞ格納ﾎﾟｲﾝﾀ
        int     ackc10                  // S10F6 応答情報格納用構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshResponseS10F6 (
        ByVal eqid As Int32,
        ByVal trid As Int32,
        ByRef info As dsh_info.TTERMTEXT_INFO,
        ByVal ackc10 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS10F6(
        int eqid,
        uint trid,
        ref TTERMTEXT_INFO info,
        int ackc10 );
```

（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

S10F5 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

端末表示（Multi-Block）の表示メッセージ情報が格納されているポインタです。

ackc10

送信する応答メッセージ ack の値です。

（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

端末表示（Multi-Block)メッセージ S10F5 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(dsh_ulib.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 ackc10 情報から S10F6 メッセージを組み立て、その後、S10F6 メッセージを送信します。