DshGemMsgPro　GEM メッセージ・エンコード/デコード

ソフトウェア・ライブラリ

LIB 関数説明書

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

Vol-２／２

・リモートコントロール、拡張リモートコントロール関連
・キャリアアクション、ポート制御関連
・端末表示関連
・スプール関連
・その他の汎用関数

２０１３年９月

株式会社データマップ

文書番号　DshGemMsgPro-13-30332-00

［取り扱い注意］

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

【改訂履歴】

番号	改訂日付	項 目	概 略
1.	2013 年 9 月	初版	

目　次

[GEM-PRO　関連ドキュメント]

GEM-PRO ドキュメント一覧表

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

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

２．エンコード／デコードに使用する構造体の操作関数（Vol-1 からの続き）

２．８　ホスト・リモートコマンド関連関数

ホスト・リモートコマンド関連メッセージとして S2F41, S2F49 の２種類のものがあります。
この２つのメッセージ関連の情報構造体に関連するライブラリ関連関数は下表のとおりです。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTRCMD_INFO()	TRCMD_INFO 構造体の初期設定	S2F41
	DshPutTRCMD_INFO()	同　ﾎｽﾄｺﾏﾝﾄﾞ情報設定	
	DshFreeTRCMD_INFO()	同　内部使用ﾒﾓﾘの解放	
2	DshInitTRCMD_HERR_INFO()	TRCMD_HERR_INFO 構造体の初期設定	S2F42
	DshPutTRCMD_HERR_PARA()	同　ｴﾗｰﾊﾟﾗﾒｰﾀ設定	
	DshFreeTRCMD_HERR_INFO()	同　内部使用ﾒﾓﾘの解放	
3	DshInitTERCMD_INFO()	TERCMD_INFO 構造体の初期設定	S2F49（ERC）
	DshPutTERCMD_INFO()	同　拡張ﾘﾓｰﾄｺﾏﾝﾄﾞ情報設定	
	DshFreeTERCMD_INFO()	同　内部使用ﾒﾓﾘの解放	
4	DshInitTERCMD_HERR_INFO()	TERCMD_HERR_INFO 構造体の初期設定	S2F50
	DshPutTERCMD_HERR_PARA()	同　ｴﾗｰﾊﾟﾗﾒｰﾀ設定	
	DshFreeTERCMD_HERR_INFO()	同　内部使用ﾒﾓﾘの解放	

２．８．１　DshInitTRCMD_INFO() – ホスト・コマンド構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTRCMD_INFO
        TRCMD_INFO  *info,          // TRCMD_INFO 構造体のﾎﾟｲﾝﾀ
        char         *rcmd,         // ﾎｽﾄ・ｺﾏﾝﾄﾞ
        int          cp_count       // 付属ﾊﾟﾗﾒｰﾀ数
);
```

[VB.Net]

```
Sub DshInitTRCMD_INFO
        ByRef info As TRCMD_INFO,
        rcmd     As String,
        cp_count As Integer)
```

[C#]

```
void DshInitTRCMD_INFO
        ref TRCMD_INFO info,
        string  rcmd,
        int     cp_count );
```

（2）引数

info

TRCMD_INFO 構造体のポインタです。

rcmd

ホスト・コマンドです。

cp_count

付属パラメータ数です。

（3）戻り値

なし。

（4）説明

本関数は、TRCMD_INFO 構造体を初期設定するために使用します。
構造体内には、ホスト・コマンドならびに cp_count 分のパラメータ情報を保存します。

構造体の使用が済んだら、DshFree)関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        char        *rcmd;              // rcmd
        int         cp_count;           // parameter count
        TRCMD_PARA  **cp_list;          // paramete list
}TRCMD_INFO;
```

```
typedef struct{
        char        *cpname;            // cpname
        int         cpval_fmt;          // cpval item fmt
        int         cpval_size;         // cpval data array size
        void        *cpval;             // cpval
}TCMD_PARA;
```

２．８．２　DshPutTRCMD_INFO() – パラメータ情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTRCMD_INFO(
        TRCMD_INFO    *info,              // TRCMD_INFO 構造体のﾎﾟｲﾝﾀ
        char     *cpname,                 // 追加するｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ名
        int      cpval_fmt,               // ﾊﾟﾗﾒｰﾀ値のﾌｫｰﾏｯﾄ
        int      cpval_size,              // ﾊﾟﾗﾒｰﾀ値の配列ｻｲｽﾞ
        void     cpval                    // ﾊﾟﾗﾒｰﾀ値格納ﾎﾟｲﾝﾀ
        );
```

[VB.Net]

```
Function DshPutTRCMD_INFO(
        ByRef info    As TRCMD_INFO,
        ByRef cpname As String,
        cpval_fmt     As Integer,
        cpval_size    As Integer,
        cpval         As IntPtr ) As Integer
```

[C#]

```
int DshPutTRCMD_INFO(
        ref TRCMD_INFO info,
        string  cpname,
        int     cpval_fmt,
        int     cpval_size,
        IntPtr  cpval_);
```

（2）引数

info

TRCMD_INFO 構造体のポインタです。

cpname

パラメータ名です。

cpval_fmt

パラメータ値のフォーマットです。

cpval_size

パラメータの配列サイズです。

cpval

パラメータ値のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明したDshInitTRCMD_INFO()関数で初期設定されたinfo内に、1個のパラメータ情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

cp_count 分を超える数のパラメータを追加しようとした場合、戻り値として(-1)が返却されます。

２．８．３　DshFreeTRCMD_INFO() －　ホスト・コマンド構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCMD_INFO
        TRCMD_INFO  *info               // ﾒﾓﾘを開放したいﾎｽﾄ・ｺﾏﾝﾄﾞ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTRCMD_INFO
        ByRef info As TRCMD_INFO )
```

[C#]

```
void DshFreeTRCMD_INFO
        ref TRCMD_INFO info );
```

（2）引数

info

メモリを解放したいホスト・コマンド構造体のポインタです。

（3）戻り値

なし。

（4）説明

TRCMD_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、TRCMD_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。

２．８．４　DshInitTRCMD_HERR_INFO() – ホスト・コマンドエラー情報の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTRCMD_HERR_INFO(
        TRCMD_HERR_INFO  *erinfo,       // TRCMD_HERR_INFO 構造体のﾎﾟｲﾝﾀ
        int         hcack,              // HCACK
        int         err_count           // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTRCMD_HERR_INFO(
        ByRef erinfo As TRCMD_HERR_INFO,
        hcack     As Integer,
        err_count As Integer)
```

[C#]

```
void DshInitTRCMD_HERR_INFO(
        ref TRCMD_HERR_INFO erinfo,
        int     hcack,
        int     err_count );
```

（2）引数

erinfo

TRCMD_HERR_INFO 構造体のポインタです。

hcack

HCACK です。

err_count

エラー情報の保存数です。(THERR_INFO)

（3）戻り値

なし。

（4）説明

本関数は、TRCMD_HERR_INFO 構造体を初期設定するために使用します。
構造体内には、HCACK と err_count の数だけのエラー情報（TRCMD_HERR_INFO 構造体）を保存します。

構造体の使用が済んだら、DshFreeTRCMD_HERR_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        int         hcack;              // B
        int         err_count;
        char        **cpname_list;      // cpname
        int         *cpack_list;
} TRCMD_HERR_INFO;
```

２．８．５　DshPutTRCMD_HERR_INFO() – エラー情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTRCMD_HERR_INFO(
        TRCMD_HERR_INFO   *erinfo,          // TRCMD_HERR_INFO 構造体のﾎﾟｲﾝﾀ
        int     err_code,                   // ｴﾗｰｺｰﾄﾞ
        char    *err_text                   // ｴﾗｰﾃｷｽﾄ
        );
```

[VB.Net]

```
Function DshPutTRCMD_HERR_INFO(
        ByRef erinfo As TRCMD_HERR_INFO,
        err_code As Integer,
        err_text As String) As Integer
```

[C#]

```
int DshPutTRCMD_HERR_INFO(
        ref TRCMD_HERR_INFO erinfo,
        int     err_code,
        string err_text);
```

（2）引数

erinfo

TRCMD_HERR_INFO 構造体のポインタです。

err_code

エラーコードです。

err_text

エラーテキストです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	erinfo が満杯であったため追加できなかった。

（4）説明

先に説明した DshInitTRCMD_HERR_INFO)関数で初期設定された erinfo 内に、1 個のエラー情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

err_count 分を超える数のエラー情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．８．６　DshFreeTRCMD_HERR_INFO() – ホスト・コマンドエラー情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCMD_HERR_INFO(
        TRCMD_HERR_INFO  *erinfo              // ﾒﾓﾘを開放したいｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTRCMD_HERR_INFO(
        ByRef erinfo As TRCMD_HERR_INFO )
```

[C#]

```
void DshFreeTRCMD_HERR_INFO(
        ref TRCMD_HERR_INFO erinfo );
```

（2）引数

erinfo

メモリを解放したいホスト・コマンドエラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

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

２．８．７　DshInitTERCMD_INFO() – 拡張リモート・コマンド構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTERCMD_INFO
        TERCMD_INFO  *info,          // TERCMD_INFO 構造体のﾎﾟｲﾝﾀ
        char         *objspec,       // OBJSPEC
        char         *rcmd,          // 拡張ﾘﾓｰﾄ・ｺﾏﾝﾄﾞ
        int           cp_count       // 付属ﾊﾟﾗﾒｰﾀ数
);
```

[VB.Net]

```
Sub DshInitTERCMD_INFO
        ByRef info As TERCMD_INFO,
        objspec  As String,
        rcmd     As String,
        cp_count As Integer)
```

[C#]

```
void DshInitTERCMD_INFO
        ref TERCMD_INFO info,
        string objspec,
        string rcmd,
        int cp_count );
```

（2）引数

info

TERCMD_INFO 構造体のポインタです。

objspec

OBJSPEC オブジェクトを示す文字列 ID です。

rcmd

拡張リモート・コマンドです。

cp_count

付属パラメータ数です。

（3）戻り値

なし。

（4）説明

本関数は、TERCMD_INFO 構造体を初期設定するために使用します。

構造体内には、オブジェクト ID,　拡張リモート・コマンドならびに cp_count 分のパラメータ情報を保存します。

構造体の使用が済んだら、DshFree)関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        char        *objspec;          // object spec
        char        *rcmd;             // rcmd
        int         cp_count;          // parameter count
        TERCMD_PARA **cp_list;         // paramete list
}TERCMD_INFO;
```

```
typedef struct{
        char        *objspec;          // object spec
        char        *rcmd;             // rcmd
        int         cp_count;          // parameter count
        TERCMD_PARA **cp_list;         // paramete list
}TERCMD_INFO;
```

２．８．８　DshPutTERCMD_INFO() – パラメータ情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTERCMD_INFO(
        TERCMD_INFO   *info,              // TERCMD_INFO 構造体のﾎﾟｲﾝﾀ
        char     *cpname,                 // 追加するｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ名
        int      cpval_fmt,               // ﾊﾟﾗﾒｰﾀ値のﾌｫｰﾏｯﾄ
        int      cpval_size,              // ﾊﾟﾗﾒｰﾀ値の配列ｻｲｽﾞ
        void     cpval                    // ﾊﾟﾗﾒｰﾀ値格納ﾎﾟｲﾝﾀ
        );
```

[VB.Net]

```
Function DshPutTERCMD_INFO(
        ByRef info As TERCMD_INFO,
        ByRef cpname As String,
        cpval_fmt    As Integer,
        cpval_size   As Integer,
        cpval        As IntPtr ) As Integer
```

[C#]

```
int DshPutTERCMD_INFO(
        ref TERCMD_INFO info,
        string  cpname,
        int     cpval_fmt,
        int     cpval_size,
        IntPtr  cpval_);
```

（2）引数

info

TERCMD_INFO 構造体のポインタです。

cpname

パラメータ名です。

cpval_fmt

パラメータ値のフォーマットです。

cpval_size

パラメータの配列サイズです。

cpval

パラメータ値のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明したDshInitTERCMD_INFO()関数で初期設定されたinfo内に、1個のパラメータ情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

cp_count 分を超える数のパラメータを追加しようとした場合、戻り値として(-1)が返却されます。

２．８．９　DshFreeTERCMD_INFO() － 拡張リモート・コマンド構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTERCMD_INFO
        TERCMD_INFO  *info              // ﾒﾓﾘを開放したいﾎｽﾄ・ｺﾏﾝﾄﾞ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTERCMD_INFO
        ByRef info As TERCMD_INFO )
```

[C#]

```
void DshFreeTERCMD_INFO
        ref TERCMD_INFO info );
```

（2）引数

info

メモリを解放したい拡張リモート・コマンド構造体のポインタです。

（3）戻り値

なし。

（4）説明

TERCMD_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、TERCMD_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。

２．８．10　DshInitTERCMD_ERR_INFO() – 拡張リモート・コマンドエラー情報の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTERCMD_ERR_INFO(
        TERCMD_ERR_INFO  *erinfo,       // TERCMD_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        int         hcack,              // HCACK
        int         err_count           // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTERCMD_ERR_INFO(
        ByRef erinfo As TERCMD_ERR_INFO,
        hcack     As Integer,
        err_count As Integer)
```

[C#]

```
void DshInitTERCMD_ERR_INFO(
        ref TERCMD_ERR_INFO erinfo,
        int     hcack,
        int     err_count );
```

（2）引数

erinfo

TERCMD_ERR_INFO 構造体のポインタです。

hcack

HCACK です。

err_count

エラー情報の保存数です。(TERR_INFO)

（3）戻り値

なし。

（4）説明

本関数は、TERCMD_ERR_INFO 構造体を初期設定するために使用します。
構造体内には、HCACK と err_count の数だけのエラー情報（TERCMD_ERR_INFO 構造体）を保存します。

構造体の使用が済んだら、DshFreeTERCMD_ERR_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        int         hcack;              // B
        int         err_count;
        char        **cpname_list;      // cpname
        int         *cpack_list;
} TERCMD_ERR_INFO;
```

２．８．11　DshPutTERCMD_ERR_PARA() – エラー情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTERCMD_ERR_PARA(
        TERCMD_ERR_INFO   *erinfo,          // TERCMD_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        int     order,                      // cpname_list, cpack_list 配列位置
        int     err_code,                   // ｴﾗｰｺｰﾄﾞ
        char    *err_text                   // ｴﾗｰﾃｷｽﾄ
        );
```

[VB.Net]

```
Function DshPutTERCMD_ERR_PARA(
        ByRef erinfo As TERCMD_ERR_INFO,
        order     As Integer,
        err_code  As Integer,
        err_text  As String) As Integer
```

[C#]

```
int DshPutTERCMD_ERR_PARA(
        ref TERCMD_ERR_INFO erinfo,
        int     order,
        int     err_code,
        string  err_text);
```

（2）引数

erinfo

TERCMD_ERR_INFO 構造体のポインタです。

order

cpname_list, cpack_list 配列リストの設定位置

err_code

エラーコードです。

err_text

エラーテキストです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	erinfo が満杯であったため追加できなかった。

（4）説明

先に説明した DshInitTERCMD_ERR_INFO()関数で初期設定された erinfo 内に、1 個のエラー情報をエラー情報の指定された配列位置に設定します。

既に設定済であったり、err_count 位置を超える指定した場合、戻り値として(-1)が返却されます。

２．８．12　DshFreeTERCMD_ERR_INFO() – 拡張リモート・コマンドエラー情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTERCMD_ERR_INFO(
        TERCMD_ERR_INFO  *erinfo              // ﾒﾓﾘを開放したいｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTERCMD_ERR_INFO(
        ByRef erinfo As TERCMD_ERR_INFO )
```

[C#]

```
void DshFreeTERCMD_ERR_INFO(
        ref TERCMD_ERR_INFO erinfo );
```

（2）引数

erinfo

メモリを解放したい拡張リモート・コマンドエラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TERCMD_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。
開放した後、TERCMD_ERR_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

２．９　キャリア・アクション関連関数

キャリア・アクション関連メッセージとして S3F17 があります。
このメッセージ関連の情報構造体に関連するライブラリ関連関数は下表のとおりです。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTCACT_PARA()	TCACT_PARA 構造体の初期設定	S3F17
	DshPutTCACT_PARA()	同　1 個の TCACT_PARA の設定	
	DshFreeTCACT_PARA()	同　内部使用ﾒﾓﾘの解放	
	DshInitTCACT_PARA()	TCACT_PARA 構造体の初期設定	
	DshPutTCACT_SLOT_INFO()	ｷｬﾘｱｺﾝﾃﾝﾄの設定　(slot 情報)	
	DshFreeTCACT_PARA()	同　内部使用ﾒﾓﾘの解放	
2	DshInitTCACT_ERR_INFO()	TCACT_ERR_INFO 構造体の初期設定	S3F18
	DshPutTCACT_ERR_INFO()	同　1 個のｴﾗｰ情報を設定	S3F24
	DshFreeTCACT_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	S3F26

２．９．１　DshInitTCACT_INFO() – キャリア・アクション構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTCACT_INFO
        TCACT_INFO  *info,          // TCACT_INFO 構造体のﾎﾟｲﾝﾀ
        int         action,         // ｱｸｼｮﾝ index 値
        char        *carid,         // ｷｬﾘｱ ID
        int         ptn,            // ﾎﾟｰﾄ番号
        int         attr_count      // 付属ﾊﾟﾗﾒｰﾀ数
);
```

[VB.Net]

```
Sub DshInitTCACT_INFO
        ByRef info As TCACT_INFO,
        action     As Integer,
        carid      As String,
        ptn        As Integer,
        attr_count As Integer)
```

[.NET CCA_]

```
void DshInitTCACT_INFO
        ref TCACT_INFO info,
        int     action,
        string  carid,
        int     ptn,
        int     attr_count );
```

（2）引数

info

TCACT_INFO 構造体のポインタです。

action

アクション・インデクスで指定します。　本節の（6）を参照

carid

OBJSPEC オブジェクトを示す文字列 ID です。

ptn

ポート番号です。

attr_count

属性数です。

（3）戻り値

なし。

（4）説明

本関数は、TCACT_INFO 構造体を初期設定するために使用します。

構造体内には、アクション・インデクス番号、キャリア ID,　attr_count 分の属性情報を保存します。

構造体の使用が済んだら、DshFreeTCACT_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

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

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

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

```
typedef struct{
        int         count;
        int         *slot_list;
} TCACT_SLOT_INFO;
```

（6）アクション・インデクスと属性インデクス表

①アクション・インデクス表

action_x(ｲﾝﾃﾞｸｽ)	ｱｸｼｮﾝ・ｲﾝﾃﾞｸｽ記号	Action 名
0	CA_Bind	Bind
1	CA_CancelBind	CancelBind
2	CA_CancelCarrier	CancelCarrier
3	CA_CancelCarrierAtPort	CancelCarrierAtPort
4	CA_CancelCarrierNotification	CancelCarrierNotification
5	CA_CancelCarrierOut	CancelCarrierOut
6	CA_CarrierIn	CarrierIn
7	CA_CarrierNotification	CarrierNotification
8	CA_CarrierOut	CarrierOut
9	CA_CarrierReCreate	CarrierReCreate
10	CA_CarrierRelease	CarrierRelease
11	CA_ProceedWithCarrier	ProceedWithCarrier

②属性インデクス表

attr_x	Index 記号	属性名	ﾃﾞｰﾀの型
0	CA_ObjType	ObjType	文字列
1	CA_ ObjId	ObjId	文字列
2	CA_Capacity	Capacity	数値
3	CA_CarrierAccessingStatus	CarrierAccessingStatus	数値
4	CA_CarrierIDStatus	CarrierIDStatus	数値
5	CA_ContentMap	ContentMap	TPORTG_CONENT 構造体
6	CA_LocationID	LocationID	文字列
7	CA_SlotMap	SlotMap	TPORTG_SLOT_INFO1 構造体
8	CA_SlotMapStatus	SlotMapStatus	数値
9	CA_SubStrateCount	SubStrateCount	数値
10	CA_Usage	Usage	文字列

２．９．２　DshPutTCACT_INFO() – パラメータ情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTCACT_INFO(
        TCACT_INFO    *info,          // TCACT_INFO 構造体のﾎﾟｲﾝﾀ
        TCACT_PARA    *attrdata       // 追加する属性ﾃﾞｰﾀ構造体のﾎﾟｲﾝﾀ
        );
```

[VB.Net]

```
Function DshPutTCACT_INFO(
        ByRef info     As TCACT_INFO,
        ByRef attrdata As TCACT_PARA ) As Integer
```

[.NET CCA_]

```
int DshPutTCACT_INFO(
        ref TCACT_INFO  info,
        ref TCACT_PARA  attrdata _);
```

（2）引数

info

TCACT_INFO 構造体のポインタです。

attrdata

属性データ構造体のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明したDshInitTCACT_INFO()関数で初期設定されたinfo内に、1個のパラメータ情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

attr_count 分を超える数のパラメータを追加しようとした場合、戻り値として(-1)が返却されます。

２．９．３　DshFreeTCACT_INFO() －　キャリア・アクション情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTCACT_INFO
      TCACT_INFO  *info              // ﾒﾓﾘを開放したいｷｬﾘｱｱｸｼｮﾝ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTCACT_INFO
        ByRef info As TCACT_INFO )
```

[.NET CCA]

```
void DshFreeTCACT_INFO
        ref TCACT_INFO info );
```

（2）引数

info

メモリを解放したいキャリア・アクション情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TCACT_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、TCACT_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。

２．９．４　DshInitTCACT_PARA() – キャリア・属性構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTCACT_PARA
      TCACT_PARA  *info,          // TCACT_PARA 構造体のﾎﾟｲﾝﾀ
      int         attr_x,         // 属性 index 値
      void        attrdata        // 属性ﾃﾞｰﾀﾎﾟｲﾝﾀ(format はattr_x によって決まる。)
);
```

[VB.Net]

```
Sub DshInitTCACT_PARA
      ByRef info As TCACT_PARA,
      attr_x   As Integer,
      attrdata As IntPtr )
```

[.NET CCA_]

```
void DshInitTCACT_PARA
      ref TCACT_PARA info,
      int     attr_x,
      IntPtr  attrdata
      );
```

（2）引数

info

TCACT_PARA 構造体のポインタです。

attr_x

属性・インデクスで指定します。　２．８．７－(6)参照

attrdata

属性値が格納されている領域のポインタです。attr_x の値によって、数値、文字列、構造体のポインタなどが決まる。

attr_x = CA_ContentMap の場合は、TCACT_CONTENT 構造体のポインタになります。

attr_X = CA_SlotMap の場合は、 TCACT_SLOT_INFO 構造体のポインタになります。

（3）戻り値

なし。

（4）説明

本関数は、TCACT_PARA 構造体を初期設定するために使用します。

構造体内には、attr_x、属性インデクス番号と属性値データが保存されている領域のポインタが渡されます。

attrdata がどのような値のポインタになるかについては、２．８．７－(6)の一覧表を参照ください。

構造体の使用が済んだら、DshFreeTCACT_PARA()関数によって内部で使用したメモリを解放してください。

２．９．５　DshFreeTCACT_PARA() －　キャリア・アクション属性情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTCACT_PARA
      TCACT_PARA  *info              // ﾒﾓﾘを開放したいｷｬﾘｱｱｸｼｮﾝ属性情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTCACT_PARA
        ByRef info As TCACT_PARA )
```

[.NET CCA]

```
void DshFreeTCACT_PARA
        ref TCACT_PARA info );
```

（2）引数

info

メモリを解放したいキャリア・アクション属性情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TCACT_PARA 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、TCACT_PARA 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。

２．９．６　DshPutTCACT_CONTENT() – キャリア・コンテントマップ情報構造体への追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTCACT_CONTENT(
        TCACT_PARA    *info,          // TCACT_CONTENT 構造体のﾎﾟｲﾝﾀ
        int   order,                  // TCACT_CONTENT 内の lotid, substid 配列の位置
        char *lotid,                  // LOTID
        char *suubstid                // SUBSTID
        );
```

[VB.Net]

```
Function DshPutTCACT_CONTENT(
        ByRef info As TCACT_PARA,
        order    As Integer,
        lotid    As String,
        substid As String ) As Integer
```

[.NET CCA_]

```
int DshPutTCACT_CONTENT(
        ref TCACT_PARA  info,
        int      order,
        string   lotid,
        string   substid);
```

（2）引数

info

TCACT_PARA 構造体のポインタです。

order

TCACT_CONTENT 構造体の lotid, substid 配列位置です。

lotid

ロット ID です。(NULL の場合は値なしになります。)

substid

基板 ID です。(NULL の場合は値なしになります。)

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明した TCACT_PARA 構造体の attr_index の値が CA_ContentMap の場合、attrdata メンバーの値として TCACT_CONTENT 構造体のポインタが設定されます。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

attr_count 分を超える数のパラメータを追加しようとした場合、戻り値として(-1)が返却されます。

２．９．７　DshPutTCACT_SLOT_INFO() – キャリア・スロットマップ情報構造体への追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTCACT_SLOT_INFO(
        TCACT_PARA    *info,          // TCACT_SLOT_INFO 構造体のポインタ
        int   order,                  // TCACT_SLOT_INFO 内の slot_list 配列の位置
        int   mapdata                 // map data
        );
```

[VB.Net]

```
Function DshPutTCACT_SLOT_INFO(
        ByRef info As TCACT_PARA,
        order      As Integer,
        mapdata    As Integer ) As Integer
```

[.NET CCA_]

```
int DshPutTCACT_SLOT_INFO(
        ref TCACT_PARA  info,
        int order,
        int mapdata );
```

（2）引数

info

TCACT_PARA 構造体のポインタです。

order

TCACT_SLOT_INFO 構造体の lotid, substid 配列位置です。

mapdata

マップデータ（有・無情報）です。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明した TCACT_PARA 構造体の attr_index の値が CA_SLOT_INFO の場合、attrdata メンバーの値として TCACT_SLOT_INFO 構造体のポインタが設定されます。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

attr_count 分を超える数のパラメータを追加しようとした場合、戻り値として(-1)が返却されます。

２．９．８　DshInitTCACT_ERR_INFO() – キャリア・アクション情報エラー情報の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTCACT_ERR_INFO(
        TCACT_ERR_INFO  *erinfo,        // TCACT_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        int             caack,          // CAACK
        int             err_count       // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTCACT_ERR_INFO(
        ByRef erinfo As TCACT_ERR_INFO,
        caack     As Integer,
        err_count As Integer)
```

[.NET CCA_]

```
void DshInitTCACT_ERR_INFO(
        ref TCACT_ERR_INFO erinfo,
        int     caack,
        int     err_count );
```

（2）引数

erinfo

TCACT_ERR_INFO 構造体のポインタです。

caack

CAACK です。

err_count

エラー情報の保存数です。(TERR_INFO)

（3）戻り値

なし。

（4）説明

本関数は、TCACT_ERR_INFO 構造体を初期設定するために使用します。
構造体内には、CAACK と err_count の数だけのエラー情報（TERR_INFO 構造体）を保存します。

構造体の使用が済んだら、DshFreeTCACT_ERR_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

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

２．９．９　DshPutTCACT_ERR_PARA() – エラー情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTCACT_ERR_PARA(
        TCACT_ERR_INFO   *erinfo,          // TCACT_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        int     order,                     // err_list 配列位置
        int     err_code,                  // ｴﾗｰｺｰﾄﾞ
        char    *err_text                  // ｴﾗｰﾃｷｽﾄ
        );
```

[VB.Net]

```
Function DshPutTCACT_ERR_PARA(
        ByRef erinfo As TCACT_ERR_INFO,
        order    As Integer,
        err_code As Integer,
        err_text As String) As Integer
```

[.NET CCA]

```
int DshPutTCACT_ERR_PARA(
        ref TCACT_ERR_INFO erinfo,
        int     order,
        int     err_code,
        string err_text);
```

（2）引数

erinfo

TCACT_ERR_INFO 構造体のポインタです。

order

err_list 配列リストの設定位置

err_code

エラーコードです。

err_text

エラーテキストです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	erinfo が満杯であったため追加できなかった。

（4）説明

先に説明した DshInitTCACT_ERR_INFO()関数で初期設定された erinfo 内に、1 個のエラー情報を err_list の指定された配列位置に設定します。

既に設定済であったり、err_count 位置を超える指定した場合、戻り値として(-1)が返却されます。

２．９．10　DshFreeTCACT_ERR_INFO() – キャリア・アクション情報エラー情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTCACT_ERR_INFO(
        TCACT_ERR_INFO  *erinfo              // ﾒﾓﾘを開放したいｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTCACT_ERR_INFO(
        ByRef erinfo As TCACT_ERR_INFO )
```

[.NET CCA_]

```
void DshFreeTCACT_ERR_INFO(
        ref TCACT_ERR_INFO erinfo );
```

（2）引数

erinfo

メモリを解放したいキャリア・アクション情報エラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TCACT_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。
開放した後、TCACT_ERR_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

２．10　ポートコントロール関連関数

ポートコントロール関連メッセージとして S3F23, S3F23, S3F25, S3F27 の 4 種類のものがあります。
この 4 つのメッセージ関連の情報構造体に関連するライブラリ関連関数は下表のとおりです。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTPORTG_INFO()	TPORTG_INFO 構造体の初期設定	S3F23
	DshPutTPORG_INFO()	同　1 個のﾎﾟｰﾄｸﾞﾙｰﾌﾟの設定	
	DshFreeTPORTG_INFO()	同　内部使用ﾒﾓﾘの解放	
2	DshInitTPORT_INFO()	TPORT_INFO 構造体の初期設定	S3F25
	DshPutTPORT_INFO()	同　1 個のﾎﾟｰﾄの設定	
	DshFreeTPORT_INFO()	同　内部使用ﾒﾓﾘの解放	
3	DshInitTACCESS_INFO()	TACCESS_INFO 構造体の初期設定	S3F27
	DshPutTACCESS_INFO()	同　1 個のﾎﾟｰﾄの設定	
	DshFreeTACCESS_INFO()	同　内部使用ﾒﾓﾘの解放	
4	DshInitTACCESS_ERR_INFO()	TACCESS_ERR_INFO 構造体の初期設定	S3F28
	DshPutTACCESS_ERR_INFO()	同　1 個のｴﾗｰ情報を設定	
	DshFreeTACCESS_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	

S3F24, S3F26 メッセージ関連関数は、S3F18 メッセージと同様の関数を使用します。
２．９．８～２．９．10 を参照ください。

２．10．１　DshInitTPORTG_INFO() – ポートグループ・アクション構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPORTG_INFO
        TPORTG_INFO  *info,            // TPORTG_INFO 構造体のﾎﾟｲﾝﾀ
        char         *action,          // ｱｸｼｮﾝ名
        char         *portgroupname,   // ﾎﾟｰﾄｸﾞﾙｰﾌﾟ名
        int          para_count        // 付属ﾊﾟﾗﾒｰﾀ数
);
```

[VB.Net]

```
Sub DshInitTPORTG_INFO
        ByRef info As TPORTG_INFO,
        action          As String,
        portgroupname   As String,
        para_count      As Integer)
```

[.NET CCA]

```
void DshInitTPORTG_INFO
        ref TPORTG_INFO info,
        string     action,
        string     portgroupname,
        int        para_count );
```

（2）引数

info

TPORTG_INFO 構造体のポインタです。

action

ポートグループで実行するアクション名です。

portgroupname

ポートグループ名です。

para_count

付属パラメータ数です。

（3）戻り値

なし。

（4）説明

本関数は、TPORTG_INFO 構造体を初期設定するために使用します。

構造体内には、アクション名、ポートグループ ID,　パラメータ情報を保存します。

構造体の使用が済んだら、DshFreeTPORTG_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

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

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

２．10．２　DshPutTPORTG_INFO() – パラメータ情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTPORTG_INFO(
        TPORTG_INFO   *info,              // TPORTG_INFO 構造体のﾎﾟｲﾝﾀ
        char     *pname,                  // 追加するﾊﾟﾗﾒｰﾀ名
        int      fmt,                     // ﾊﾟﾗﾒｰﾀ値のﾌｫｰﾏｯﾄ
        int      asize,                   // ﾊﾟﾗﾒｰﾀ値の配列ｻｲｽﾞ
        void     pval                     // ﾊﾟﾗﾒｰﾀ値格納ﾎﾟｲﾝﾀ
        );
```

[VB.Net]

```
Function DshPutTPORTG_INFO(
        ByRef info As TPORTG_INFO,
        ByRef pname As String,
        fmt   As Integer,
        asize As Integer,
        pval_ As IntPtr ) As Integer
```

[C#]

```
int DshPutTPORTG_INFO(
        ref TPORTG_INFO info,
        string  pname,
        int     fmt,
        int     asize,
        IntPtr  pval_);
```

（2）引数

info

TPORTG_INFO 構造体のポインタです。

pname

パラメータ名です。

fmt

パラメータ値のフォーマットです。

asize

パラメータの配列サイズです。

pval

パラメータ値のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明したDshInitTPORTG_INFO()関数で初期設定されたinfo内に、1個のパラメータ情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

pn_count 分を超える数のパラメータを追加しようとした場合、戻り値として(-1)が返却されます。

２．10．３　DshFreeTPORTG_INFO() －　ポートグループ・アクション情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPORTG_INFO
        TPORTG_INFO  *info              // ﾒﾓﾘを開放したいｱｸｼｮﾝ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPORTG_INFO
        ByRef info As TPORTG_INFO )
```

[.NET CCA_]

```
void DshFreeTPORTG_INFO
        ref TPORTG_INFO info );
```

（2）引数

info

メモリを解放したいポートグループ・アクション情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPORTG_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、TPORTG_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。

２．10．４　DshInitTPORT_INFO() – ポート・アクション構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPORT_INFO
        TPORT_INFO  *info,          // TPORT_INFO 構造体のﾎﾟｲﾝﾀ
        char        *action,        // ｱｸｼｮﾝ名
        int         ptn,            // ﾎﾟｰﾄ番号
        int         para_count      // 付属ﾊﾟﾗﾒｰﾀ数
);
```

[VB.Net]

```
Sub DshInitTPORT_INFO
        ByRef info As TPORT_INFO,
        action     As String,
        ptn        As Integer,
        para_count As Integer)
```

[.NET CCA]

```
void DshInitTPORT_INFO
        ref TPORT_INFO info,
        string   action,
        int      ptn,
        int      para_count );
```

（2）引数

info

TPORT_INFO 構造体のポインタです。

action

ポートで実行するアクション名です。

ptn

ポート番号です。

para_count

付属パラメータ数です。

（3）戻り値

なし。

（4）説明

本関数は、TPORT_INFO 構造体を初期設定するために使用します。

構造体内には、アクション名、ポート番号、para_count 分のパラメータ情報を保存します。

構造体の使用が済んだら、DshFreeTPORT_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

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

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

２．10．５　DshPutTPORT_INFO() – パラメータ情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTPORT_INFO(
        TPORT_INFO    *info,              // TPORT_INFO 構造体のﾎﾟｲﾝﾀ
        char     *pname,                  // 追加するﾊﾟﾗﾒｰﾀ名
        int      fmt,                     // ﾊﾟﾗﾒｰﾀ値のﾌｫｰﾏｯﾄ
        int      asize,                   // ﾊﾟﾗﾒｰﾀ値の配列ｻｲｽﾞ
        void     pval                     // ﾊﾟﾗﾒｰﾀ値格納ﾎﾟｲﾝﾀ
        );
```

[VB.Net]

```
Function DshPutTPORT_INFO(
        ByRef info As TPORT_INFO,
        ByRef pname As String,
        int   fmt,
        int   asize,
        pval_ As IntPtr ) As Integer
```

[C#]

```
int DshPutTPORT_INFO(
        ref TPORT_INFO info,
        string  pname,
        int     fmt,
        int     asize,
        IntPtr  pval_);
```

（2）引数

info

TPORT_INFO 構造体のポインタです。

pname

パラメータ名です。

fmt

パラメータ値のフォーマットです。

asize

パラメータの配列サイズです。

pval

パラメータ値のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明したDshInitTPORT_INFO()関数で初期設定されたinfo内に、1個のパラメータ情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

pn_count 分を超える数のパラメータを追加しようとした場合、戻り値として(-1)が返却されます。

２．10．６　DshFreeTPORT_INFO() －　ポート・アクション情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPORT_INFO
        TPORT_INFO  *info               // ﾒﾓﾘを開放したいｱｸｼｮﾝ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPORT_INFO
        ByRef info As TPORT_INFO )
```

[.NET CCA]

```
void DshFreeTPORT_INFO
        ref TPORT_INFO info );
```

（2）引数

info

メモリを解放したいポート・アクション情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPORT_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、TPORT_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。

２．10．７　DshInitTACCESS_INFO() – ポートアクセスモード構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTACCESS_INFO
        TACCESS_INFO  *info,          // TACCESS_INFO 構造体のﾎﾟｲﾝﾀ
        int           accessmode,     // ｱｸｾｽﾓｰﾄﾞ
        int           port_count      // ﾎﾟｰﾄ数
);
```

[VB.Net]

```
Sub DshInitTACCESS_INFO(
        ByRef info    As TACCESS_INFO,
        accessmode    As Integer,
        port_count    As Integer)
```

[.NET CCA]

```
void DshInitTACCESS_INFO
        ref TACCESS_INFO info,
        int      accessmode,
        int      port_count );
```

（2）引数

info

　　TACCESS_INFO 構造体のポインタです。

accessmode

　　ポートに指定するアクセスモードです。

port_count

　　変更したいポート数です。

（3）戻り値

なし。

（4）説明

本関数は、TACCESS_INFO 構造体を初期設定するために使用します。
構造体内には、アクセスモード、ポート数分のポート番号を保存します。

構造体の使用が済んだら、DshFreeTACCESS_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

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

２．10．８　DshPutTACCESS_INFO() – ポート番号の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTACCESS_INFO(
        TACCESS_INFO    *info,          // TACCESS_INFO 構造体のﾎﾟｲﾝﾀ
        int     ptn                     // 追加するﾎﾟｰﾄ番号です。
        );
```

[VB.Net]

```
Function DshPutTACCESS_INFO(
        ByRef info As TACCESS_INFO,
        ptn    As Integer ) As Integer
```

[C#]

```
int DshPutTACCESS_INFO(
        ref TACCESS_INFO info,
        int  ptn_);
```

（2）引数

info

TACCESS_INFO 構造体のポインタです。

ptn

ptn_list に追加するポート番号です。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明した DshInitTACCESS_INFO()関数で初期設定された info 内の ptn_list に、1 個のポート番号を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

port_count 分を超える数のポートを追加しようとした場合、戻り値として(-1)が返却されます。

２．10．９　DshFreeTACCESS_INFO() －　ポートアクセスモード情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTACCESS_INFO
      TACCESS_INFO  *info              // ﾒﾓﾘを開放したいｱｸｾｽﾓｰﾄﾞ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTACCESS_INFO
        ByRef info As TACCESS_INFO )
```

[.NET CCA]

```
void DshFreeTACCESS_INFO
        ref TACCESS_INFO info );
```

（2）引数

info

メモリを解放したいポートアクセスモード情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TACCESS_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、TACCESS_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。

２．10．10　DshInitTACCESS_ERR_INFO() – の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTACCESS_ERR_INFO(
      TACCESS_ERR_INFO  *erinfo,     // TACCESS_ERR_INFO 構造体のﾎﾟｲﾝﾀ
      int         caack,             // CAACK
      int         err_count          // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTACCESS_ERR_INFO(
      ByRef erinfo As TACCESS_ERR_INFO,
      caack     As Integer,
      err_count As Integer)
```

[.NET CCA_]

```
void DshInitTACCESS_ERR_INFO(
      ref TACCESS_ERR_INFO erinfo,
      int   caack,
      int   err_count );
```

（2）引数

erinfo

　　TACCESS_ERR_INFO 構造体のポインタです。

caack

　　CAACK です。

err_count

　　エラー情報の保存数です。(TERR_INFO)

（3）戻り値

なし。

（4）説明

本関数は、TACCESS_ERR_INFO 構造体を初期設定するために使用します。
構造体内には、CAACK と err_count の数だけのエラー情報（TACCESS_ERR_PORT 構造体）を保存します。

構造体の使用が済んだら、DshFreeTACCESS_ERR_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        int         caack;              // B
        int         err_count;
        TACCESS_ERR_PORT   **err_list;
} TCACT_ERR_INFO;
```

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

２．10．11　DshPutTACCESS_ERR_INFO() – アクセスモード変更エラー情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTACCESS_ERR_INFO(
        TACCESS_ERR_INFO    *erinfo,            // TACCESS_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        int     port,                           // ﾎﾟｰﾄ番号
        int     err_code,                       // ｴﾗｰｺｰﾄﾞ
        char    *err_text                       // ｴﾗｰﾃｷｽﾄ
        );
```

[VB.Net]

```
Function DshPutTACCESS_ERR_INFO(
        ByRef erinfo As TACCESS_ERR_INFO,
        port      As Integer,
        err_code As Integer,
        err_text As String) As Integer
```

[.NET CCA]

```
int DshPutTACCESS_ERR_INFO(
        ref TACCESS_ERR_INFO erinfo,
        int     port,
        int     err_code,
        string err_text);
```

（2）引数

erinfo

TACCESS_ERR_INFO 構造体のポインタです。

port

TACCESS_ERR_PORT 構造体に設定するポート番号です。

err_code

エラーコードです。

err_text

エラーテキストです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	erinfo が満杯であったため追加できなかった。

（4）説明

先に説明した DshInitTACCESS_ERR_INFO()関数で初期設定された erinfo 内に、1 個のポートエラー情報をエに追加します。

既に設定済であったり、err_count 位置を超える指定した場合、戻り値として(-1)が返却されます。

２．10．12　DshFreeTACCESS_ERR_INFO() – TACCESS_ERR_INFO 情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTACCESS_ERR_INFO(
      TACCESS_ERR_INFO  *erinfo              // ﾒﾓﾘを開放したいｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTACCESS_ERR_INFO(
        ByRef erinfo As TACCESS_ERR_INFO )
```

[.NET CCA_]

```
void DshFreeTACCESS_ERR_INFO(
        ref TACCESS_ERR_INFO erinfo );
```

（2）引数

erinfo

メモリを解放したいポートグループ・アクション情報エラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TACCESS_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。
開放した後、TACCESS_ERR_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

２．11　端末表示要求関連関数

端末表示関連メッセージとして S10F1、S10F3、S10F5 があります。ここでは、表示情報を構造体内に保存して使用する S10F5 メッセージに使用する TTERMTEXT_INFO 構造体操作に使用する関数を説明します。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTTERMTEXT_INFO()	TTERMTEXT_INFO 構造体の初期設定	S10F5
	DshPutTTERMTEXT_INFO()	同　文字列を設定	
	DshFreeTTERMTEXT_INFO()	同　内部使用ﾒﾓﾘの解放	

２．11．１　DshInitTTERMTEXT_INFO() – 端末表示情報構造体の初期設定

（1）呼出書式

[c,C++]

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

[VB.Net]

```
Sub DshInitTTERMTEXT_INFO
        ByRef info As TTERMTEXT_INFO,
        tid   As Integer,
        count As Integer)
```

[.NET CCA_]

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

（2）引数

info

TTERMTEXT_INFO 構造体のポインタです。

tid

端末 ID です。

count

表示テキスト数です。

（3）戻り値

なし。

（4）説明

本関数は、TTERMTEXT_INFO 構造体を初期設定するために使用します。
構造体内には、端末 ID,　表示テキストとその行数を保存します。

構造体の使用が済んだら、DshFreeTTERMTEXT_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        int     tid;
        int     text_count;     // # of text
        char    **text_list;
}TTERMTEXT_INFO;
```

２．11．２　DshPutTTERMTEXT_INFO() – テキストの追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTTERMTEXT_INFO(
        TTERMTEXT_INFO   *info,          // TTERMTEXT_INFO 構造体のﾎﾟｲﾝﾀ
        char     *text                   // 追加するﾊﾟﾗﾒｰﾀ名
        );
```

[VB.Net]

```
Function DshPutTTERMTEXT_INFO(
        ByRef info As TTERMTEXT_INFO,
        ByRef text As String
         ) As Integer
```

[C#]

```
int DshPutTTERMTEXT_INFO(
        ref TTERMTEXT_INFO info,
        string  text );
```

（2）引数

info

TTERMTEXT_INFO 構造体のポインタです。

text

表示テキストです。(1 行分)

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明した DshInitTTERMTEXT_INFO()関数で初期設定された info 内に、テキスト 1 行分を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

count 分を超える数のパラメータを追加しようとした場合、戻り値として(-1)が返却されます。

２．11．３　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構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTTERMTEXT_INFO
        ByRef info As TTERMTEXT_INFO )
```

[.NET CCA_]

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

（2）引数

info

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。

２．12　スプール関連関数

スプール関連メッセージとして S2F43 があります。
スプールメッセージ関連の情報構造体に関連するライブラリ関連関数は下表のとおりです。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitItemGet()	TSPOOL_INFO 構造体の初期設定	S2F43 (SPOOL)
	DshPutTSPOOL_INFO()	同 stream を 1 個設定	
	DshFreeTSPOOL_INFO()	同 内部使用ﾒﾓﾘの解放	
	DshInitTSTRE_INFO()	TSTRE_INFO 構造体の初期設定	
	DshPutTSTRE_INFO()	同 function を 1 個設定	
	DshFreeTSTRE_INFO()	同 内部使用ﾒﾓﾘの解放	
2	DshInitTSPOOL_ERR_INFO()	TSPOOL_ERR_INFO 構造体の初期設定	S2F44
	DshPutTSPOOL_ERR_INFO()	同 stream1 個分の設定	
	DshFreeTSPOOL_ERR_INFO()	同 内部使用ﾒﾓﾘの解放	
	DshInitTSTRE_ERR_INFO()	TSTRE_ERR_INFO 構造体の初期設定	
	DshPutTSTRE_ERR_INFO()	同 function 1 個分の設定	
	DshFreeTSTRE_ERR_INFO()	同 内部使用ﾒﾓﾘの解放	

２．12．１　DshInitItemGet() – スプール情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitItemGet(
        TSPOOL_INFO  *info,          // TSPOOL_INFO ｽﾌﾟｰﾙ情報構造体のﾎﾟｲﾝﾀ
        int           s_count        // ｽﾄﾘｰﾑ数
);
```

[VB.Net]

```
Sub DshInitItemGet(
        ByRef info As TSPOOL_INFO,
        s_count  As Integer)
```

[C#]

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

（2）引数

info

TSPOOL_INFO 構造体のポインタです。

s_count

スプール対象にしたい Stream の数です。

（3）戻り値

なし。

（4）説明

本関数は、TSPOOL_INFO 構造体を初期設定するために使用します。

TSPOOL_INFO 内に s_count 分のストリームのスプール情報を格納するための処理を行います。

構造体の使用が済んだら、DshFreeTSPOOL_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

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

```
typedef struct{                       // S2F43 からの取得情報
        int     stream;               // stream
        int     f_count;              // # of functions
        int     *func_list;           // fuction list
} TSTRE_INFO;
```

２．12．２　DshPutTSPOOL_INFO() – スプール対象ストリーム情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTSPOOL_INFO(
        TSPOOL_INFO    *info,          // TSPOOL_INFO ｽﾌﾟｰﾙ情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        TSTRE_INFO     *sinfo          // TSTRE_INFO ｽﾄﾘｰﾑ情報構造体のﾎﾟｲﾝﾀｰ
        );
```

[VB.Net]

```
Function DshPutTSPOOL_INFO(
        ByRef info  As TSPOOL_INFO,
        ByRef sinfo As TSTRE_INFO) As Integer
```

[C#]

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

（2）引数

info

TSPOOL_INFO スプール構造体のﾎﾟｲﾝﾀです。

sinfo

Stream のスプール情報を保存する TSTRE_INFO 構造体のﾎﾟｲﾝﾀです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明した DshInitItemGet()関数で初期設定された info 内に、1 個の Stream のスプール情報を追加します。

追加によって、本関数が呼び出される順番にに保存されます。
DshInitItemGet()関数で設定した s_count 分だけの Stream のスプール情報を加えることがします。

s_count 分を超える数のレポート情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．12．３　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  *info              // ﾒﾓﾘを開放したいスプール情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

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

[C#]

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

（2）引数

info

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。
info が NULL ならば、何も処理しません。

２．12．４　DshInitTSTRE_INFO() – ストリームのスプール情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTSTRE_INFO(
      TSTRE_INFO  *info,          // TSTRE_INFO  ストリームのスプール情報構造体のﾎﾟｲﾝﾀ
      int          stream,        // Stream
      int          f_count        // Function の数
);
```

[VB.Net]

```
Sub DshInitTSTRE_INFO(
      ByRef info As TSTRE_INFO,
      stream     As Integer,
      ByVal      f_count As Integer)
```

[C#]

```
void DshInitTSTRE_INFO(
      ref TSTRE_INFO info,
      int  stream,
      int  f_count );
```

（2）引数

info

　　TSTRE_INFO 構造体のポインタです。

stream

　　Stream です。

f_count

　　stream で指定された中の Function の数です。

（3）戻り値

　なし。

（4）説明

　本関数は、TSTRE_INFO 構造体を初期設定するために使用します。
　TSTRE_INFO 内に f_count 分の stream に指定されたストリームに属する Function を格納するための処理を行います。

　構造体の使用が済んだら、DshFreeTSTRE_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{                         // S2F43 からの取得情報
        int     stream;                 // stream
        int     f_count;                // # of functions
        int     *func_list;             // fuction list
} TSTRE_INFO;
```

２．12．５　DshPutTSTRE_INFO() – ファンクションの追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTSTRE_INFO(
        TSTRE_INFO    *info,        // TSTRE_INFO ストリームのスプール値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int    f_code               // 追加する Function Code
        );
```

[VB.Net]

```
Function DshPutTSTRE_INFO(
        ByRef info As TSTRE_INFO,
        f_code      As Integer ) As Integer
```

[C#]

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

（2）引数

info

TSTRE_INFO 構造体のポインタです。

f_code

追加する Function Code です。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明した DshInitTSTRE_INFO()関数で初期設定された info 内のに、1 個の Fnciton コードを追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。
DshInitTSTRE_INFO()関数で設定した f_count 分だけの Function を加えることがします。

f_count 分を超える数のレポート情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．12．６　DshFreeTSTRE_INFO() – ストリームの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 DshFreeTSTRE_INFO(
        TSTRE_INFO  *info               // ﾒﾓﾘを開放したいｽﾄﾘｰﾑのｽﾌﾟｰﾙ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTSTRE_INFO(
        ByRef info As TSTRE_INFO)
```

[C#]

```
void DshFreeTSTRE_INFO(
        ref TSTRE_INFO info );
```

（2）引数

info

メモリを解放したいストリームの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

TSTRE_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、TSTRE_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。

２．12．７　DshInitTSPOOL_ERR_INFO() – スプールエラー情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTSPOOL_ERR_INFO(
      TSPOOL_ERR_INFO  *info,        // TSPOOL_ERR_INFO ｽﾌﾟｰﾙｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
      int         rsack,             // RSACK
      int          err_count         // TSTRE_ERR_INFO 構造体情報保存数
);
```

[VB.Net]

```
Sub DshInitTSPOOL_ERR_INFO(
      ByRef info As TSPOOL_ERR_INFO,
      rsack     As Integer,
      err_count As Integer )
```

[C#]

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

（2）引数

info

TSPOOL_ERR_INFO 構造体のポインタです。

err_count

stream 単位のエラー情報の数です。

（3）戻り値

なし。

（4）説明

本関数は、TSPOOL_ERR_INFO 構造体を初期設定するために使用します。

TSPOOL_ERR_INFO 内に err_count 分のストリーム単位のエラー情報を格納するための処理を行います。

構造体の使用が済んだら、DshFreeTSPOOL_ERR_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

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

```
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;
```

２．12．８　DshPutTSPOOL_ERR_INFO() – ストリーム単位のエラー情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTSPOOL_ERR_INFO(
        TSPOOL_ERR_INFO    *info,       // TSPOOL_ERR_INFO ｽﾌﾟｰﾙ情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        TSTRE_ERR_INFO     *errinfo     // TSTRE_ERR_INFO ｽﾄﾘｰﾑ情報構造体のﾎﾟｲﾝﾀｰ
        );
```

[VB.Net]

```
Function DshPutTSPOOL_ERR_INFO(
        ByRef info    As TSPOOL_ERR_INFO,
        ByRef errinfo As TSTRE_ERR_INFO) As Integer
```

[C#]

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

（2）引数

info

TSPOOL_ERR_INFO スプール構造体のポインタです。

errinfo

スプール単位のエラー情報を保存する TSTRE_ERR_INFO 構造体のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明した DshInitTSPOOL_ERR_INFO()関数で初期設定された info 内の stre_list に、1 個の Stream エラー情報を追加します。

追加によって、本関数が呼び出される順番にに保存されます。
DshInitTSPOOL_ERR_INFO()関数で設定した err_count 分だけのエラー情報を加えることがします。

err_count 分を超える数のレポート情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．12．９　DshFreeTSPOOL_ERR_INFO() – スプールエラー情報構造体メモリの開放

（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ｽﾌﾟｰﾙｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

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

[C#]

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

（2）引数

info

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_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。
開放した後、TSPOOL_ERR_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。

２．12．10　DshInitTSTRE_ERR_INFO() – ストリーム単位のエラー情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTSTRE_ERR_INFO(
        TSTRE_ERR_INFO  *info,          // TSTRE_ERR_INFO  ｽﾄﾘｰﾑ単位のｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
        int          strack,            // STRACK
        int          stream,            // Stream
        int         f_count             // Function の数
);
```

[VB.Net]

```
Sub DshInitTSTRE_ERR_INFO(
        ByRef info As TSTRE_ERR_INFO,
        strack     As Integer,
        stream     As Integer,
        f_count    As Integer)
```

[C#]

```
void DshInitTSTRE_ERR_INFO(
        ref TSTRE_ERR_INFO info,
        int  strack,
        int  stream,
        int  f_count );
```

（2）引数

info

TSTRE_ERR_INFO 構造体のポインタです。

strack

STRACK です。

stream

Stream です。

f_count

stream で指定された中の Function の数です。

（3）戻り値

なし。

（4）説明

本関数は、TSTRE_ERR_INFO 構造体を初期設定するために使用します。
TSTRE_ERR_INFO 内に STRACK と、f_count 分の Function を格納するための処理を行います。

構造体の使用が済んだら、DshFreeTSTRE_ERR_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
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;
```

２．12．11　DshPutTSTRE_ERR_INFO() – ファンクションの追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTSTRE_ERR_INFO(
        TSTRE_ERR_INFO    *info,        // TSTRE_ERR_INFO 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int          f_code,            // 追加する Function Code
        int          func_err           // ﾌｧﾝｸｼｮﾝに対するｴﾗｰｺｰﾄﾞ
        );
```

[VB.Net]

```
Function DshPutTSTRE_ERR_INFO(
        ByRef info As TSTRE_ERR_INFO,
        f_code    As Integer,
        func_err As Integer ) As Integer
```

[C#]

```
int DshPutTSTRE_ERR_INFO(
        ref TSTRE_ERR_INFO info,
        int f_code,
        int func_err );
```

（2）引数

info

TSTRE_ERR_INFO 構造体のﾎﾟｲﾝﾀです。

f_code

追加する Function Code です。

func_err

Function に対するエラーコードです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info が満杯であったため追加できなかった。

（4）説明

先に説明した DshInitTSTRE_ERR_INFO()関数で初期設定された info 内のに、1 個の Fnciton コードと、それに対するエラーコードを追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。
DshInitTSTRE_ERR_INFO()関数で設定した f_count 分だけのエラー情報を加えることがします。

f_count 分を超える数のレポート情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．12．12　DshFreeTSTRE_ERR_INFO() – ストリームの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 DshFreeTSTRE_ERR_INFO(
        TSTRE_ERR_INFO  *info              // ﾒﾓﾘを開放したいｽﾄﾘｰﾑのｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTSTRE_ERR_INFO(
        ByRef info As TSTRE_ERR_INFO)
```

[C#]

```
void DshFreeTSTRE_ERR_INFO(
        ref TSTRE_ERR_INFO info );
```

（2）引数

info

メモリを解放したいストリームのスプールエラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TSTRE_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。
開放した後、TSTRE_ERR_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．その他のライブラリ関数

以下の関数が提供されます。

関数一覧表

	関数名	機能	備考
1	DV_InitItemGet()	DSHMSG 構造体をﾃﾞｰﾀｱｲﾃﾑ設定のために初期化する。	
2	DV_GetItem()	ﾃﾞｰﾀｱｲﾃﾑをﾊﾞｯﾌｧから 1 個だけ取得する。	
3	DV_InitItemPut()	DSHMSG 構造体をﾃﾞｰﾀｱｲﾃﾑ取得のための初期化する。	
4	DV_PutItem()	ﾃﾞｰﾀｱｲﾃﾑをﾊﾞｯﾌｧに 1 個設定する。	
5	dsh_get_item_name()	ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞの名前を取得する。	
6	dsh_get_item_unit_size()	ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞに対するﾃﾞｰﾀﾊﾞｲﾄ長を取得する。	
7	dsh_edit_vdval()	ﾃﾞｰﾀｱｲﾃﾑのﾃﾞｰﾀ値を文字列に変換する。	

３．１　SECS-II メッセージデータアイテム設定／取得関連構造体、エラー情報

メッセージの組立、展開処理は、ユーザが与えるメッセージ用バッファと処理を管理するための構造体を使用することになります。使用する構造体は、DSHDR2　HSMS 通信ドライバーも使用している DSHMSG 構造体です。

３．１．１　DSHMSG 構造体

C, C++

```
typedef struct{
    UINT    stream;             // stream id
    UINT    function;           // function id
    UINT    wbit;               // wait bit
    int     length;             // msg length
    UCHAR   *buffer;            // msg buffer
    int     error;              // error
    int     next;               // next item
    UCHAR   *txtp;              // (内部処理用)
    int     txtc;               // (   〃      )
    int     work[2];            // (内部処理用)
} DSHMSG, *PDSHMSG;
```

C#

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

VB.Net

```
Public Structure DSHMSG
  Public stream As Integer      ' stream id
  Public [function] As Integer  ' function id
  Public wbit As Integer        ' wait bit
  Public length As Integer      ' msg length
  Public buffer As IntPtr       ' msg buffer
  Public [error] As Integer     ' error
  Public [next] As Integer      ' next item
  Public txtp As Integer        ' (内部処理用)
  Public txtc As Integer        ' (   〃      )
  Public work1 As Integer
  Public work2 As Integer
End Structure
```

ユーザが直接 DSHMSG 構造体にアクセスするメンバーは次の通りです。

メッセージ作成の際に必ずユーザが設定する必要があります。

（1）stream, function, wbit
（2）buffer　－　テキスト用バッファポインタ
（3）length　－　受信メッセージの場合は受信したテキストの長さ

３．１．２　関数で使用するデータアイテムコードと表現

データアイテムの表現の定義は、c,C++はDSH.ｈ，c#は dshdr2.cs で定義されています。
データアイテムコードは、説明の中では、データフォーマット(format)として使用します。

GEM-PRO 表現	ｺｰﾄﾞ(16 進)
ICODE_L	0
ICODE_A	0x10
ICODE_J	0x11
ICODE_B	0x08
ICODE_I1	0x19
ICODE_I2	0x1a
ICODE_I4	0x1c
ICODE_I8	0x14
ICODE_U1	0x29
ICODE_U2	0x2a
ICODE_U4	0x2c
ICODE_U8	0x24
ICODE_BOOLEAN	0x09
ICODE_END(end)	0x3d

３．１．３　関数戻り値

エラー記号	値	意味
EI_NORMAL	0	正常
EI_ERROR	(-1)	length　＜ 0 であった。
EI_TYPE_ERR	(-17)	ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞがｴﾗｰ
EI_ITEM_POS_ERROR	(-18)	Put できなかった。
EI_ARRAY_SIZE_ERR	(-19)	ﾊﾞｯﾌｧｻｲｽﾞが不足していた。

３．１．４　DSHMSG 構造体の使い方（プログラミング例）

上に示す通り、DSHMSG 内には、stream, function, wait bit, buffer, length がありますが、ここに SECS-II メッセージを保存します。
メッセージの TEXT は、buffer で示すメモリに内に格納し、length の値がテキストのバイト長になります。

DSHMSG 構造体を使って、送信メッセージへのデータアイテムの設定(put)、受信メッセージからのデータアイテムの取得(get)を行います。

使用する関数は、次の４つです。

DV_InitItemGet()	：	データアイテム取得のための初期設定
DV_GetItem()	：	データアイテムを１個づつ取得する。
DV_InitItemPut()	：	データアイテム設定のための初期設定
DV_PutItem()	：	データアイテムを１個づつ設定する。

下に、S5F1 の送信メッセージの生成、受信メッセージの処理について例を示します。
（ここでは、各関数の戻り値のチェックは省略しています。）

送信ﾒｯｾｰｼﾞ作成時のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ

```
DSHMSG  info;
BYTE    buff[128];
int     ALCD = 0x80 + 20;
int     ALID = 100;
char    *ALTX = “ALARM-100 .......“;
info.steam = 5;
info.function = 1;
info.wbit = 1;
info.buffer = buff;
info.length = 128;
DV_InitItemPut( &info );
DV_PutItem( &info, ICODE_B, &ALCD< 1 );
DV_PutItem( &info, ICODE_U4, &ALID, 1 );
DV_PutItem( &info, ALTX, ICODE_A, 40 );

 これで、info.buffer に S5F1 のﾒｯｾｰｼﾞ・ﾃｷｽﾄが 作成されます。
```

受信ﾒｯｾｰｼﾞの処理時のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ

```
DSHMSG  info;
BYTE    buff[128];
int     ALCD = 0;
int     ALID = 0;
char    ALTX[41];

info.buffer に受信した MSG TEXT
DV_InitItemGet( &info );
DV_GetItem( &info, ICODE_B, &ALCD< 1 );
DV_GetItem( &info, ICODE_U4, &ALID, 1 );
DV_GetItem( &info, ALTX, ICODE_A, 41 );

 これで、それぞれのﾃﾞｰﾀｱｲﾃﾑの処理ができます。
```

３．２　データアイテム設定／取得関連関数

SECS-II メッセージの送受信処理にあたって、送信時はメッセージの組立、受信時にはメッセージの解読をする必要があります。

GEM-PRO のユーザは、API 関数を使って GEM に準拠するメッセージのエンコード、デコードを GEM-PRO に任せれば処理してくれます。

しかし、GEM 以外のシステム固有のメッセージの送受信が必要になることがあります。その際にメッセージ組立、展開するために使用する関数について説明します。

関数一覧表

	関数名	機能	備考
1	DV_InitItemGet()	DSHMSG 構造体をﾃﾞｰﾀｱｲﾃﾑ設定のために初期化する。	
2	DV_GetItem()	ﾃﾞｰﾀｱｲﾃﾑをﾊﾞｯﾌｧから 1 個だけ取得する。	
3	DV_InitItemPut()	DSHMSG 構造体をﾃﾞｰﾀｱｲﾃﾑ取得のための初期化する。	
4	DV_PutItem()	ﾃﾞｰﾀｱｲﾃﾑをﾊﾞｯﾌｧに 1 個設定する。	

３．２．１　DV_InitItemGet() – データアイテム取得用初期設定

（1）呼出書式

[c,C++]

```
API int APIX DV_InitItemGet(
      DSHMSG  *info                  // DSHMSG 構造体のポインタ
);
```

[VB.Net]

```
Sub DV_InitItemGet(
      ByRef info As DSHMSG
)
```

[C#]

```
void DV_InitItemGet(
      ref DSHMSG info
);
```

（2）引数

info

DSHMSG 構造体のポインタです。

（3）戻り値

なし。

（4）説明

本関数は、DSHMSG 構造体を初期設定するために使用します。

info 内の内部情報をテキスト（buffer）の先頭のデータ・アイテムを取得できるようにします。

もし、info 内の length が = 0 の場合は、next = ICODE_END に設定します。
（next = ICODE_END の場合は、もう取得するデータアイテムが無いことを意味します。）

また、info 内の length > 0 の場合は、next にテキストの先頭のデータアイテムコードを設定します。
この next は、DV_GetItem()関数によってデータアイテムを取得後に、次のデータアイテムのコードに更新されます。

３．２．２　DV_GetItem() – データアイテム取得

（1）呼出書式

[c,C++]

```
API int APIX DV_GetItem(
        DSHMSG  *info ,             // DSHMSG 構造体のﾎﾟｲﾝﾀ
        int     icode,              // 取出したいﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞ
        void    *data,              // 取出したﾃﾞｰﾀを格納するためのﾊﾞｯﾌｧ
        int     size                // data の配列ｻｲｽﾞ
);
```

[VB.Net]

```
Sub DV_GetItem(
        ByRef info As DSHMSG,
        icode   As Integer,
        data    As IntPtr,
        size    As Integer
)
```

[C#]

```
void DV_GetItem(
        ref DSHMSG info,
        int     icode,
        IntPtr  data,
        int     size
);
```

（2）引数

info

DSHMSG 構造体のポインタです。この buffer に length バイトのテキストが保存されています。

icode

期待しているデータアイテムのコードです。

data

データを格納するバッファのポインタです。

size

データバッファの配列サイズです。(取得最大サイズ)

（3）戻り値

No.	戻り値	意　味
1.	> 0	取得できたﾃﾞｰﾀ数(指定ｱｲﾃﾑｺｰﾄﾞ単位)
2.	= 0	ｱｲﾃﾑﾃﾞｰﾀは無かった。
3.	EI_TYPE_ERR	指定されたｺｰﾄﾞのｱｲﾃﾑではなかった。
4.	EI_ARRAY_SIZE_ERROR	size より大きいｱｲﾃﾑﾃﾞｰﾀｻｲｽﾞであった。

（4）説明

本関数は、DSHMSG 構造体の buffer から icode で指定されたコードのデータアイテムのデータを 1 個取得します。

データアイテムの取得は、先頭から順に取り出すことになります。
D_InitItemGet()直後は現取得位置は先頭のデータアイテム位置になります。

正常に取得した後は、取得位置を次のアイテム位置に進めます。(アイテム取得位置は取得の都度自動的に次の位置に進められます。)

現取得位置のデータアイテムの有無、アイテムが指定コードと同じかどうか，データアイテムのサイズがsize 以内であるかどうか、を調べます。そして、取得条件が満たされればそのデータを取得し、戻り値として取得したデータの数を返却します。
例えば、2 バイト整数のデータアイテムを 2 個、即ち 4 バイト取得した場合には、戻り値=2 になります。
戻り値=0 は、長さ=0 のデータアイテムが取得されたことを意味します。
なお、データアイテム，icode が ICODE_L(リスト)の場合は、データ格納ポインタ data は使用されません。

取得条件が満たされなければ、[戻り値] の表に示す通りの戻り値を返却します。この場合取得位置を更新しません。

本関数実行後、DSHMSG msg 内の next メンバーに 次に D_GetItem()関数で取得するデータアイテムコード（ICODE_A など）が設定されます。
このデータアイテムが ICODE_END であれば、これ以上のデータアイテムのリストが無いことを意味します。

３．２．３　DV_InitItemPut() – データアイテム設定用初期設定

（1）呼出書式

[c,C++]

```
API int APIX DV_InitItemPut(
      DSHMSG  *info                  // DSHMSG 構造体のポインタ
);
```

[VB.Net]

```
Sub DV_InitItemPut(
      ByRef info As DSHMSG
)
```

[C#]

```
void DV_InitItemPut(
      ref DSHMSG info
);
```

（2）引数

info

DSHMSG 構造体のポインタです。

（3）戻り値

なし。

（4）説明

本関数は、DSHMSG 構造体を初期設定するために使用します。

info 内の内部情報をテキストの（buffer）先頭にデータ・アイテムを設定できるようにします。

その後、DV_PutItem()関数によってデータアイテムを設定した後に、設定位置を次の位置に更新します。

３．２．４　DV_PutItem() – データアイテム取得

（1）呼出書式

[c,C++]

```
API int APIX DV_PutItem(
        DSHMSG  *info ,             // DSHMSG 構造体のﾎﾟｲﾝﾀ
        int     icode,              // 取出したいﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞ
        void    *data,              // 取出したﾃﾞｰﾀを格納するためのﾊﾞｯﾌｧ
        int     size                // data の配列ｻｲｽﾞ
);
```

[VB.Net]

```
Sub DV_PutItem(
        ByRef info As DSHMSG,
        icode   As Integer,
        data    As IntPtr,
        size    As Integer
)
```

[C#]

```
void DV_PutItem(
        ref DSHMSG info,
        int     icode,
        IntPtr  data,
        int     size
);
```

（2）引数

info

DSHMSG 構造体のポインタです。この buffer に length バイトのテキストが保存します。

icode

設定するデータアイテムのコードです。

data

データが格納されているバッファのポインタです。

size

データの配列サイズです。

（3）戻り値

No.	戻り値	意　味
1.	＞0	設定したﾃﾞｰﾀ数(指定ｱｲﾃﾑｺｰﾄﾞ単位)
2.	= 0	ｱｲﾃﾑｺｰﾄﾞを設定したがﾃﾞｰﾀは無かった。
3.	EI_TYPE_ERROR	指定されたｱｲﾃﾑｺｰﾄﾞは正しくなかった。
4.	EI_ITEM_POS_ERROR	設定位置が見つからなかった。
5.	EI_ARRAY_SIZE_ERR	ﾊﾞｯﾌｧｵｰﾊﾞｰﾌﾛｰ(ﾃﾞｰﾀが格納領域に納まらない)
6.	EI_ERROR	buffer ﾎﾟｲﾝﾀ値ｴﾗｰその他のｴﾗｰを検出した。

（4）説明

本関数は、DSHMSG 構造体の buffer から icode で指定されたコードのデータアイテムのデータを 1 個設定します。

データアイテムの設定は、バッファの先頭から順に行います。

本関数は、msg 内の buffer が指定する領域の現設定位置へ、icode で指定されたアイテムコードと data に準備されたデータアイテムを size 個分だけ設定します。

設定位置は D_InitItemPut()によってテキストの先頭に設定されます。
正常に設定した後は、設定位置を次のアイテム位置に進めます。(アイテム設定位置は設定の都度自動的に次の位置に進められます。) また、msg 内の length には設定後のテキストのバイト長が設定されます。

指定されたデータコードとメッセージ内の buffer ポインタの妥当性などを調べます。その結果、問題がなければ data 内のデータアイテムを n 個分 msg 内の buffer 現設定位置に設定します。
戻り値は、実際に設定したデータ数です。
引数 n の値は =0 の値を指定することができます。

設定条件が満たされなければ、（3）**戻り値** の表に示す通りの戻り値を返却します。この場合設定位置を更新しません。

３．３　その他の関数

SECS-II データアイテムの編集などの関数があります。

関数一覧表

	関数名	機能	備考
1	dsh_get_item_name()	ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞの名前を取得する。	
2	dsh_get_item_unit_size()	ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞに対するﾃﾞｰﾀﾊﾞｲﾄ長を取得する。	
3	dsh_edit_vdval()	ﾃﾞｰﾀｱｲﾃﾑのﾃﾞｰﾀ値を文字列に変換する。	

データアイテムコードと名前とデータの単位バイトサイズは次表のとおりです。

データアイテム表

	ﾃﾞｰﾀｱｲﾃﾑｺｰﾄﾞ記号	名前	ﾊﾞｲﾄ長
1	ICODE_L	“L”	1
2	ICODE_B	“B”	1
3	ICODE_BOOLEAN	“Boolean”	1
4	ICODE_A	“A”	1
5	ICODE_J	“J”	1
6	ICODE_I1	“I1”	1
7	ICODE_I2	“I2”	2
8	ICODE_I4	“I4”	4
9	ICODE_I8	“I8”	8
10	ICODE_U1	“U1”	1
11	ICODE_U2	“U2”	2
12	ICODE_U4	“U4”	4
13	ICODE_U8	“U8”	8
14	ICODE_F4	“F4”	4
15	ICODE_F8	“F8”	8

３．３．１　dsh_get_item_name() – データアイテムコードの名前取得

（1）呼出書式

[c,C++]

```
API char* APIX dsh_get_item_name(
      int  icode
);
```

[VB.Net]

```
Function  dsh_get_item_name(
      icode  As Integer
) As String
```

[C#]

```
string dsh_get_item_name(
      int     icode
);
```

（2）引数

icode

名前を取得したいデータアイテムのコードです。

（3）戻り値

名前のポインタを返却します。

（4）説明

本関数は、データアイテムコードの名前を取得します。

null(=0)で終わる文字列のポインタが返却されます。

なお、定義されていないコードに対しては、NULL を返却します。

３．３．２　dsh_get_item_unit_size() – データアイテムデータのバイト長取得

（1）呼出書式

[c,C++]

```
API int APIX dsh_get_item_unit_size(
      int  icode
);
```

[VB.Net]

```
Function  dsh_get_item_unit_size(
      icode  As Integer
) As Integer
```

[C#]

```
int   dsh_get_item_unit_size(
      int     icode
);
```

（2）引数

icode

バイト長を取得したいデータアイテムのコードです。

（3）戻り値

icode のデータアイテムコードのバイト長を返却します。

（３．３のデータアイテム表を参照ください）

（4）説明

本関数は、データアイテムコードのデータのバイト長を取得します。

なお、定義されていないコードに対しては、0 を返却します。

３．３．３　dsh_edit_vdval() – データアイテムデータの文字列への編集

（1）呼出書式

[c,C++]

```
API char APIX dsh_edit_vdval(
        char  *buff,
        int   fmt,
        void  *sval,
        int   hex
);
```

[VB.Net]

```
Function  dsh_edit_vdval(
        buff  As  IntPtr,
        fmt   As  Integer,
        sval  As  IntPtr,
        hex   As Integer
) As String
```

[C#]

```
string  dsh_edit_vdval(
        IntPtr  buff,
        int     fmt,
        IntPtr  sval,
        int     hex
);
```

（2）引数

buff

編集結果を格納するためのバッファです。
編集結果（文字列）を格納するのに十分な領域が必要です。

fmt

sval のデータのフォーマット(表-3.3 のデータアイテムコード表参照)

sval

編集対象のデータ値が格納されている領域のポインタです。

hex

整数のデータ（B, Boolean を含め）の表現形式を指定します。
0 = 10 進表示、1 = 16 進表示

（3）戻り値

編集結果の文字列を返却します。

（4）説明

本関数は、sval に fmt のフォーマットで格納されているデータを文字列に変換します。
整数のデータ（B, Boolean を含め）の場合、hex に指定された値によって、10 進または 16 進で表現します。