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-7／１５

３．13　RCP レシピ情報アクセスサービス関数

２００９年6月

株式会社データマップ

文書番号　DSHGEM-LIB-09-30327-02

[取り扱い注意]

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

【改訂履歴】

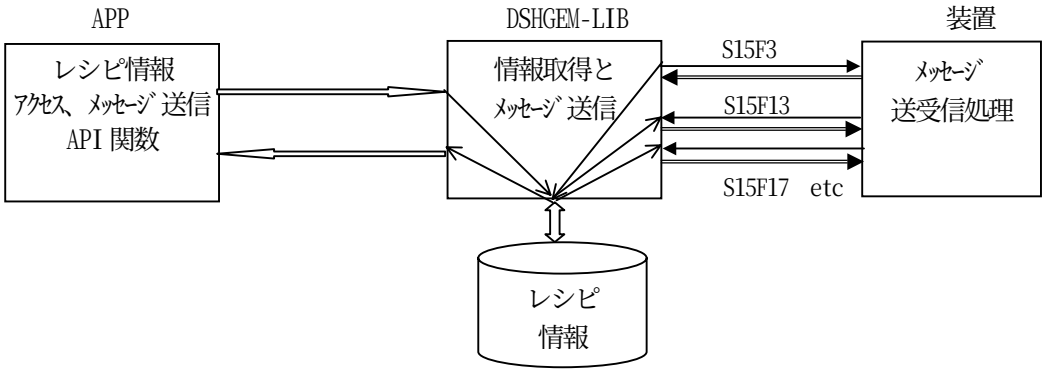
番号	改訂日付	項 目	概 略
1.	2009.6	改訂版	以前の DSHGEM-LIB-07-3032x-00 を全面改訂 .Net VB2008, C#2008 対応関数の説明を追加した。

目　次

(VOL－８に続く)

３．13　RCP レシピ情報アクセスサービス関数

ここで述べるレシピ情報は、DSHGEM-LIB が管理します。従って、APP はこれらの情報をアクセスと関連メッセージを送信するために以下の DSHGEM-LIB API 関数を使用します。



（1）情報アクセスと送信 API 関数

レシピ情報のアクセスとホストへのメッセージ送信に関連するサービスのための API 関数名は一覧表のとおりです。

	API 関数名	機能
1	GemAllocRcpInfo()	ﾚｼﾋﾟ情報領域を割当て登録します。。
2	GemSetRcpInfo()	ﾚｼﾋﾟ情報を設定・変更します。。
3	GemSetRcpInfoX()	ｲﾝﾃﾞｸｽ指定でﾚｼﾋﾟ情報を設定・変更します。。
4	GemGetRcpInfo()	RCPID 指定でﾚｼﾋﾟ情報を取得します。。
5	GemGetRcpInfoX()	ﾚｼﾋﾟｲﾝﾃﾞｸｽ指定でﾚｼﾋﾟ情報を取得します。。
6	GemDelRcpInfo()	RCPID 指定でﾚｼﾋﾟ情報を削除します。。
7	GemDelRcpInfoX()	ﾚｼﾋﾟｲﾝﾃﾞｸｽ指定でﾚｼﾋﾟ情報を削除します。。
8	GemSetRcpState()	RCPID 指定で RCPID の状態を設定します。。
9	GemSetRcpStateX()	ﾚｼﾋﾟｲﾝﾃﾞｸｽ指定で RCPID の状態を設定します。。
10	GemGetRcpState()	RCPID 指定で RCPID の状態を取得します。。
11	GemGetRcpStateX()	ﾚｼﾋﾟｲﾝﾃﾞｸｽ指定で RCPID の状態を取得します。。
12	GemGetRcpList()	ﾚｼﾋﾟ ID の一覧ﾘｽﾄを取得します。。
13	GemGetRcpId()	指定したﾚｼﾋﾟｲﾝﾃﾞｸｽの RCPID を取得します。。
14	GemGetRcpIdIndex()	指定した RCPID の情報ｲﾝﾃﾞｸｽを取得します。。
15	GemSendS15F3()	ﾚｼﾋﾟﾈｰﾑｽﾍﾟｰｽのｱｸｼｮﾝ要求をします。。
16	GemSendS15F5()	ﾚｼﾋﾟﾈｰﾑｽﾍﾟｰｽのﾘﾈｰﾑ要求をします。。
17	GemSendS15F7()	ﾚｼﾋﾟｽﾍﾟｰｽ要求をします。。
18	GemSendS15F9()	ﾚｼﾋﾟｽﾃｰﾀｽを要求します。。
19	GemSendS15F13()	ﾚｼﾋﾟ情報をﾎｽﾄに送信します。。
20	GemSendS15F17()	ﾚｼﾋﾟ情報をﾎｽﾄから取得します。。

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

（2）ライブラリ関数

他に APP が使用できるレシピ情報処理用 API 関数として、以下の関数があります。

	API 関数名	機能
1	DshDecodeS15F3()	S15F3 のﾒｯｾｰｼﾞ内ﾚｼﾋﾟﾈｰﾑｱｸｼｮﾝ情報を TRCP_ACT_INFO 構造体にﾃﾞｺｰﾄﾞ収納するための関数です。
2	DshEncodeS15F3()	TRCP_ACT_INFO 構造体のﾚｼﾋﾟﾈｰﾑｱｸｼｮﾝ情報を S15F3 ﾒｯｾｰｼﾞにｴﾝｺｰﾄﾞするための関数です。
3	DshDecodeS15FRsp()	S15F3，S15F5，S15F13，S15F15，S15F17 の応答メッセージを TRCP_ERR_INFO 構造体にデコードします。
4	DshFreeTRCP_ACT_INFO()	ﾚｼﾋﾟﾈｰﾑｱｸｼｮﾝ情報が格納されている TRCP_ACT_INFO 構造体の内部で使用されているﾒﾓﾘを開放するための関数です。
5	DshCopyTRCP_ACT_INFO()	TRCP_ACT_INFO のﾚｼﾋﾟﾈｰﾑｱｸｼｮﾝ情報を別の構造体にｺﾋﾟｰします。
6	DshMakeS15F3Response()	S15F4 応答ﾒｯｾｰｼﾞを生成するための関数。u_s15f3.c で使用します。
7	DshDecodeS15F5()	S15F5 のﾒｯｾｰｼﾞ内ﾚｼﾋﾟﾘﾈｰﾑ情報を TRCP_RENAME_INFO 構造体にﾃﾞｺｰﾄﾞ収納するための関数です。
8	DshFreeTRCP_RENAME_INFO()	ﾚｼﾋﾟﾘﾈｰﾑ情報が格納されている TRCP_RENAME_INFO 構造体の内部で使用されているﾒﾓﾘを開放するための関数です。
9	DshCopyTRCP_RENAME_INFO()	TRCP_RENAME_INFO のﾚｼﾋﾟﾘﾈｰﾑ情報を別の構造体にｺﾋﾟｰします。
10	DshMakeS15F5Response()	S15F6 応答ﾒｯｾｰｼﾞを生成するための関数。u_s15f5.c で使用します。
11	DshDecodeS15F8()	S15F7 の応答メッセージを TRCP_S15F8_INFO 構造体にデコードします。
12	DshFreeTRCP_S15F8_INFO()	TRCP_S15F8_INFO 構造体内に使用されているﾒﾓﾘを開放します。
13	DshDecodeS15F10()	S15F9 の応答メッセージを TRCP_S15F10_INFO 構造体にデコードします。
14	DshFreeTRCP_S15F10_INFO()	TRCP_S15F10_INFO 構造体内に使用されているﾒﾓﾘを開放します。
15	DshDecodeS15F13()	S15F13 のﾒｯｾｰｼﾞ内レシピ情報を TRCP_INFO 構造体にﾃﾞｺｰﾄﾞ収納するための関数です。
16	DshEncodeS15F13()	TRCP_INFO 構造体のﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を S15F13 ﾒｯｾｰｼﾞにｴﾝｺｰﾄﾞするための関数です。
17	DshFreeTRCP_INFO()	レシピ情報が格納されている TRCP_INFO 構造体の内部で使用されているﾒﾓﾘを開放するための関数です。
18	DshCopyTRCP_INFO()	TRCP_INFO のﾚｼﾋﾟ情報を別の構造体にｺﾋﾟｰします。
19	DshInitRcpInfo()	ﾚｼﾋﾟ情報構造体 TRCP_INFO を初期設定します。。
20	DshPutRcpPara()	ﾚｼﾋﾟﾊﾟﾗﾒｰﾀ情報を TRCP_INFO 内に設定します。。
21	DshMakeS15F13Response()	S15F14 応答ﾒｯｾｰｼﾞを生成するための関数。u_s15f13.c で使用します。

22	DshFreeTRCP_ERR_INFO()	TRCP_ERR_INFO 構造体内に使用されているﾒﾓﾘを開放します。
23	DshDecodeS15F18()	S15F17 の応答メッセージを TRCP_S15F18_INFO 構造体にデコードします。
24	DshFreeTRCP_S15F18_INFO()	TRCP_S15F18_INFO 構造体内に使用されているﾒﾓﾘを開放します。
25	DshInitTRCP_ERR_INFO()	TRCP_ERR_INFO 内に rmack とｴﾗｰ情報の個数を設定します。
26	DshPutTRCP_ERR_INFO()	TRCP_ERR_INFO 内の err_list 上に１個のｴﾗｰ情報(errcode, errtext)を追加します。

（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	DshResponseS15F4()	S15F3 レシピネームスペースアクション応答
2	DshResponseS15F6()	S15F5 レシピネームスペースリネーム応答
3	DshResponseS15F14()	S15F13 レシピ生成要求応答

3．13．1　使用する情報格納構造体

レシピ情報を操作する関数は、共通の情報格納のための TRCP_INFO 構造体を使用します。
レシピ情報関連の構造体は下記のとおりです。

（1）TRCP_INFO – Recipe Information S15F13

```
typedef struct{
        int         index;
        int         state;          // ﾚｼﾋﾟ状態
        char        *name;          // name
        char        *rcpspec;       // rcpid
        int         para_count;     // # of pparameter
        TRCP_PARA   **para_list;    // parameter list
        char        *rcpbody;
        }TRCP_INFO;                 // Recipe Information
```

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

（2）TRCP_PARA – Recipe Parameter Information - S15F13

```
typedef struct{
        char        *rcpparnm;      // para name
        int         par_fmt;        // format
        int         par_size;       // array size
        void        *rcpparval;     // para value;
        }TRCP_PARA;                 // Recipe Parameter
```

（3）TRCP_ERR_INFO – Response Information – S15F14

```
typedef struct{
        int         rmack;          // U1
        int         err_count;
        TERR_INFO   **err_list;
} TRCP_ERR_INFO;
```

（注）TRCP_ERR_INFO の構造は TOBJ_ERR_INFO と同じなので、初期化、設定には
TOBJ_ERR_INFO 用の関数を使用します。

（4）TERR_INFO – Response Error Information – S15F14

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

（5）TRCP_ACT_INFO - Recipe Action(Create / Delete) Information - S15f3

```
typedef struct{
        char        *rmnsspec;          // rcpid
        int         rmnscmd;            // action command 1=create,5=delete
}TRCP_ACT_INFO;                         // Recipe Action
```

（6）TRCP_RENAME_INFO – Recipe Rename Information – S15F5

```
typedef struct{
        char        *rmnsspec;          // rcpid
        char        *rmnewns;           // new rcpid name
}TRCP_RENAME_INFO;                      // Recipe Rename
```

（7） TRCP_S15F8_INFO - S15F8 Ack 情報

```
typedef struct{
        ULONG       rmspace;            // ﾚｼﾋﾟﾈｰﾑｽﾍﾟｰｽにある１個のﾚｼﾋﾟ保存容量(ﾊﾞｲﾄ)
        int         rmack;              // ack
        int         err_count;          // 検出ｴﾗｰ数
        TERR_INFO   **err_list;         // 検出したｴﾗｰ情報ﾘｽﾄ
} TRCP_S15F8_INFO;
```

（8）TRCP_S15F10_INFO – S15F10 Ack 情報

```
typedef struct{
        int         rcpstat;            // status
        char        *rcpver;            // version
        int         rmack;              // ack
        int         err_count;          // 検出ｴﾗｰ数
        TERR_INFO   **err_list;         // 検出したｴﾗｰ情報ﾘｽﾄ
} TRCP_S15F10_INFO;
```

（9）TRCP_ATTR – レシピ属性情報 S15F18

```
typedef struct{
        char        *attrid;            // 属性 ID
        int         format;             // 属性ﾃﾞｰﾀのﾌｫｰﾏｯﾄ
        int         asize;              // 属性ﾃﾞｰﾀの配列ｻｲｽﾞ
        void        *attrdata;          // 属性ﾃﾞｰﾀの格納ﾎﾟｲﾝﾀ
} TRCP_ATTR;
```

(10) TRCP_SECNM - レシピセクション属性情報 S15F18

```
typedef struct{
        char        *rcpsecnm;          // ﾚｼﾋﾟｾｸｼｮﾝﾈｰﾑ
        int         attr_count;         // 属性情報の数
        TRCP_ATTR   **attr_list;        // 属性情報のﾘｽﾄ
        } TRCP_SECNM;
```

(11) TRCP_S15F18_INFO – S15F18 ack 情報

```
typedef struct{
        TRCP_SECNM  *m_secnm;           // ﾚｼﾋﾟｾｸｼｮﾝ属性情報のﾎﾟｲﾝﾀ
        char        *rcpbody;           // ﾚｼﾋﾟﾎﾞﾃﾞｨ
        int         dset_count;         // secnm_list のﾃﾞｰﾀｾｯﾄ数
        TRCP_SECNM  **secnm_list;       // ｴｰｼﾞｪﾝﾄ固有の属性情報ﾘｽﾄ
        int         rmack;              // S15F18 の ack
        int         err_count;          // 検出したｴﾗｰ数
        TERR_INFO   **err_list;         // 検出したｴﾗｰ情報ﾘｽﾄ
} TRCP_S15F18_INFO;
```

3．13．2　RCP レシピ情報アクセス関数

3．13．2．1　GemAllocRcpInfo() - レシピの登録

（1）呼出書式

[c,C++]

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

[.NET VB]

```
Function GemAllocRcpInfo (
        ByVal eqid As Int32,
        ByVal rcpid As String,
        ByRef index As Int32) As Int32
```

[.NET C#]

```
int GemAllocRcpInfo(
        int eqid,
        byte[] rcpid,
        ref int index);
```

（2）引数

eqid

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

rcpid

登録したいレシピ ID が格納されているポインタです。

index

登録された情報領域のインデクス値が格納される領域のポインタです。

（3）戻り値

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

（4）説明

レシピを新規にシステムに登録するための関数です。
登録は、引数 rcpid で与えられるレシピ ID をシステムに登録します。

正常に登録できた場合は、index で指定される領域に登録された情報領域のインデクスが設定返却されます。もし、rcpid に指定されたレシピが既に登録済みであった場合には関数の戻り値 1 を返却します。index には既に登録されている情報領域のインデクスが設定されます。

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

３．13．２．２　GemSetRcpInfo()　- レシピ情報の設定
.　　GemSetRcpInfoX() - インデクス指定でのレシピ情報の設定

（1）呼出書式

[c,C++]

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

API int APIX GemSetRcpInfoX(
        int         eqid,               // 通信対象装置 ID(0,1,2,...)
        int         index,              // RCPID が格納されている領域のインデクス
        TRCP_INFO *pinfo                // レシピ情報格納構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemSetRcpInfo (
        ByVal eqid As Int32,
        ByRef pinfo As dsh_info.TRCP_INFO) As Int32

Function GemSetRcpInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef pinfo As dsh_info.TRCP_INFO) As Int32
```

[.NET C#]

```
int GemSetRcpInfo(
        int eqid,
        ref TRCP_INFO pinfo );

int GemSetRcpInfoX(
        int eqid,
        int index,
        ref TRCP_INFO pinfo );
```

（2）引数

eqid

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

pinfo

設定したいレシピ情報が格納されている格納構造体領域のポインタです。

index

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

（3）戻り値

戻り値	意味
0	正常に設定できた。

(-1)	設定できなかった。

（4）説明

本関数は、pinfo に格納されているレシピ情報の設定・変更に使用します。

引数 pinfo 内の構造体メンバーrcpspec に指定されるレシピ ID の情報として設定されます。

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

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

３．13．２．３　GemGetRcpInfo() - レシピ情報の取得
.　　GemGetRcpInfoX() – インデクス指定でのレシピ情報の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetRcpInfo(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        char        *rcpid;                 // RCPID が格納されている領域のポインタ
        TRCP_INFO *pinfo                    // レシピ情報を格納する構造体ポインタ
);

API int APIX GemGetRcpInfoX(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        int         index;                  // RCPID が格納されている領域のインデクス
        TRCP_INFO *pinfo                    // レシピ情報を格納する構造体ポインタ
);
```

[.NET VB]

```
Function GemGetRcpInfo (
        ByVal eqid As Int32,
        ByVal rcpid As String,
        ByRef pinfo As dsh_info.TRCP_INFO) As Int32

Function GemGetRcpInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef pinfo As dsh_info.TRCP_INFO) As Int32
```

[.NET C#]

```
int GemGetRcpInfo(
        int eqid,
        byte[] rcpid,
        ref TRCP_INFO pinfo);

int GemGetRcpInfoX(
        int eqid,
        int index,
        ref TRCP_INFO pinfo );
```

（2）引数

eqid

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

rcpid

レシピ ID が格納されている領域のポインタです。

pinfo

取得したレシピ情報を格納する構造体領域のポインタを格納するポインタです。

index

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

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

（3）戻り値

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

（4）説明

rcpid または index に指定されている RCPID のレシピ情報を pinfo が指す構造体に格納します。

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

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

```
TRCP_INFO   *pinfo;

if ( GemGetRcpInfo( eqid, rcpid, pinfo ) == 0 ){
    pinfo の処理
    処理終了後
    DshFreeTRCP_INFO( pinfo );          // pinfo 内に使用されているﾒﾓﾘの開放
}
```

３．13．２．４　GemDelRcpInfo()　- レシピの削除
.　　GemDelRcpInfoX() - インデクスでのレシピの削除

（1）呼出書式

[c,C++]

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

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

[.NET VB]

```
Function GemDelRcpInfo (
        ByVal eqid As Int32,
        ByVal rcpid As String) As Int32

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

[.NET C#]

```
int GemDelRcpInfo(
        int eqid,
        byte[] rcpid);

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

（2）引数

eqid

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

rcpid

レシピ ID が格納されている領域のポインタです。

index

削除したいレシピ情報のインデクスです。

（3）戻り値

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

（4）説明

rcpid または index に指定されているレシピ ID の情報をシステムの登録から削除します。

３．13．２．５　GemSetRcpState() － レシピ状態の設定
.　　GemSetRcpStateX() － インデクスでのレシピ状態の設定

（1）呼出書式

[c,C++]

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

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

[.NET VB]

```
Function GemSetRcpState (
        ByVal eqid As Int32,
        ByVal rcpid As String,
        ByVal state As Int32) As Int32

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

[.NET C#]

```
int GemSetRcpState(
        int eqid,
        byte[] rcpid,
        int state);

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

（2）引数

eqid

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

rcpid

レシピ ID が格納されている領域のポインタです。

state

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

index

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

（3）戻り値

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

（4）説明

レシピ情報の状態値を設定します。
状態値の意味合いは値=(-1)以外についてはユーザが定義して使用します。

３．13．２．６　GemGetRcpState() － レシピ状態の取得
.　　GemGetRcpStateX() － インデクスでのレシピ状態の取得

（1）呼出書式

[c,C++]

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

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

[.NET VB]

```
Function GemGetRcpState (
        ByVal eqid As Int32,
        ByVal rcpid As String,
        ByRef state As Int32) As Int32

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

[.NET C#]

```
int GemGetRcpState(
        int eqid,
        byte[] rcpid,
        ref int state);

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

（2）引数

eqid

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

rcpid

レシピ ID が格納されている領域のポインタです。

state

取得した RCPID の状態値を格納する領域のポインタです。

index

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

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

（3）戻り値

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

（4）説明

レシピ情報の状態値を取得します。

状態値の意味合いは値=(-1)以外についてはユーザが定義して使用します。

3．13．2．7　GemGetRcpList() – 全登録レシピ ID 取得関数

（1）呼出書式

[c,C++]

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

[.NET VB]

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

[.NET C#]

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

（2）引数

eqid

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

list

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

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

（4）説明

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

TTEXT_DLIST 構造体は次のとおりです。

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

3．13．２．８　GemGetRcpId() - インデクスから RCPID（レシピ ID）の取得

（1）呼出書式

[c,C++]

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

[.NET VB]

```
Function GemGetRcpId (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal rcpid As String) As Int32
```

[.NET C#]

```
int GemGetRcpId(
        int eqid,
        int index,
        byte[] rcpid);
```

（2）引数

eqid

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

index

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

rcpid

RCPID を格納するための領域のポインタです。

（3）戻り値

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

（4）説明

RCP 情報のインデクスから RCPID（レシピ ID）を取得し、rcpid に格納します。
正常に取得できた場合は関数戻り値として 0 が返却されます。

３．13．２．９　GemGetRcpIdIndex() -RCPID（レシピ ID）からインデクスの取得

（1）呼出書式

[c,C++]

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

[.NET VB]

```
Function GemGetRcpIdIndex (
        ByVal eqid As Int32,
        ByVal rcpid As String,
        ByRef index As Int32) As Int32
```

[.NET C#]

```
int GemGetRcpIdIndex(
        int eqid,
        byte[] rcpid,
        ref int index);
```

（2）引数

eqid

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

rcpid

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

index

取得したインデクスの値を格納するための領域のポインタです。

（3）戻り値

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

（4）説明

rcpid に指定される RCPID（レシピ ID）からレシピ情報インデクスを取得するための関数です。
取得されたインデクスは index で指定された領域に格納されます。
正常に取得できた場合は関数戻り値として 0 が返却されます。

3．13．2．10 GemSendS15F3() - レシピ・スペースアクション要求送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS15F3(
        int     eqid,                       // 通信対象装置 ID(0,1,2,...)
        char    *rcpid,                     // ﾚｼﾋﾟ ID 格納領域のﾎﾟｲﾝﾀ
        int     rmnscmd,                    // ｱｸｼｮﾝｺﾏﾝﾄﾞ
        TRCP_ERR_INFO *erinfo,              // 応答情報格納用構造体のﾎﾟｲﾝﾀ
        int (WINAPI *s15f3Callback)(),      // 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 GemSendS15F3 (
        ByVal eqid As Int32,
        ByVal rmnsspec As String,
        ByVal rmnscmd As Int32,
        ByRef erinfo As dsh_info.TRCP_ERR_INFO,
        ByVal callback As vcallback.callback_S15F3,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS15F3(
        int eqid,
        byte[] rmnsspec,
        int rmnscmd,
        ref TRCP_ERR_INFO erinfo,
        CallbackS15F3 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。

rcpid

アクション要求をしたいレシピ ID が格納されているポインタです。

rmnscmd

レシピネームスペースで実行されるコマンドです。(1=生成、5=削除)

erinfo

受信した応答メッセージ S15F4 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。

s15f3Callback

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 S15F4 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

rcpid で指定されたレシピネームスペースについて rmnscmd で指定したアクションの実行要求のために S15F3 メッセージを送信します。

S15F4 応答メッセージ受信で得られた情報はデコードされて erinfo で指定された構造体領域に返却されます。

送信要求から S15F4 応答メッセージ受信までの制御は引数の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。

s15f3Callback の指定	制御の流れ
なし (=0)	S15F3 送信後、応答ﾒｯｾｰｼﾞ S15F4 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 に応答情報が格納されているﾎﾟｲﾝﾀが返却されます。
あり	送信要求後、S15F3 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 erinfo に応答情報が格納されているﾎﾟｲﾝﾀが返却されます。 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、その中に含まれている応答情報を erinfo で指定された TRCP_ERR_INFO 構造体の中に格納され返却されます。ユーザ側で 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 DshFreeTRCP_ERR_INFO()関数を使って行ってください。

```
DshFreeTRCP_ERR_INFO (erinfo)
```

（5）コールバック関数

[c,C++]

```
API int  APIX  s15f3Callback(
        int    eqid,                  // 装置 ID
        int    end_status,            // 実行結果
        TRCP_ERR_INFO *erinfo,        // 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_S15F3(ByVal eqid As Integer, ByVal end_status As Integer, ByRef erinfo As
dsh_info.TRCP_ERR_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS15F3(int eqid, int end_status, ref TRCP_ERR_INFO errinfo, 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 に応答情報が格納されているﾎﾟｲﾝﾀが返却されます。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．13．2．11 GemSendS15F5() - レシピ・スペース・リネーム要求送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS15F5(
        int     eqid,                       // 通信対象装置ID(0,1,2,...)
        char    *cur_rcpid,                 // 現ﾚｼﾋﾟID格納領域のﾎﾟｲﾝﾀ
        char    *new_rcpid,                 // 新ﾚｼﾋﾟID格納領域のﾎﾟｲﾝ
        TRCP_ERR_INFO *erinfo,              // 応答情報格納用構造体のﾎﾟｲﾝﾀ
        int (WINAPI *s15f5Callback)(),      // 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 GemSendS15F5 (
        ByVal eqid As Int32,
        ByVal rmnsspec As String,
        ByVal rmnewns As String,
        ByRef erinfo As dsh_info.TRCP_ERR_INFO,
        ByVal callback As vcallback.callback_S15F5,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS15F5(
        int eqid,
        byte[] rmnsspec,
        byte[] rmnewns,
        ref TRCP_ERR_INFO erinfo,
        CallbackS15F5 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。

cur_rcpid

現在のレシピID(=ﾚｼﾋﾟﾈｰﾑ)が格納されているポインタです。

rmnscmd

新しく変えたいレシピID(ﾚｼﾋﾟﾈｰﾑ)でが格納されているポインタです。

erinfo

受信した応答メッセージS15F6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。

s15f5Callback

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 S15F6 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

cur_rcpid で指定されたレシピネームを new_rcpid に指定されたレシピネームにリネームするために S15F5 メッセージを送信します。

S15F6 応答メッセージ受信で得られた情報はデコードされて erinfo で指定された構造体領域に返却されます。

送信要求から S15F6 応答メッセージ受信までの制御は引数の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。

s15f5Callback の指定	制御の流れ
なし (=0)	S15F5 送信後、応答ﾒｯｾｰｼﾞ S15F6 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 に応答情報が格納されているﾎﾟｲﾝﾀが返却されます。
あり	送信要求後、S15F5 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 erinfo に応答情報が格納されているﾎﾟｲﾝﾀが返却されます。 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、その中に含まれている応答情報を erinfo で指定された TRCP_ERR_INFO 構造体の中に格納され返却されます。ユーザ側で 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 DshFreeTRCP_ERR_INFO()関数を使って行ってください。

```
DshFreeTRCP_ERR_INFO (erinfo)
```

（5）コールバック関数

[c,C++]

```
API int  APIX  s15f5Callback(
        int    eqid,                  // 装置 ID
        int    end_status,            // 実行結果
        TRCP_ERR_INFO *erinfo,        // 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_S15F5(ByVal eqid As Integer, ByVal end_status As Integer, ByRef erinfo As dsh_info.TRCP_ERR_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS15F5(int eqid, int status, ref TRCP_ERR_INFO errinfo, 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 に応答情報が格納されているﾎﾟｲﾝﾀが返却されます。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．13．２．12　GemSendS15F7() – レシピ・スペース取得メッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS15F7(
        int     eqid,                       // 通信対象装置 ID(0,1,2,...)
        char    *rcpid,                     // レシピ ID 格納領域のﾎﾟｲﾝﾀ
        TRCP_S15F8_INFO *erinfo,            // 応答情報格納用構造体のﾎﾟｲﾝﾀ
        int (WINAPI *s15f7Callback)(),      // 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 GemSendS15F7 (
        ByVal eqid As Int32,
        ByVal objspec As String,
        ByRef erinfo As dsh_info.TRCP_S15F8_INFO,
        ByVal callback As vcallback.callback_S15F7,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS15F7(
        int eqid,
        byte[] objspec,
        ref TRCP_S15F8_INFO erinfo,
        CallbackS15F7 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。

rcpid

問合せたいレシピ ID が格納されているポインタです。

erinfo

受信した応答メッセージ S15F8 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。

s15f7Callback

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 S15F8 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

rcpid で指定されたレシピ情報を格納するために必要な記憶容量の問い合わせをホストに対して行います。S15F7 メッセージで送信します。
S15F8 応答メッセージ受信で得られた情報はデコードされて erinfo で指定された構造体領域に返却されます。

送信要求から S15F8 応答メッセージ受信までの制御は引数の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。

s15f7Callback の指定	制御の流れ
なし (=0)	S15F7 送信後、応答ﾒｯｾｰｼﾞ S15F8 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 に応答情報が格納されているﾎﾟｲﾝﾀが返却されます。
あり	送信要求後、S15F7 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 erinfo に応答情報が格納されているﾎﾟｲﾝﾀが返却されます。 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、その中に含まれている応答情報を erinfo で指定された TRCP_S15F8_INFO 構造体の中に格納され返却されます。ユーザ側で 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 DshFreeTRCP_S15F8_INFO()関数を使って行ってください。

```
DshFreeTRCP_S15F8_INFO (erinfo)
```

（5）コールバック関数

[c,C++]

```
API int  APIX   s15f7Callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TRCP_S15F8_INFO *erinfo,        // 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_S15F7(ByVal eqid As Integer, ByVal end_status As Integer, ByRef erinfo As
dsh_info.TRCP_S15F8_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS15F7(int eqid, int end_status, ref TRCP_S15F8_INFO errinfo, 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 に応答情報が格納されているﾎﾟｲﾝﾀが返却されます。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．13．２．13　GemSendS15F9() – レシピ・ステータス取得メッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS15F9(
        int   eqid,                         // 通信対象装置 ID(0,1,2,...)
        char *rcpid,                        // レシピ ID 格納領域のﾎﾟｲﾝﾀ
        TRCP_S15F10_INFO *erinfo,           // 応答情報格納用構造体のﾎﾟｲﾝﾀ
        int (WINAPI *s15f9Callback)(),      // 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 GemSendS15F9 (
          ByVal eqid As Int32,
          ByVal rcpspec As String,
          ByRef erinfo As dsh_info.TRCP_S15F10_INFO,
          ByVal callback As vcallback.callback_S15F9,
          ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS15F9(
          int eqid,
          byte[] rcpspec,
          ref TRCP_S15F10_INFO erinfo,
          CallbackS15F9 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。

rcpid

ステータスを受信したいレシピ ID が格納されているポインタです。

erinfo

受信した応答メッセージ S15F10 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。

s15f9Callback

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 S15F10 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

rcpidで指定されたレシピの状態情報をホストから取得するための関数です。S15F9メッセージで送信します。
S15F10応答メッセージ受信で得られた情報はデコードされてerinfoで指定された構造体領域に返却されます。

送信要求からS15F10応答メッセージ受信までの制御は引数の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。

s15f9Callbackの指定	制御の流れ
なし	S15F9送信後、応答ﾒｯｾｰｼﾞS15F10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S15F10応答情報が返却されます。
あり	送信要求後、S15F9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erinfoにS15F10応答情報が返却されます。 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、その中に含まれている応答情報をerinfoで指定されたTRCP_S15F10_INFO構造体の中に格納され返却されます。ユーザ側で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DshFreeTRCP_S15F10_INFO()関数を使って行ってください。

```
DshFreeTRCP_S15F10_INFO (erinfo)
```

（5）コールバック関数

[c,C++]

```
API int  APIX   s15f9Callback(
        int    eqid                    // 装置ID
        int    end_status,             // 実行結果
        TRCP_S15F10_INFO *erinfo,      // 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_S15F9(ByVal eqid As Integer, ByVal end_status As Integer, ByRef erinfo As dsh_info.TRCP_S15F10_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS15F9(int eqid, int end_status, ref TRCP_S15F10_INFO errinfo, 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 に S15F10 応答情報が返却されます。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．13．2．14 GemSendS15F13() - レシピ情報メッセージ送信関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS15F13(
        int     eqid,                       // 通信対象装置 ID(0,1,2,...)
        TRCP_INFO *info,                    // レシピ情報格納領域のﾎﾟｲﾝﾀ
        TRCP_ERR_INFO *erinfo,              // S15F14 応答情報格納用構造体のﾎﾟｲﾝﾀ
        int (WINAPI *s15f13Callback)(),     // 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 GemSendS15F13 (
        ByVal eqid As Int32,
        ByRef info As dsh_info.TRCP_INFO,
        ByVal updt_flag As Int32,
        ByRef erinfo As dsh_info.TRCP_ERR_INFO,
        ByVal callback As vcallback.callback_S15F13,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS15F13(
        int eqid,
        ref TRCP_INFO info,
        int updt_flag,
        ref TRCP_ERR_INFO erinfo,
        CallbackS15F13 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

送信したいレシピ情報が格納されている構造体のポインタです。

erinfo

受信した応答メッセージ S15F14 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。

s15f13Callback

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 S15F14 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

レシピ情報の送信要求を行います。S15F13 メッセージで送信します。

要求を受けた DSHGEM-LIB は、info に格納されているレシピ情報を S15F13 メッセージにエンコードし相手に送信します。

S15F14 応答メッセージ受信で得られた情報はデコードされて erinfo で指定された構造体領域に返却されます。

送信要求から S15F14 応答メッセージ受信までの制御は引数の S15F13 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。

s15f13Callback の指定	制御の流れ
なし (=0)	S15F13 送信後、応答ﾒｯｾｰｼﾞ S15F14 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 S15F14 応答情報が返却されます。
あり	送信要求後、S15F13 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 erinfo に S15F14 応答情報が返却されます。 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、その中に含まれている応答情報がデコードされ erinfo で指定された TRCP_ERR_INFO 構造体の中に格納され返却されます。ユーザ側で 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 DshFreeTRCP_ERR_INFO()関数を使って行ってください。

```
DshFreeTRCP_ERR_INFO (erinfo)
```

（5）コールバック関数

[c,C++]

```
API int  APIX   s15f13Callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TRCP_ERR_INFO *erinfo,          // S15F14 応答情報格納用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_S15F13(ByVal eqid As Integer, ByVal end_status As Integer, ByRef erinfo As dsh_info.TRCP_ERR_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS15F13(int eqid, int status, ref TRCP_ERR_INFO errinfo, 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 に S15F14 応答情報が返却されます。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．13．2．15 GemSendS15F17() - レシピ情報問合せ関数

（1）呼出書式

[c,C++]

```
API int APIX GemSendS15F17(
        int     eqid,                       // 通信対象装置 ID(0,1,2,...)
        char    *rcpid,                     // ﾚｼﾋﾟ情報格納領域のﾎﾟｲﾝﾀ
        int     rcpseccode,                 // ﾚｼﾋﾟｾｸｼｮﾝｺｰﾄﾞ
        TRCP_S15F18_INFO *erinfo,           // S15F18 応答情報格納用構造体のﾎﾟｲﾝﾀ
        int (WINAPI *s15f17Callback)(),     // 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 GemSendS15F17 (
        ByVal eqid As Int32,
        ByVal rcpspec As String,
        ByVal rcpseccode As Int32,
        ByRef erinfo As dsh_info.TRCP_S15F18_INFO,
        ByVal callback As vcallback.callback_S15F17,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS15F17(
        int eqid,
        byte[] rcpspec,
        int rcpseccode,
        ref TRCP_S15F18_INFO erinfo,
        CallbackS15F17 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。

rcpid

ホストに問合せたいレシピ ID が格納されているポインタです。

rcpseccode

レシピのセクションを指定します。

erinfo

受信した応答メッセージ S15F18 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。

s15f17Callback

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 S15F18 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

ホストにレシピ情報の問合せ要求を行います。S15F17 メッセージを使います。
問合せ指定はレシピ ID で行います。
要求を受けた DSHGEM-LIB は、S15F17 メッセージにエンコードし、ホストに送信します。
送信要求から S15F18 応答メッセージ受信までの制御は引数の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。

s15f17Callback の指定	制御の流れ
なし (=0)	S15F17 送信後、応答ﾒｯｾｰｼﾞ S15F18 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 S15F18 応答情報が返却されます。
あり	送信要求後、S15F17 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 erinfo に S15F18 応答情報が返却されます。 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、その中に含まれている応答情報がデコードされ erinfo で指定された TRCP_S15F18_INFO 構造体の中に格納され返却されます。ユーザ側で 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 DshFreeTRCP_S15F18_INFO()関数を使って行ってください。

```
DshFreeTRCP_S15F18_INFO (erinfo)
```

（5）コールバック関数

[c,C++]

```
API int  APIX  s15f17Callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TRCP_S15F18_INFO *erinfo,       // S15F18 応答情報格納用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_S15F17(ByVal eqid As Integer, ByVal end_status As Integer, ByRef erinfo As
dsh_info.TRCP_S15F18_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS15F17(int eqid, int end_status, ref TRCP_S15F18_INFO errinfo, 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 に S15F18 応答情報が返却されます。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

3．13．3　RCP レシピ関連ライブラリ関数

3．13．3．1　DshDecodeS15F3() - S15F3 をレシピ・アクション情報にデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS15F3(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TRCP_ACT_INFO *pinfo            // ﾃﾞｺｰﾄﾞした情報を格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS15F3 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef info As dsh_info.TRCP_ACT_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS15F3(
        ref DSHMSG smsg,
        ref TRCP_ACT_INFO info );
```

（2）引数

smsg

S15F3 の SECS ﾒｯｾｰｼﾞ情報が格納されている構造体のポインタです。

pinfo

デコードしたレシピアクション情報を格納する構造体のポインタです。

（3）戻り値

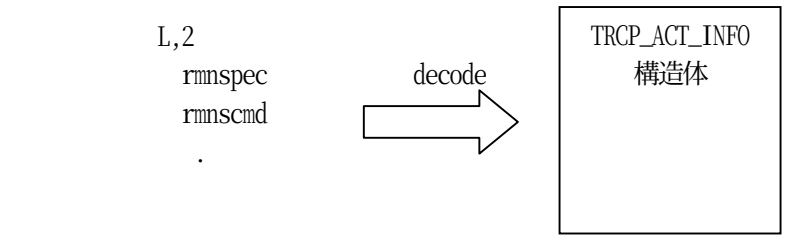
戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

S15F3 メッセージに含まれるレシピアクション情報を、ユーザプログラムが処理しやすい TRCP_ACT_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 DshFreeTACT_INFO()関数を使って開放してください。

smsg S15F3



3．13．3．2　DshEncodeS15F3() - レシピ・アクション情報を S15F3 へエンコード

（1）呼出書式

[c,C++]

```
API int  APIX DshEncodeS15F3(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        BYTE   *buffer,                 // S15F3 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
        TRCP_ACT_INFO *pinfo            // ｴﾝｺｰﾄﾞしたいｱｸｼｮﾝ情報がされている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshEncodeS15F3 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef buff As Byte,
        ByVal bufflen As Int32,
        ByRef pinfo As dsh_info.TRCP_ACT_INFO) As Int32
```

[.NET C#]

```
int DshEncodeS15F3(r
        ef DSHMSG smsg,
        byte[] buff,
        int bufflen,
        ref TRCP_ACT_INFO info);
```

（2）引数

smsg

エンコードした S15F3 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 S15F3 のテキストを格納するバッファポインタです。

bufflen

buffer のバイトサイズです。

pinfo

エンコードしたいレシピアクション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

TRCP_ACT_INFO 構造体に格納されているレシピアクション情報を、S15F3 の SECS メッセージにエンコードします。

3．13．3．3　DshDecodeS15FRsp() - S15F4，S15F6，S15F14，S15F16，S15F18 のデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS15FRsp(
        DSHMSG *smsg,                   // SECS 応答ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TRCP_ERR_INFO *erinfo           // 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 DshDecodeS15FRsp (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef erinfo As dsh_info.TRCP_ERR_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS15FRsp(
        ref DSHMSG smsg,
        ref TRCP_ERR_INFO erinfo );
```

（2）引数

smsg

S15F4，F6，F14,F16,または F18 の SECS 応答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

デコードしたレシピ情報を格納する TRCP_ERR_INFO 構造体のポインタです。

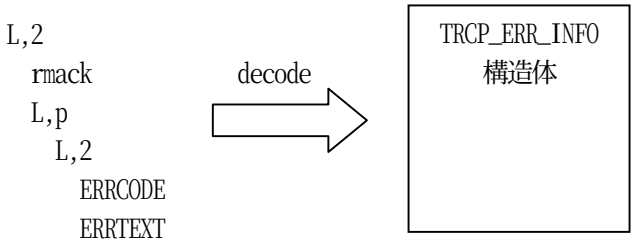
（3）戻り値

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

S15F3，S15F5，S15F13，S15F15，S15F17 の応答メッセージに含まれるレシピ関連応答情報を、ユーザプログラムが処理しやすい TRCP_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 DshFreeTRCP_ERR_INFO()関数を使って開放してください。

smsg S15F4，F6，F14，F16，F18



３．13．３．４　DshFreeTRCP_ACT_INFO() - レシピ・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 DshFreeTRCP_ACT_INFO(
      TRCP_ACT_INFO *pinfo            // ﾒﾓﾘを開放したい Action 情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTRCP_ACT_INFO (
        ByRef info As dsh_info.TRCP_ACT_INFO)
```

[.NET C#]

```
void DshFreeTRCP_ACT_INFO(
        ref TRCP_ACT_INFO info );
```

（2）引数

pinfo

メモリを解放したいレシピ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

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

３．13．３．５　DshCopyTRCP_ACT_INFO() - レシピ・アクション情報構造体メモリのコピー

（1）呼出書式

[c,C++]

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

[.NET VB]

```
Function DshCopyTRCP_ACT_INFO (
          ByRef dinfo As dsh_info.TRCP_ACT_INFO,
          ByRef sinfo As dsh_info.TRCP_ACT_INFO) As Int32
```

[.NET C#]

```
int DshCopyTRCP_ACT_INFO(
          ref TRCP_ACT_INFO dinfo,
          ref TRCP_ACT_INFO sinfo);
```

（2）引数

pinfo

レシピアクション情報のコピー先構造体メモリのポインタです。

sinfo

コピー元のレシピアクション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 TRCP_ACT_INFO 構造体内に格納されているレシピアクション情報を dinfo で指定された TRCP_ACT_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、DshFreeTRCP_ACT_INFO()関数を使って開放してください。

3．13．3．6　DshMakeS15F3Response() - S15F3 の応答メッセージの生成

（1）呼出書式

[c,C++]

```
API int APIX DshMakeS15F3Response(
        TRCP_ACTION_INFO *info,          // ﾚｼﾋﾟｱｸｼｮﾝ情報格納領域ﾎﾟｲﾝﾀ
        TRCP_ERR_INFO *erinfo,           // S15F4 に設定する応答情報格納領域のﾎﾟｲﾝﾀ
        DSHMSG *smsg,                    // S15F4 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,                    // S15F4 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 DshMakeS15F3Response (
          ByRef pinfo As dsh_info.TRCP_ACT_INFO,
          ByRef erinfo As dsh_info.TRCP_ERR_INFO,
          ByRef smsg As dshdr2.DSHMSG,
          ByRef buff As Byte,
          ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS15F3Response(
          ref TRCP_ACT_INFO pinfo,
          ref TRCP_ERR_INFO erinfo,
          ref DSHMSG smsg,
          byte[] buff,
          int buff_size );
```

（2）引数

info

レシピ・アクション情報が格納されている領域のポインタです。

erinfo

S15F4 メッセージに設定する応答情報が格納されている領域のポインタです。

msg

S15F4 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

S15F4 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

S15F3 に対する S15F4 応答メッセージを sinfo に含まれるレシピアクション情報と応答情報に従って作成します。

応答情報内の、rmack を S15F4 の RMACK として設定します。
rmack はユーザがレシピ・アクション情報を評価した結果です。

erinfo 応答情報の初期設定と情報設定にはそれぞれ DshInitTOBJ_ERR_INFO()、DshPutTOBJ_ERR_INFO()関数を使用することができます。(3.21 を参照してください)

3．13. 3．7　DshDecodeS15F5() - S15F5 をレシピ・リネーム情報にデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS15F5(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TRCP_RENAME_INFO *pinfo         // ﾃﾞｺｰﾄﾞした情報を格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS15F5 (
          ByRef smsg As dshdr2.DSHMSG,
          ByRef info As dsh_info.TRCP_RENAME_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS15F5(
          ref DSHMSG smsg,
          ref TRCP_RENAME_INFO info );
```

（2）引数

smsg

S15F5 の SECS ﾒｯｾｰｼﾞ情報が格納されている構造体のポインタです。

pinfo

デコードしたレシピ・リネーム情報を格納する構造体のポインタです。

（3）戻り値

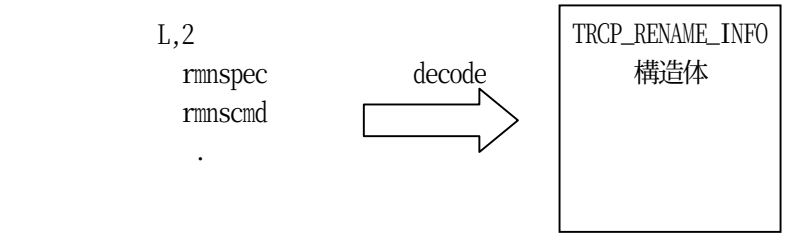
戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

S15F5 メッセージに含まれるレシピ・リネーム情報を、ユーザプログラムが処理しやすい TRCP_RENAME_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 DshFreeTTRCP_RENAME_INFO()関数を使って開放してください。

smsg S15F5



３．13．３．８　DshFreeTRCP_RENAME_INFO() - レシピ・リネーム情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_RENAME_INFO(
      TRCP_RENAME_INFO *pinfo            // ﾒﾓﾘを開放したいﾘﾈｰﾑ情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTRCP_RENAME_INFO (
        ByRef info As dsh_info.TRCP_RENAME_INFO)
```

[.NET C#]

```
void DshFreeTRCP_RENAME_INFO(
        ref TRCP_RENAME_INFO info );
```

（2）引数

pinfo

メモリを解放したいレシピ・リネーム情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

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

３．13．３．９　DshCopyTRCP_RENAME_INFO() - レシピ・リネーム情報構造体メモリのコピー

（1）呼出書式

[c,C++]

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

[.NET VB]

```
Function DshCopyTRCP_RENAME_INFO Lib  (
          ByRef dinfo As dsh_info.TRCP_RENAME_INFO,
          ByRef sinfo As dsh_info.TRCP_RENAME_INFO) As Int32
```

[.NET C#]

```
int DshCopyTRCP_RENAME_INFO(
          ref TRCP_RENAME_INFO dinfo,
          ref TRCP_RENAME_INFO sinfo);
```

（2）引数

pinfo

レシピ・リネーム情報のコピー先構造体メモリのポインタです。

sinfo

コピー元のレシピ・リネーム情報が格納されている構造体メモリのポインタです。

（3）戻り値

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

（4）説明

sinfo が指す TRCP_RENAME_INFO 構造体内に格納されているレシピ・リネーム情報を dinfo で指定された TRCP_RENAME_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、DshFreeTRCP_RENAME_INFO()関数を使って開放してください。

3．13. 3．10 DshMakeS15F5Response() - S15F5 の応答メッセージの生成

（1）呼出書式

[c,C++]

```
API int APIX DshMakeS15F5Response(
        TRCP_RENAME_INFO *info,         // ﾚｼﾋﾟﾘﾈｰﾑ情報格納領域ﾎﾟｲﾝﾀ
        TRCP_ERR_INFO *erinfo,          // S15F6 に設定する応答情報か格納領域のﾎﾟｲﾝﾀ
        DSHMSG *smsg,                   // S15F6 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,                  // S15F6 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 DshMakeS15F5Response (
          ByRef pinfo As dsh_info.TRCP_RENAME_INFO,
          ByRef erinfo As dsh_info.TRCP_ERR_INFO,
          ByRef smsg As dshdr2.DSHMSG,
          ByRef buff As Byte,
          ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS15F5Response(
          ref TRCP_RENAME_INFO pinfo,
          ref TRCP_ERR_INFO erinfo,
          ref DSHMSG smsg,
          byte[] buff,
          int buff_size );
```

（2）引数

info

レシピ・レネーム情報が格納されている領域のポインタです。

erinfo

S15F6 メッセージに設定する応答情報が格納されている領域のポインタです。

msg

S15F6 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

S15F6 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

S15F5 に対する S15F6 応答メッセージを info に含まれるレシピ・リネーム情報と erinfo に含まれる応答情報に従って作成します。

応答情報内の rmack を S15F6 の RMACK として設定します。
rmack はユーザがレシピ・リネーム情報を評価した結果です。

erinfo 応答情報の初期設定と情報設定にはそれぞれ DshInitTOBJ_ERR_INFO()、DshPutTOBJ_ERR_INFO()関数を使用することができます。

3．13. 3．11 DshDecodeS15F8() - S15F8 応答メッセージのデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS15F8(
        DSHMSG *smsg,                   // SECS 応答ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TRCP_S15F8_INFO *erinfo         // 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 DshDecodeS15F8 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef erinfo As dsh_info.TRCP_S15F8_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS15F8(
        ref DSHMSG smsg,
        ref TRCP_S15F8_INFO erinfo );
```

（2）引数

smsg

S15F8 の SECS 応答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

デコードしたレシピスペースデータ情報を格納する TRCP_S15F8_INFO 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。

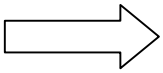
（4）説明

S15F7 の応答メッセージに含まれるレシピスペースデータの情報を、ユーザプログラムが処理しやすい TRCP_S15F8_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 DshFreeTRCP_S15F8_INFO()関数を使って開放してください。

```
smsg S15F8
  L,2
    rmspace
    L,2
      rmack
      L,p
        L,2
          ERRCODE
          ERRTEXT
        .
```

Decode



TRCP_S15F8_INFO
構造体

３．13. ３．12 DshFreeTRCP_S15F8_INFO() － S15F8 応答情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_S15F8_INFO(
      TRCP_S15F8_INFO *erinfo        // ﾒﾓﾘを開放したい応答情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTRCP_S15F8_INFO (
        ByRef info As dsh_info.TRCP_S15F8_INFO)
```

[.NET C#]

```
void DshFreeTRCP_S15F8_INFO(
        ref TRCP_S15F8_INFO info );
```

（2）引数

erinfo

メモリを解放したいレシピ関連応答情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

S15F7 の応答メッセージデコード時に TRCP_S15F8_INFO 構造体内で情報格納用に割当て使用されているメモリを全て解放します。

開放した後、TRCP_S15F8_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。

3．13. 3．13 DshDecodeS15F10() - S15F10 応答メッセージのデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS15F10(
        DSHMSG *smsg,                   // SECS 応答ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TRCP_S15F10_INFO *erinfo        // 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 DshDecodeS15F10 (
          ByRef smsg As dshdr2.DSHMSG,
          ByRef info As dsh_info.TRCP_S15F10_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS15F10(
          ref DSHMSG smsg,
          ref TRCP_S15F10_INFO info );
```

（2）引数

smsg

S15F10 の SECS 応答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

デコードしたレシピ情報を格納する TRCP_S15F10_INFO 構造体のポインタです。

（3）戻り値

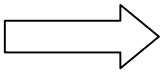
戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

S15F9 の応答メッセージに含まれるレシピ関連メッセージの情報を、ユーザプログラムが処理しやすい TRCP_S15F10_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 DshFreeTRCP_S15F10_INFO()関数を使って開放してください。

```
smsg S15F10
  L,3
    rcpstat
    rcpvers
    L,2
      rmack
      L,p
        L,2
          ERRCODE
          ERRTEXT
        .
```



TRCP_S15F10_INFO
構造体

３．13. ３．14 DshFreeTRCP_S15F10_INFO() – S15F10 応答情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_S15F10_INFO(
      TRCP_S15F10_INFO *erinfo      // ﾒﾓﾘを開放したい応答情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTRCP_S15F10_INFO (
        ByRef info As dsh_info.TRCP_S15F10_INFO)
```

[.NET C#]

```
void DshFreeTRCP_S15F10_INFO(
        ref TRCP_S15F10_INFO info );
```

（2）引数

erinfo

メモリを解放したいレシピ関連応答情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

S15F9 の応答メッセージデコード時に TRCP_S15F10_INFO 構造体内で情報格納用に割当て使用されているメモリを全て解放します。

開放した後、TRCP_S15F10_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。

3．13. 3．15 DshDecodeS15F13() - S15F13 をレシピ情報にデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS15F13(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TRCP_INFO *pinfo                // ﾃﾞｺｰﾄﾞした情報を格納する構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeS15F13 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef info As dsh_info.TRCP_INFO,
        ByRef updt_flag As Int32) As Int32
```

[.NET C#]

```
int DshDecodeS15F13Ext(
        ref DSHMSG smsg,
        ref TRCP_INFO tinfo,
        ref int updt_flag );
```

（2）引数

smsg

S15F13 の SECS ﾒｯｾｰｼﾞ情報が格納されている構造体のポインタです。

pinfo

デコードしたレシピ情報を格納する構造体のポインタです。

（3）戻り値

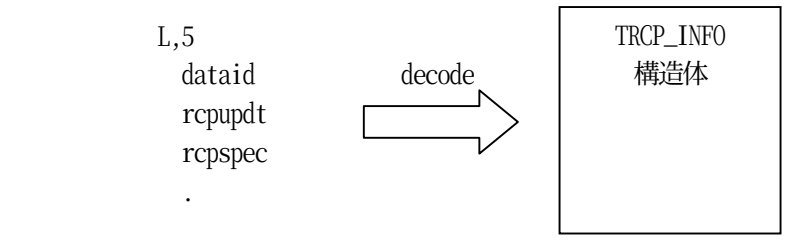
戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

S15F13 メッセージに含まれるレシピ情報を、ユーザプログラムが処理しやすい TRCP_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 DshFreeTRCP_INFO()関数を使って開放してください。

smsg S15F13



3．13. 3．16 DshEncodeS15F13() - レシピ情報を S15F13 へエンコード

（1）呼出書式

[c,C++]

```
API int  APIX DshEncodeS15F13(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        BYTE   *buffer,                 // S15F13 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
        TRCP_INFO *pinfo                // ｴﾝｺｰﾄﾞしたいﾚｼﾋﾟ情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshEncodeS15F13 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef buff As Byte,
        ByVal bufflen As Int32,
        ByRef rinfo As dsh_info.TRCP_INFO,
        ByVal updt_flag As Int32) As Int32
```

[.NET C#]

```
int DshEncodeS15F13(
        ref DSHMSG smsg,
        byte[] buff,
        int bufflen,
        ref TRCP_INFO rinfo,
        int updt_flag );
```

（2）引数

smsg

エンコードした S15F13 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 S15F13 のテキストを格納するバッファポインタです。

bufflen

buffer のバイトサイズです。

pinfo

エンコードしたいレシピ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。

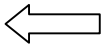
（4）説明

TRCP_INFO 構造体に格納されているレシピ情報を、S15F13 の SECS メッセージにエンコードします。

smsg S15F13

L,4

TRCP_INFO
構造体 T

rcpid encode
mdln 
.

３．13. ３．17 DshFreeTRCP_INFO() - レシピ情報構造体メモリの開放

（1）呼出書式

[c,C++]

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

[.NET VB]

```
Sub DshFreeTRCP_INFO (
        ByRef info As dsh_info.TRCP_INFO)
```

[.NET C#]

```
void DshFreeTRCP_INFO(
        ref TRCP_INFO info );
```

（2）引数

pinfo

メモリを解放したいレシピ情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

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

3．13. 3．18 DshCopyTRCP_INFO() - レシピ情報構造体メモリのコピー

（1）呼出書式

[c,C++]

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

[.NET VB]

```
Function DshCopyTRCP_INFO (
          ByRef dinfo As dsh_info.TRCP_INFO,
          ByRef sinfo As dsh_info.TRCP_INFO) As Int32
```

[.NET C#]

```
int DshCopyTRCP_INFO(
          ref TRCP_INFO dinfo,
          ref TRCP_INFO sinfo );
```

（2）引数

dinfo

レシピ情報のコピー先構造体メモリのポインタです。

sinfo

コピー元のレシピ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TRCP_INFO 構造体内に格納されているレシピ情報をdinfo が指定するTRCP_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、DshFreeTRCP_INFO()関数を使って開放してください。

3．13. 3．19 DshInitRcpInfo - レシピ TRCP_INFO の初期設定

（1）呼出書式

[c,C++]

```
API int  APIX DshInitRcpInfo(
        TRCP_INFO    *info,              // TRCP_INFO 構造体のﾎﾟｲﾝﾀ
        char         *rcpspec,           // ﾚｼﾋﾟ ID
        char         *rcpbody,           // RCPBODY 名
        int          para_count          // ｺﾏﾝﾄﾞｺｰﾄﾞ情報ﾘｽﾄのｻｲｽﾞ
);
```

[.NET VB]

```
Sub DshInitRcpInfo (
        ByRef info As dsh_info.TRCP_INFO,
        ByVal rcpspec As String,
        ByVal rcpbody As String,
        ByVal para_count As Int32)
```

[.NET C#]

```
void DshInitRcpInfo(
        ref TRCP_INFO info,
        byte[] rcpspec,
        byte[] rcpbody,
        int para_count );
```

（2）引数

info

レシピ TRCP_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。

rcpspec

設定するレシピ ID（文字列）です。

rcpbody

ﾚｼﾋﾟの RCPBODY 情報です。

para_count

TRCP_INFO に含まれるパラメータの最大数です。

（3）戻り値

なし。

（4）説明

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

また、レシピのパラメータ情報の設定については次の DshPutRcpPara()関数を使ってください。

TRCP_INFO 構造体の使用後、DshFreeTRCP_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。

３．13. ３．20 DshPutRcpPara() -レシピパラメータ情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutRcpPara(
        TRCP_INFO   *info,              // ﾚｼﾋﾟ情報構造体のﾎﾟｲﾝﾀ
        char        *rcpparnm,          // ﾚｼﾋﾟﾊﾟﾗﾒｰﾀ名
        int         para_fmt,           // para 値の format
        int         para_size,          // para 値の配列ｻｲｽﾞ
        void        *rcpparval          // para 値が格納されている領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshPutRcpPara (
          ByRef info As dsh_info.TRCP_INFO,
          ByVal rcpparnm As String,
          ByVal para_fmt As Int32,
          ByVal para_size As Int32,
          ByVal rcpparval As IntPtr) As Int32
```

[.NET C#]

```
int DshPutRcpPara(
          ref TRCP_INFO info,
          byte[] rcpparnm,
          int para_fmt,
          int para_size,
          byte[] rcpparval );
```

（2）引数

info

レシピ情報構造体のポインタです。

rcpparnm

レシピパラメータの名前です。

para_fmt

パラメータ値のフォーマットです。

para_size

パラメータ値の配列サイズです。

rcpparval

パラメータ値が格納されている領域のポインタです。

（3）戻り値

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

（4）説明

先に DshInitRcpInfo()で初期設定されたレシピパラメータ情報リスト para_list にパラメータ情報 TRCP_PARA を１個加えます。

para_list 内のパラメータ情報リスト上に、本関数が実行される順に情報を追加していきます。
設定後 0 を返却します。
もし、info 内の para_count で指定された分のパラメータが既に設定済みであった場合は、(-1)を返却します。

3．13. 3．21 DshMakeS15F13Response() - S15F13 の応答メッセージの生成

（1）呼出書式

[c,C++]

```
API int APIX DshMakeS15F13Response(
        TRCP_INFO *info,                // ﾚｼﾋﾟ情報格納領域ﾎﾟｲﾝﾀ
        TRCP_ERR_INFO *erinfo,          // S15F14 に設定する応答情報か格納領域のﾎﾟｲﾝﾀ
        DSHMSG *smsg,                   // S15F14 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,                   // S15F14 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 DshMakeS15F13Response (
          ByRef pinfo As dsh_info.TRCP_INFO,
          ByRef erinfo As dsh_info.TRCP_ERR_INFO,
          ByRef smsg As dshdr2.DSHMSG,
          ByRef buff As Byte,
          ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS15F13Response(
          ref TRCP_INFO pinfo,
          ref TRCP_ERR_INFO erinfo,
          ref DSHMSG smsg,
          byte[] buff,
          int buff_size );
```

（2）引数

info

レシピ情報が格納されている領域のポインタです。

erinfo

S15F14 メッセージに設定する応答情報が格納されている領域のポインタです。

msg

S15F14 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

S15F14 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

S15F13 に対する S15F14 応答メッセージを info に含まれるレシピ情報と erinfo に含まれる応答情報に従って作成します。

応答情報に含まれている rmack を S15F14 の RMACK として設定します。
rmack はユーザがレシピ情報を評価した結果です。

erinfo 応答情報の初期設定と情報設定にはそれぞれ DshInitTOBJ_ERR_INFO()、DshPutTOBJ_ERR_INFO()関数を使用することができます。(3.21 を参照してください)

３．13. ３．22 DshFreeTRCP_ERR_INFO() - レシピ関連応答情報構造体メモリの開放

（1）呼出書式

[c,C++]

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

[.NET VB]

```
Sub DshFreeTRCP_ERR_INFO (
        ByRef info As dsh_info.TRCP_ERR_INFO)
```

[.NET C#]

```
void DshFreeTRCP_ERR_INFO(
        ref TRCP_ERR_INFO info );
```

（2）引数

erinfo

メモリを解放したいレシピ関連応答情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

S15F13などの応答メッセージデコード時にTRCP_ERR_INFO構造体内で情報格納用に割当て使用されているメモリを全て解放します。
開放した後、TRCP_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。

3．13. 3．23 DshDecodeS15F18() - S15F18 応答メッセージのデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeS15F18(
        DSHMSG *smsg,                   // SECS 応答ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TRCP_S15F18_INFO *erinfo        // 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 DshDecodeS15F18 (
        ByRef smsg As dshdr2.DSHMSG,
        ByRef info As dsh_info.TRCP_S15F18_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS15F18(
        ref DSHMSG smsg,
        ref TRCP_S15F18_INFO info );
```

（2）引数

smsg

S15F18 の SECS 応答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

デコードしたレシピ情報を格納する TRCP_S15F18_INFO 構造体のポインタです。

（3）戻り値

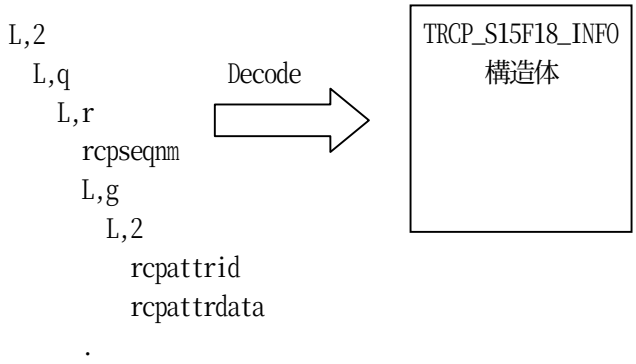
戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

S15F17 の応答メッセージに含まれるレシピ関連メッセージの情報を、ユーザプログラムが処理しやすい TRCP_S15F18_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 DshFreeTRCP_S15F18_INFO()関数を使って開放してください。

smsg S15F18



３．13. ３．24 DshFreeTRCP_S15F18_INFO() – S15F18 応答情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_S15F18_INFO(
      TRCP_S15F18_INFO *erinfo      // ﾒﾓﾘを開放したい応答情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTRCP_S15F18_INFO (
        ByRef info As dsh_info.TRCP_S15F18_INFO)
```

[.NET C#]

```
void DshFreeTRCP_S15F18_INFO(
        ref TRCP_S15F18_INFO info );
```

（2）引数

erinfo

メモリを解放したいレシピ関連応答情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

S15F17 の応答メッセージデコード時に TRCP_S15F18_INFO 構造体内で情報格納用に割当て使用されているメモリを全て解放します。

開放した後、TRCP_S15F18_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。

3．13. 3．25 DshInitTRCP_ERR_INFO () - レシピ応答情報の初期化

（1）呼出書式

[c,C++]

```
API void APIX   DshInitTRCP_ERR_INFO(
        TRCP_ERR_INFO *info,                // ｴﾗｰ情報格納構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int         rmack,                  // ackﾃﾞｰﾀ
        int         err_count               // ｴﾗｰ情報のﾘｽﾄｻｲｽﾞ（個数  0,1,2...）
);
```

[.NET VB]

```
Sub DshInitTRCP_ERR_INFO (
        ByRef rcp_errinfo As dsh_info.TRCP_ERR_INFO,
        ByVal rmack As Int32,
        ByVal errcount As Int32)
```

[.NET C#]

```
void DshInitTRCP_ERR_INFO(
        ref TRCP_ERR_INFO rcp_errinfo,
        int rmack,
        int errcount );
```

（2）引数

info

TRCP_ERR_INFO 応答情報構造体のポインタです。

rmack

rmack - ACK の値です。

err_count

エラー情報構造体の数です。 = 0 の場合はエラー情報がないことになります。

（3）戻り値

なし。

（4）説明

本関数は、レシピ関連応答メッセージ TRCP_ERR_INFO 構造体に初期設定を行います。
info で指定された構造体の rmack メンバーに引数 rmack の値を設定し、err_count メンバーにも引数 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 TRCP_ERR_INFO エラー情報構造体のポインタリストを設けます。

3．13. 3．26 DshPutTRCP_ERR_INFO () - レシピエラー情報の設定

（1）呼出書式

[c,C++]

```
API void APIX   DshPutTRCP_ERR_INFO (
        TRCP_ERR_INFO  *errinfo,            // ｴﾗｰ情報格納構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int         errcode,                // error code
        char        *errtext                // error text
);
```

[.NET VB]

```
Function DshPutTRCP_ERR_INFO (
        ByRef errinfo As dsh_info.TRCP_ERR_INFO,
        ByVal errcode As Int32,
        ByVal errtext As String) As Int32
```

[.NET C#]

```
int DshPutTRCP_ERR_INFO(
        ref TRCP_ERR_INFO errinfo,
        int errcode,
        byte[] errtext );
```

（2）引数

errinfo

レシピエラー情報構造体のポインタです。

errcode

設定するエラーコードです。(メッセージ内のアイテムは U1(51)です。)

errtext

設定するエラーテキスト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 errlist 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１個 TRCP_ERR_INFO 構造体領域を設け、その構造体に errcode と errtext を格納し、0 を返却します。TRCP_ERR_INFO と内部メンバーのメモリは本関数が取得します。

本関数の実行前に DshInitTRCP_ERR_INFO()関数を使って errinfo を初期化しておく必要があります。

3．13．４　ユーザ作成ライブラリ関数

3．13．４．１　DshResponseS15F4() – S15F4 レシピネームスペースアクション応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS15F4(
        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
        TRCP_ACT_INFO *info,            // ﾚｼﾋﾟﾈｰﾑｽﾍﾟｰｽｱｸｼｮﾝﾒｯｾｰｼﾞ格納ﾎﾟｲﾝﾀ
        TRCP_ERR_INFO *erinfo           // S15F4 応答情報格納用構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshResponseS15F4 (
        ByVal eqid As Int32,
        ByVal trid As Int32,
        ByRef info As dsh_info.TRCP_ACT_INFO,
        ByRef erinfo As dsh_info.TRCP_ERR_INFO) As Int32
```

[.NET C#]

```
int DshResponseS15F4(
        int eqid,
        uint trid,
        ref TRCP_ACT_INFO info,
        ref TRCP_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

S15F3 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

レシピネームスペースアクションの表示メッセージ情報が格納されているポインタです。

erinfo

送信する応答メッセージのための応答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	正常に送信できた。
(-1)	送信できなかった。

（4）説明

レシピネームスペースアクションメッセージ S15F3 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 erinfo 情報から S15F4 メッセージを組み立て、その後、S15F4 メッセージを送信します。

なお、S15F4 メッセージの組み立てに、DshMakeS15F3Response()関数を使用できます。

3．13．４．２　DshResponseS15F6() - S15F6 レシピネームスペースリネーム応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS15F6(
        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
        TRCP_RENAME_INFO *info,         // ﾚｼﾋﾟﾈｰﾑｽﾍﾟｰｽﾘﾈｰﾑﾒｯｾｰｼﾞ格納ﾎﾟｲﾝﾀ
        TRCP_ERR_INFO *erinfo           // S15F6 応答情報格納用構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshResponseS15F6 (
        ByVal eqid As Int32,
        ByVal trid As Int32,
        ByRef info As dsh_info.TRCP_RENAME_INFO,
        ByRef erinfo As dsh_info.TRCP_ERR_INFO) As Int32
```

[.NET C#]

```
int DshResponseS15F6(
        int eqid,
        uint trid,
        ref TRCP_RENAME_INFO info,
        ref TRCP_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

S15F5 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

レシピネームスペースリネームの表示メッセージ情報が格納されているポインタです。

erinfo

送信する応答メッセージのための応答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	正常に送信できた。
(-1)	送信できなかった。

（4）説明

レシピネームスペースリネームメッセージ S15F5 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 erinfo 情報から S15F6 メッセージを組み立て、その後、S15F6 メッセージを送信します。

なお、S15F6 メッセージの組み立てに、DshMakeS15F5Response()関数を使用できます。

3．13．４．３　DshResponseS15F14() – S15F14 レシピ生成要求応答メッセージ

（1）呼出書式

[c,C++]

```
API int APIX DshResponseS15F14(
        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
        TRCP_INFO *info,                // ﾚｼﾋﾟ生成要求ﾒｯｾｰｼﾞ格納ﾎﾟｲﾝﾀ
        TRCP_ERR_INFO *erinfo           // S15F14 応答情報格納用構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshResponseS15F14 (
          ByVal eqid As Int32,
          ByVal trid As Int32,
          ByRef info As dsh_info.TRCP_INFO,
          ByRef erinfo As dsh_info.TRCP_ERR_INFO) As Int32
```

[.NET C#]

```
int DshResponseS15F14(
          int eqid,
          uint trid,
          ref TRCP_INFO info,
          ref TRCP_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

S15F13 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

レシピ生成要求の表示メッセージ情報が格納されているポインタです。

erinfo

送信する応答メッセージのための応答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	正常に送信できた。
(-1)	送信できなかった。

（4）説明

レシピ生成要求メッセージ S15F13 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 erinfo 情報から S15F14 メッセージを組み立て、その後、S15F14 メッセージを送信します。

なお、S15F14 メッセージの組み立てに、DshMakeS15F13Response()関数を使用できます。