

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- 4 / 15

- 3 . 6 CE 収集イベント情報アクセス、関連メッセージ送信関数
- 3 . 7 Report レポート情報アクセス、関連メッセージ送信関数
- 3 . 8 アラーム情報アクセス、関連メッセージ送信関数

2009年6月

株式会社データマップ



### [ 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

3.6	CE 収集イベント情報アクセスと通知関数.....	2
3.6.1	使用する構造体.....	4
3.6.2	CE 収集イベント情報アクセスと通知関数.....	5
3.6.2.1	GemGetCeName() – イベント名取得関数.....	5
3.6.2.2	GemGetCeEnabled() – イベント有効状態取得関数.....	6
3.6.2.3	GemSetCeEnabled() – イベント有効状態設定関数.....	7
3.6.2.4	GemGetCeRpCount() – イベントにリンクされているレポート数取得関数.....	8
3.6.2.5	GemGetCeRpid() – イベントにリンクされている指定順位のレポートID取得関数.....	9
3.6.2.6	GemGetCeAllRpid – イベントにリンクされている全レポートID取得関数.....	10
3.6.2.7	GemGetCeRpName() – イベントにリンクされている指定順位のレポート名取得関数.....	11
3.6.2.8	GemSetCeRpLink () – イベントのレポートリンクの設定関数.....	12
3.6.2.9	GemGetReservedCeid – 予約収集イベントのCEID取得関数.....	13
3.6.2.10	GemGetCeList() – 全登録CEID取得関数.....	15
3.6.2.11	GemSendS2F35() – リンクイベントレポート送信.....	16
3.6.2.12	GemSendS2F37() – 有効/無効イベントレポート送信.....	19
3.6.2.13	GemSendS6F15() – イベントレポート要求.....	22
3.6.2.14	GemNotifyEvent() – 収集イベント通知要求.....	25
3.6.2.15	GemAnNotifyEvent() – 注釈付き収集イベント通知要求.....	27
3.6.3	CE 収集イベント関連ライブラリ関数.....	28
3.6.3.1	DshDecodeS6F11() - S6F11 イベントレポートのデコード.....	28
3.6.3.2	DshMakeS6F11Response() - S6F11 の応答メッセージの生成.....	29
3.6.3.3	DshDecodeS6F13() - S6F13 注釈付きイベントレポートのデコード.....	30
3.6.3.4	DshMakeS6F13Response() - S6F13 の応答メッセージの生成.....	31
3.6.3.5	DshFreeTCE_CONTENT () – 収集イベント情報構造体メモリの開放.....	32
3.6.3.6	DshInitTCE_LIST – イベントリンク情報リストの初期設定.....	33
3.6.3.7	DshInitTCE_LINK – イベントレポートリンク情報の初期設定.....	35
3.6.3.8	DshAddTCE_LINK – イベントレポートリンク情報にレポートIDを追加する.....	36
3.6.3.9	DshFreeTCE_LIST () – イベントリンクレポート情報リスト構造体メモリの開放.....	37
3.6.3.10	DshEncodeS2F35() – イベント定義情報をS2F35へエンコード.....	38
3.6.3.11	DshDecodeS2F35() – S2F35をイベント定義情報へデコード.....	40
3.6.3.12	DshEncodeS2F37() – イベント有効/無効情報のS2F37へエンコード.....	41
3.6.3.13	DshDecodeS2F37() – S2F37のイベント有効/無効情報のデコード.....	43
3.6.4	ユーザ作成ライブラリ関数.....	45
3.6.4.1	DshResponseS6F12() – S6F12 イベントレポート送信応答メッセージ.....	45
3.6.4.2	DshResponseS6F14() – S6F14 注釈付きイベントレポート送信応答メッセージ.....	46
3.7	Report レポート情報アクセス関数.....	47
3.7.1	使用する構造体.....	47
3.7.2	レポート情報アクセスと通知関数.....	48
3.7.2.1	GemGetRpName() – レポート名取得関数.....	48
3.7.2.2	GemGetRpVCount() – レポートにリンクされている変数の数の取得関数.....	49
3.7.2.3	GemGetRpVid() – レポートにリンクされている指定順位の変数ID取得関数.....	50
3.7.2.4	GemGetRpAllVid - RP最小(Min)値取得関数.....	51
3.7.2.5	GemGetRpVName() – レポートにリンクされている指定順位の変数名取得関数.....	52
3.7.2.6	Gem SetRpVLink () – レポートの変数リンクの設定関数.....	53
3.7.2.7	GemGetRpList() – 全登録レポートID取得関数.....	54

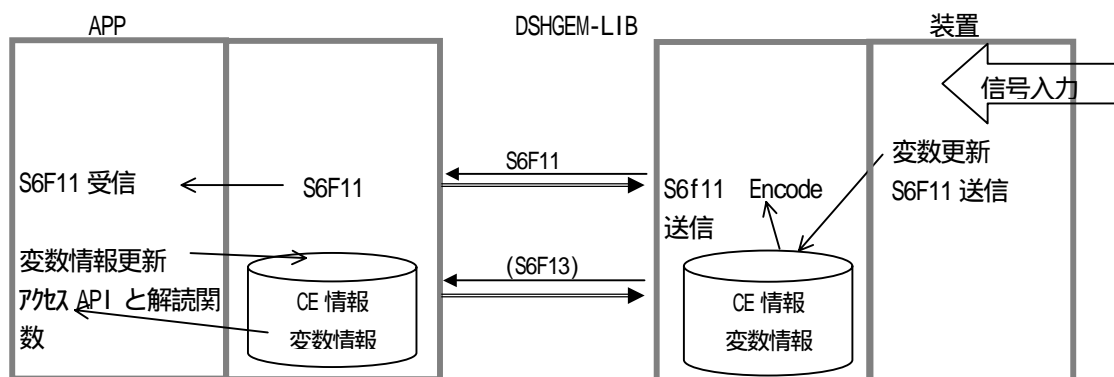
3.7.2.8	GemSendS2F330	- レポート設定送信	55
3.7.2.9	GemSendS6F190	- 個別レポート要求メッセージ送信	58
3.7.3	レポート関連ライブラリ関数		61
3.7.3.1	DshInitTRP_LIST	- レポートリンク情報リストの初期設定	61
3.7.3.2	DshInitTRP_LINK	- レポートリンク情報の初期設定	63
3.7.3.3	DshAddTRP_LINK	- レポートリンク情報に変数IDを追加する	64
3.7.3.4	DshFreeTRP_LIST ()	- レポート情報リスト構造体メモリの開放	65
3.7.3.5	DshEncodeS2F330	- レポート定義情報をS2F33ヘエンコード	66
3.7.3.6	DshDecodeS2F330	- S2F33をレポート定義情報ヘデコード	68
3.7.3.7	DshDecodeS6F200	- S6F20をレポートデータ情報ヘデコード	69
3.8	Alarm アラーム情報アクセスと通知関数		71
3.8.1	使用する構造体		72
3.8.2	Alarm アラーム情報アクセス関数		73
3.8.2.1	GemGetAlName()	- アラーム名取得関数	73
3.8.2.2	GemGetAlEnabled()	- アラーム有効状態取得関数	74
3.8.2.3	GemSetAlEnabled()	- アラーム有効状態設定関数	75
3.8.2.4	GemGetAlcd()	- アラームのALCD取得関数	76
3.8.2.5	GemGetAltx()	- アラームのALTX取得関数	77
3.8.2.6	GemGetAlceOn()	- アラーム発生時のリンクイベントID取得関数	78
3.8.2.7	GemGetAlceOff()	- アラーム復旧時のリンクイベントID取得関数	79
3.8.2.8	GemGetAlList()	- 全登録アラームID取得関数	80
3.8.2.9	GemSendS5F30	- 有効/無効アラーム送信	81
3.8.2.10	GemSendS5F50	- アラームリスト要求メッセージ送信	84
3.8.2.11	GemNotifyAlarm()	- アラーム通知要求	87
3.8.3	アラーム関連ライブラリ関数		89
3.8.3.1	DshDecodeS5F10	- S5F1 アラームレポートのデコード	89
3.8.3.2	DshMakeS5F1Response()	- S5F1 の応答メッセージの生成	90
3.8.3.3	DshFreeTAL_S5F1_INFO ()	- アラーム情報構造体メモリの開放	91
3.8.3.4	DshDecodeS5F60	- S5F6 アラームリストのデコード	92
3.8.3.5	DshFreeTAL_S5F6_LIST ()	- アラームリスト構造体メモリの開放	93
3.8.4	ユーザ作成ライブラリ関数		94
3.8.4.1	DshResponseS5F20	- S5F2 アラーム報告送信応答メッセージ	94

(VOL - 5 に続く)



### 3.6 CE 収集イベント情報アクセスと通知関数

CE 収集イベント情報は、DSHGEM-LIB が管理します。従って、APP はこれらの情報をアクセスするために以下の DSHGEM-LIB API 関数を使用します。



装置からのイベントレポート (S6F11, S6F13) 受信した際、ライブラリ関数を使ってメッセージに含まれているイベント情報(レポート ID, 変数値)を TCE\_CONTENT 構造体にデコードすることができます。デコードすることによって APP は構造体のメンバーをアクセスすることによって処理が簡単になります。



#### (1) 情報アクセスと送信 API 関数

収集イベント情報のアクセスと装置へのメッセージ送信に関連するサービスのための API 関数名は次の一覧表のとおりです。

	API 関数名	機能
1	GemGetCeName()	指定された CEID の収集イベント名を取得します。
2	GemGetCeEnabled()	指定された CEID の Enable 状態を取得します。
3	GemSetCeEnabled()	指定された CEID の Enable 状態を設定します。
4	GemGetCeRpCount()	指定された CEID にリンクされているポート ID の個数を取得します。
5	GemGetCeRpid()	指定された CEID にリンクされているポート ID を取得します。
6	GemGetCeAllRpid()	指定された CEID にリンクされている全ポート ID を取得します。
7	GemSetReservedCeid()	指定されたインデックスの予約 CEID を設定します。
8	GemGetReservedCeid()	指定されたインデックスの予約 CEID を取得します。
9	GemGetCeList()	全登録 CEID の ID リストを取得する。
10	GemSendS2F35()	S2F35 イベントポート設定メッセージを送信します。
11	GemSendS2F37()	S2F37 イベント無効/有効設定メッセージを送信します。
12	GemSendS6F15()	S6F15 イベントポート要求メッセージを送信します。
13	GemNotifyEvent()	S6F11 収集イベント通知を送信します。
14	GemAnNotifyEvent()	S6F13 注釈付き収集イベント通知を送信します。

## (2) イベント関連ライブラリ関数

APP が使用できるイベント情報処理関連ライブラリ関数として、以下の関数があります。

	API 関数名	機能
1	DshDecodeS6F11()	S6F11 イベントレポートメッセージをデコードします。(TCE_CONTENT 構造体に)
2	DshResponseS6F12()	S6F12 応答メッセージを送信します。(ユーザライブラリ u_s6f11.c)
3	DshMakeS6F11Response()	S6F12 応答メッセージを作ります。
4	DshDecodeS6F13()	S6F11 イベントレポートメッセージをデコードします。(TCE_CONTENT 構造体に)
5	DshResponseS6F14()	S6F14 応答メッセージを送信します。(ユーザライブラリ u_s6f13.c)
6	DshMakeS6F13Response()	S6F14 応答メッセージを作ります。
7	DshFreeTCE_CONTENT()	TCE_CONTENT 構造体内に使用されているメモリを開放します。
8	DshInitTCE_LIST()	TCE_LIST 構造体の初期設定をします。
9	DshInitTCE_LINK()	TCE_LINK 構造体の初期設定をします。
10	DshAddTCE_LINK()	TCE_LINK 構造体にリンクレポート ID を追加します。
11	DshFreeTCE_LIST ()	TCE_LIST 構造体内部で仕様されているメモリを開放します。
12	DshEncodeS2F35()	TCE_LIST 構造体の内容に従って S2F35 メッセージをエンコードします。
13	DshDecodeS2F35()	S2F35 メッセージの内容を TCE_LIST 構造体内にデコードします。
14	DshEncodeS2F37()	CEID リスト内のリストの CEED(有効/無効)情報から S2F37 メッセージをエンコードします。
15	DshDecodeS2F37()	S2F37 メッセージ内の CEID と CEED(有効/無効)をデコードします。

(注)

S2F35 (Link Event Report), S2F37 ( Enable/Disable Event Report)メッセージの受信に対して、DSHGEM-LIB は自動的に DSHGEM-LIB 内で処理をし、処理結果に基づき応答メッセージを返します。従って、APP は S2F35, S2F37 メッセージに関連する関数を使って処理することはありません。

S2F35 については、含まれているリンク情報に従って、DSHEng 4 が管理している CEID に対するレポート ID の再リンクを行います。

S2F37 については、同様に、含まれている CEID, Enable/Disable 情報に従って、イベントの有効・無効の設定を行います。(無効の場合には、APP からその CEID のイベント通知要求があってもイベント通知メッセージを送信しません。)

## (3) ユーザ作成ライブラリ関数

	ライブラリ関数名	機能
1	DshResponseS6F12()	S6F11 イベントレポート送信応答
2	DshResponseS6F14()	S6F13 注釈付きイベントレポート送信応答

### 3.6.1 使用する構造体

(1) S6F11, S6F13 メッセージのデコード結果を格納する構造体です。

```
typedef struct{
    TCEID      ceid;          // 収集イベント ID
    int        rp_count;     // リンクされているレポート数
    TRP_CONTENT **rp_list;   // リンクレポート情報のリスト
    int        next_rp;     // (関数内部使用)
    int        next_v;     // (関数内部使用)
} TCE_CONTENT;

typedef struct{
    TRPID      rpid;        // リンクされているレポート ID
    int        v_count;     // レポートにリンクされている変数の数
    TV_CONTENT **v_list;    // リンク変数情報のリスト
} TRP_CONTENT;

typedef struct{
    TVID       vid;        // リンクされている変数 ID
    int        format;     // 変数のフォーマット
    int        asize;      // 変数データの配列サイズ
    void       *dptr;      // 変数データのポインタ (fmt L ならば nesting ->TV_CONTENT)
} TV_CONTENT;
```

(2) S2F35 イベントレポート設定、S2F33 レポート設定メッセージ処理に使用する構造体です。

```
typedef struct{
    int        count;       // 本構造体を含む CE の数
    TCE_LINK  **ce_list;   // CE リンク情報構造体のリストポインタ
} TCE_LIST;

typedef struct{
    TCEID      ceid;       // CEID
    int        count;     // リンクされているレポートの数
    TRPID      *rpid_list; // リンクされているレポート ID のリスト
} TCE_LINK;

typedef struct{
    int        count;       // 本構造体に含まれるレポートの数
    TRP_LINK  **rp_list;   // レポートリンク構造体のリストポインタ
} TRP_LIST;

typedef struct{
    TRPID      rpid;       // RPID
    int        count;     // リンクされている変数の数
    TVID       *vid_list;  // リンクされている VID のリスト
} TRP_LINK;
```



### 3.6.2 CE 収集イベント情報アクセスと通知関数

#### 3.6.2.1 GemGetCeName() イベント名取得関数

##### (1) 呼出書式

###### [C, C++]

```
API int APIX GemGetCeName(
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TCEID ceid,             // CEID
    char *name,              // 取得した名前の格納領域① イタ
    int *bytesize           // 名前の文字列長格納用
);
```

###### [.NET VB]

```
Function GemGetCeName (
    ByVal eqid As Int32,
    ByVal ceid As Int32,
    ByVal name As String) As Int32
```

###### [.NET C#]

```
int GemGetCeName(
    int eqid,
    uint ceid,
    byte[] name );
```

##### (2) 引数

eqid

GEM 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。

ceid

イベント ID です。

装置管理情報定義ファイルに登録されているイベントの ID でなければなりません。

name

取得した名前 (文字列) を格納する領域のポインタです。名前格納に十分な領域を準備してください。

bytesize

取得された名前の文字列長です。

##### (3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	ceid が正しくなかった。

##### (4) 説明

ceid で指定された CE (イベント) の名前を取得します。

正常に取得できた場合は、関数の戻り値は 0 になります。取得された名前のバイト長は bytesize で指定された領域に返却されます。

### 3.6.2.2 GemGetCeEnabled() イベント有効状態取得関数

#### (1) 呼出書式

##### [C, C++]

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

##### [.NET VB]

```
Function GemGetCeEnabled (
    ByVal eqid As Int32,
    ByVal ceid As Int32) As Int32
```

##### [.NET C#]

```
int GemGetCeEnabled(
    int eqid,
    uint ceid );
```

#### (2) 引数

eqid

GEM 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。

ceid

イベント ID です。

装置管理情報定義ファイルに登録されているイベントの ID でなければなりません。

#### (3) 戻り値

戻り値	意味
0	有効状態 0=無効、1=有効
(-1)	ceid の値が正しくなかった。

#### (4) 説明

ceid で指定された CE(イベント)の有効状態を取得します。

戻り値が =1 の場合、有効状態です。GemNotifyEvent()によってイベント通知できる状態です。

戻り値が =0 の場合、無効状態です。GemNotifyEvent()によってイベント通知できない状態です。

装置に対し CE の送信有効状態の通知 (送信) は GemSendS2F37()関数を使って行うことができます。

### 3.6.2.3 GemSetCeEnabled() イベント有効状態設定関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemStCenabled(
    int eqid,           // 通信対象装置 ID(0,1,2,...)
    TCEID ceid,        // CEID
    int on_off         // 有効 On/Off
);
```

##### [.NET VB]

```
Function GemSetCeEnabled (
    ByVal eqid As Int32,
    ByVal ceid As Int32,
    ByVal on_off As Int32) As Int32
```

##### [.NET C#]

```
int GemSetCeEnabled(
    int eqid,
    uint ceid,
    int on_off );
```

#### (2) 引数

eqid

GEM 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。

ceid

イベント ID です。

装置管理情報定義ファイルに登録されているイベントの ID でなければなりません。

on\_off

有効 / 無効の設定したい状態を指定します。

=1 の場合、有効状態にします。GemNotifyEvent() によってイベント通知できる状態です。

=0 の場合、無効状態にします。GemNotifyEvent() によってイベント通知できない状態です。

#### (3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	ceid の値が正しくなかった。

#### (4) 説明

ceid で指定された CE (イベント) の有効状態を on\_off 値で設定します。

### 3.6.2.4 GemGetCeRpCount() イベントにリンクされているレポート数取得関数

(1) 呼出書式

**[C, C++]**

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

**[.NET VB]**

```
Function GemGetCeRpCount (
    ByVal eqid As Int32,
    ByVal ceid As Int32) As Int32
```

**[.NET C#]**

```
int GemGetCeRpCount(
    int eqid,
    uint ceid );
```

(2) 引数

eqid

GEM 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。

ceid

イベント ID です。

装置管理情報定義ファイルに登録されているイベントの ID でなければなりません。

(3) 戻り値

戻り値	意味
>= 0	リンクされているレポート数。
(-1)	ceid の値が正しくなかった。

(4) 説明

ceid で指定された CE(イベント)にリンクされているレポート ID の数を取得します。

=0 はリンクされているイベントがないことを意味します。

### 3.6.2.5 GemGetCeRpid() イベントにリンクされている指定順位のレポート ID 取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetCeRpid(
    int eqid,           // 通信対象装置 ID(0,1,2,...)
    TCEID ceid,        // CEID
    int order,         // 順位(0,1,2...)
    TRPID *rpid        // 取得したレポート ID 格納ポインタ
);
```

##### [.NET VB]

```
Function GemGetCeRpid (
    ByVal eqid As Int32,
    ByVal ceid As Int32,
    ByVal order As Int32,
    ByRef rpid As Int32) As Int32
```

##### [.NET C#]

```
int GemGetCeRpid(
    int eqid,
    uint ceid,
    int order,
    ref uint rpid );
```

#### (2) 引数

eqid

GEM 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。

ceid

イベント ID です。

装置管理情報定義ファイルに登録されているイベントの ID でなければなりません。

order

何番目のレポート ID を取得するかを指定します。先頭の指定は 0 です。

rpid

取得したレポート ID 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)	ceid の値が正しくなかった。または order 番目のレポートは無かった。

#### (4) 説明

ceid で指定された CE (イベント) にリンクされている order 番目のレポート ID を取得します。

### 3.6.2.6 GemGetCeAllRpid イベントにリンクされている全レポート ID 取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetCeMin(
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TCEID ceid,             // CEID
    TRPID *rpid             // リンクされている rpid を格納する領域のポインタ
);
```

##### [.NET VB]

```
Function GemGetCeAllRpid (
    ByVal eqid As Int32,
    ByVal ceid As Int32,
    ByRef rpid As Int32) As Int32
```

##### [.NET C#]

```
int GemGetCeAllRpid(
    int eqid,
    uint ceid,
    ref uint rpid );
```

#### (2) 引数

eqid

GEM 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。

ceid

イベント ID です。

装置管理情報定義ファイルに登録されているイベントの ID でなければなりません。

rpid

リンクされている全レポート ID を格納するための領域ポインタです。

リンクされている可能性のある最大数文の領域を準備してください。

#### (3) 戻り値

戻り値	意味
>=0	取得できたレポート ID 数
(-1)	ceid の値が正しくなかった。

#### (4) 説明

ceid で指定された CE(イベント)にリンクされている全レポート ID を取得します。

戻り値 = 0 の場合、リンクされているレポートがないことを意味します。

戻り値 > 0 の場合、戻り値がリンクされているレポート数で、それらの ID が rpid の領域に設定されます。

### 3.6.2.7 GemGetCeRpName() イベントにリンクされている指定順位のレポート名取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetCeRpName(
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TCEID ceid,              // CEID
    int order,               // 順位(0,1,2.. )
    char *rpname             // 取得したレポート名格納ポインタ
);
```

##### [.NET VB]

```
Function GemGetCeRpName (
    ByVal eqid As Int32,
    ByVal ceid As Int32,
    ByVal order As Int32,
    ByVal rpname As String) As Int32
```

##### [.NET C#]

```
int GemGetCeRpName(
    int eqid,
    uint ceid,
    int order,
    byte[] rpname );
```

#### (2) 引数

eqid

GEM 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。

ceid

イベント ID です。

装置管理情報定義ファイルに登録されているイベントの ID でなければなりません。

order

何番目のレポートの名前を取得するかを指定します。先頭の指定は=0 です。

rpname

取得したレポート名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)	ceid の値が正しくなかった。または order 番目のレポートは無かった。

#### (4) 説明

ceid で指定された CE (イベント) にリンクされている order 番目のレポートの名前を取得します。

### 3.6.2.8 GemSetCeRpLink () イベントのレポートリンクの設定関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX Gem SetCeRpLink (
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TCE_LIST *list          // リンク作成したい情報が格納されているリスト
);
```

##### [.NET VB]

```
Function GemSetCeRpLink (
    ByVal eqid As Int32,
    ByRef list As dsh_info.TCE_LIST) As Int32
```

##### [.NET C#]

```
int GemSetCeRpLink(
    int eqid,
    ref TCE_LIST list );
```

#### (2) 引数

eqid

GEM 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

リンク対象の CEID とリンクしたいレポート ID が含まれているリストのポインタです。

#### (3) 戻り値

戻り値	意味
= 0	正常に取得できた。
(-1)	ceid の値が正しくなかった。またはレポート ID は無かった。

#### (4) 説明

list に含まれる 1 個以上の CE リンク情報に従って CE に対するレポートのリンクを設定しなおします。

list には、リストに含まれる TCE\_LINK 構造体の数とポインタリストが含まれます。

TCE\_LINK 構造体には、1 個の CEID と、それにリンクされるレポートの数とレポート ID のリストが含まれます。



### 3.6.2.9 GemGetReservedCeid 予約収集イベントのCEID取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetReservedCeid(
    int eqid,           // 通信対象装置 ID(0,1,2,...)
    int index,         // 予約CEIDのindex
    TCEID *ceid        // 取得したCEID格納用
);
```

##### [.NET VB]

```
Function GemGetReservedCeid (
    ByVal eqid As Int32,
    ByVal index As Int32,
    ByRef ceid As Int32) As Int32
```

##### [.NET C#]

```
int GemGetReservedCeid(
    int eqid,
    int index,
    ref uint ceid );
```

#### (2) 引数

eqid

GEM 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

予約されている収集イベントに割当てられたインデクスです。

インデクスと予約 CE の対応はシステムで規定されています。(c 言語ではマクロ定義されています。)

インデクス値の範囲は、0~31 でなければなりません。

ceid

予約されたインデクスで指定された収集イベントの CEID 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	正常に設定できた。
(-1)	インデクス値が正しくなかったまたは CEID が設定されていなかった。

#### (4) 説明

あるイベント、例えば、装置との通信確立状態になったタイミングで通知するイベントなど、DSHGEMLIB 内で自動的に送信しなければならない収集イベントに予約された CEID を取得するための関数です。

index で指定された収集イベントに対し予約されている CEID を、ceid で指定された領域に格納し返却します。

ユーザは、このインデクスを使って、CEID を取得し、イベント通知するために使用することができます。

index はデフォルトで次のように予約されています。

インデクス値	C 言語マクロ	適用収集イベント
0	CEX_RSV_COMMUNICATING	通信確立時に通知するイベント
1	CEX_RSV_SPOOL_END	プール送信終了時に送信するイベント
2		
3		
~		
31		

### 3.6.2.10 GemGetCeList() 全登録 CEID 取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetCeList(
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TBIN_DLIST **list       // 取得リスト格納ポインタの格納ポインタ
);
```

##### [.NET VB]

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

##### [.NET C#]

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

#### (2) 引数

eqid

GEM 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 CEID が格納されている TBIN\_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全 CEID とその名前を TBIN\_DLIST 構造体に出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 CE 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、DshFreeTBIN\_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。

TBIN\_DLIST 構造体は次のとおりです。

```
typedef struct{
    int          count;                // 取得できた ID 数
    ULONG       *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
}TBIN_DLIST;
```

### 3.6.2.11 GemSendS2F35() リンクイベントレポート送信

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemSendS2F35(
    int      eqid,                // 通信対象装置 ID(0,1,2,...)
    TCE_LIST *list,              // ceid リンク情報リスト格納構造体のポインタ
    int      *lrack,             // S2F36 の LRACK の値格納用ポインタ
    int      (WINAPI *callback)(), // 実行終了時のコールバック関数
    ULONG upara                  // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function GemSendS2F35 (
    ByVal eqid As Int32,
    ByRef list As dsh_info.TCE_LIST,
    ByRef lrack As Int32,
    ByVal callback As vcallback.callback_S2F35,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int GemSendS2F35(
    int eqid,
    ref TCE_LIST list,
    ref int lrack,
    CallbackAck callback,
    uint upara );
```

#### (2) 引数

eqid

GEM 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

イベント ID にリンクするレポート ID リストが格納されている構造体のポインタです。

lrack

応答メッセージ S2F36 に含まれる LRACK を格納する領域のポインタです。

callback

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

この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) ブロックド : 正常に送信できた。 l rack に受信した応答 ack が格納されています。 (2) 非ブロックド : 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。(応答メッセージが得られなかった)

#### (4) 説明

装置に対し、list 内に設定された CEID(収集イベント ID)とそれにリンクされる RPID(レポート ID)情報を S2F35 メッセージを使って送信します。

list 内には 1 個以上の CEID とそれらにリンクされる RPID が格納されています。

CEID にリンクされる RPID の数は 0 個以上です。

正常に応答メッセージを受信できた場合は、それに含まれる LRACK の値を l rack 領域に返却します。

TCE\_LIST 構造体 list に CEID とそれにリンクされる RPID の設定するために以下のライブラリ関数を使用することができます。

```
DshInitTCE_LIST()
```

```
DshInitTCE_LINK()
```

```
DshAddTCE_LINK()
```

送信要求から S2F35 応答メッセージ受信までの制御は引数の S2F35 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S2F35 送信後、応答メッセージ S2F36 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、l rack に S2F36 の LRACK の値が渡されます。
あり	送信要求後、S2F35 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常であった場合、l rack に S2F36 の LRACK の値が渡されます。 エラーが検出された場合、(-1)が end_status にセットされます。

通信終了後、必要に応じて DshFreeTCE\_LIST()関数を使って、TCE\_LIST 内に使用されたメモリを開放してください。

```
DshFreeTCE_LIST(list);
```

#### (5) コールバック関数

[C,C++]

```
API int APIX callback(
    int eqid,                // 装置 ID
    int end_status,         // 実行結果
    int *l rack,            // LRACK 値が格納されているポインタ
    ULONG upara             // 呼出時に指定したパラメータ
);
```

[.NET VB]

```
Function callback_S2F35(ByVal eqid As Integer, ByVal end_status As Integer, ByRef l rack As Integer,
```

ByVal upara As Integer) As Integer

**[.NET C#]**

```
int CallbackAck(int eqid, int end_status, int *lrack, 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。
(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

end\_status = 0 のときのみ lrack の内容が有効です。

### 3.6.2.12 GemSendS2F37() 有効/無効イベントレポート送信

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemSendS2F37(
    int      eqid,                // 通信対象装置 ID(0,1,2,...)
    TCEID    *list,              // 有効/無効対象 CEID のリスト
    int      count,              // list に含まれる CEID の数
    int      ceed,               // 有効/無効(1/0)の指定
    int      *erack,             // S2F38 の ERACK 格納領域のポインタ
    int      (WINAPI *callback)(), // 実行終了時のコールバック関数
    ULONG    upara               // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function GemSendS2F37 (
    ByVal eqid As Int32,
    ByRef list As Int32,
    ByVal count As Int32,
    ByVal ceed As Int32,
    ByRef erack As Int32,
    ByVal callback As vcallback.callback_S2F37,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int GemSendS2F37(
    int eqid,
    ref uint list,
    int count,
    int ceed,
    ref int erack,
    CallbackAck callback,
    uint upara );
```

#### (2) 引数

eqid

GEM 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

有効/無効対象 CEID が入っているリストです。

count

list に入っている CEID の数です。

count=0 の場合は、装置の全 CEID が有効/無効の対象になります。

ceed

有効/無効の指定をします。1 が有効、0 が無効になります。

erack

応答メッセージ S2F38 に含まれる ERACK を格納する領域のポインタです。

callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。

ユーザは任意の関数名を指定できます。  
この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) ブロックモード：正常に送信できた。 erack に受信した応答 ack が格納されています。 (2) 非ブロックモード：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。(応答メッセージが得られなかった)

### (4) 説明

list 内に設定された CEID(収集イベント ID)に対し ceed で指定された有効/無効を S2F37 メッセージを使って装置に送信します。

list 内には count 個の CEID が含まれます。count = 0 の場合は、装置で定義される全 CEID が有効/無効の対象になります。

正常に応答メッセージを受信できた場合は、それに含まれる ERACK の値を erack 領域に返却します。送信要求から S2F37 応答メッセージ受信までの制御は引数の S2F37 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S2F37 送信後、応答メッセージ S2F38 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、erack に S2F38 の ERACK の値が渡されます。
あり	送信要求後、S2F37 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常であった場合、erack に S2F38 の ERACK の値が渡されます。 エラーが検出された場合、(-1)が end_status にセットされます。

### (5) コールバック関数

[c,C++]

```
API int APIX callback(
    int eqid,                // 装置 ID
    int end_status,         // 実行結果
    int *erack,             // ERACK 値が格納されているポインタ
    ULONG upara             // 呼出時に指定したパラメータ
);
```

[.NET VB]

```
Function callback_S2F37(ByVal eqid As Integer, ByVal end_status As Integer, ByRef erack As Integer,
    ByVal upara As Integer) As Integer
```



[.NET C#]

```
int CallbackAck(int eqid, int end_status, int *erack, 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。
(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

### 3.6.2.13 GemSendS6F15() イベントレポート要求

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemSendS6F15(
    int      eqid,                // 通信対象装置 ID(0,1,2,...)
    TCEID    ceid,                // 収集イベント ID
    TCE_CONTENT *info,           // list に含まれる CEID の数
    int      (WINAPI *callback)(), // 実行終了時のコールバック関数
    ULONG    upara                // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function GemSendS6F15 (
    ByVal eqid As Int32,
    ByVal ceid As Int32,
    ByRef info As dsh_info.TCE_CONTENT,
    ByVal callback As vcallback.callback_S6F15,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int GemSendS6F15(
    int eqid,
    uint ceid,
    ref TCE_CONTENT info,
    CallbackS6F15 callback,
    uint upara );
```

#### (2) 引数

eqid

GEM 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。

ceid

イベントレポート要求する対象の CEID です。

count

list に入っている CEID の数です。count=0 の場合は、装置の全 CEID が有効/無効の対象になります。

info

S6F16 メッセージをデコードしたイベント情報格納するための TCE\_CONTENT 構造体のポインタです。

callback

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

この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) プロケット : 正常に送信できた。 info にデコードされたイベント情報が格納されています。 (2) 非プロケット : 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。(応答メッセージが得られなかった)

### (4) 説明

S6F15 を送信して ceid で設定された収集イベント ID が有するイベントレポートを装置に求めるための関数です。応答メッセージ S6F16 は S6F11 とまったく同じ構造のものになります。

正常に応答メッセージを受信できた場合は、それに含まれるイベントレポート情報をデコードし、info 領域に返却します。

送信要求から S6F15 応答メッセージ受信までの制御は引数の S6F15 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S6F15 送信後、応答メッセージ S6F16 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、erack に S6F16 の ERACK の値が渡されます。
あり	送信要求後、S6F15 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常であった場合、S6F16 をデコードしに info が指す構造体にイベント情報を格納します。 エラーが検出された場合、(-1) が end_status にセットされます。

### (5) コールバック関数

#### [C, C++]

```
API int APIX callback(
    int eqid,                // 装置 ID
    int end_status,          // 実行結果
    TCE_CONTENT *info,      // 収集イベント情報格納構造体のポインタ
    ULONG upara              // 呼出時に指定したパラメータ
);
```

#### [.NET VB]

```
Function callback_S6F15(ByVal eqid As Integer, ByVal end_status As Integer, ByRef info As
dsh_info.TCE_CONTENT, ByVal upara As Integer) As Integer
```

#### [.NET C#]

```
int CallbackS6F15(int eqid, int end_status, ref TCE_CONTENT info, uint upara);
```

end\_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。
(-1)	送信エラーを検出した。

(-14)	T3タイムアウトを検出した。
-------	----------------

end\_status = 0 のときのみ info の内容が有効です。

### 3.6.2.14 GemNotifyEvent() 収集イベント通知要求

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemNotifyEvent(
    int      eqid,           // 装置 ID
    TCEID    ceid,           // 通知 ceid
    int (WINAPI *EventCallback)(), // 実行終了時の callback 関数
    ULONG    upara           // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function GemNotifyEvent (
    ByVal eqid As Int32,
    ByVal ceid As Int32,
    ByVal callback As vcallback.callback_NotifyEvent,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int GemNotifyEvent(
    int eqid,
    uint ceid,
    CallbackNotifyEvent NEventCallback,
    uint upara );
```

#### (2) 引数

eqid

装置 ID です。

ceid

ホストに通知するための CEID(収集イベント ID)を指定します。  
S6F11 の ceid 値になります。

EventCallback

DSHGEMLIB によるイベント通知処理が終了したときに呼出される callback 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。

本パラメータとして = 0 が指定された場合は、コールバック関数が無く、ブロックモードとみなされ、関数で指定されたメッセージの通信トランザクションが終了するまで制御は戻ってきません。

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することができます。

#### (3) 戻り値

戻り値	意味
0	(1) ブロックモード : 正常に送信でき、ackc6=0 であった。 (2) 非ブロックモード : 要求が受け付けられた。
9	キャンセルされた。
(-1)	要求が受け付けられなかったか送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

その他 > 0	ブロックモードの場合、ackc6 の値が返却されます。
---------	-----------------------------

(4) 説明

ホストに収集イベントメッセージ (S6F11) を送信するための関数です。

DSHGEMLIB は指定された ceid のメッセージを自動的に組立てた上でそれをホストに送信します。

ceid にリンクされているレポート、そしてそのレポートにリンクされている変数データ情報を S6F11 メッセージに設定展開した上で送信します。

定義されていない ceid が指定された場合、list=0 の S6F11 が送信されます。

ブロックモードならびにスプール関連については 3.2.4 の GemSendPrimary() 関数の説明を参照願います。

(5) コールバック関数

[C, C++]

```
API int APIX EventCallback(
    int end_status,           // 実行結果
    TCEID ceid,              // 通知した CEID
    ULONG upara              // 呼出時に指定したパラメータ
);
```

[.NET VB]

```
Function callback_NotifyEvent(ByVal eqid As Integer, ByVal end_status As Integer, ByVal ceid As Integer, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackNotifyEvent( int eqid, int end_status, uint id, 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。
9	スプールされた。
(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。
その他 > 0	ackc6 の値が返却されます。

### 3.6.2.15 GemAnNotifyEvent() 注釈付き収集イベント通知要求

#### (1) 呼出書式

##### [C, C++]

```
API int APIXGemAnNotifyEvent(
    int      eqid,           // 装置 ID
    TCEID    ceid,          // 通知 ceid
    int (WINAPI *EventCallback)(), // 実行終了時の callback 関数
    ULONG    upara          // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function GemNotifyAnEvent (
    ByVal eqid As Int32,
    ByVal ceid As Int32,
    ByVal callback As vcallback.callback_NotifyAnEvent,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int GemNotifyAnEvent(
    int eqid,
    uint ceid,
    CallbackNotifyEvent DshEndNotifyEvent,
    uint upara );
```

#### (2) 引数

GemgNotifyEvent()関数とまったく同じです。GemNotifyEvent()関数を参照ください。

#### (3) 戻り値

GemNotifyEvent()関数とまったく同じです。GemNotifyEvent()関数を参照ください。

#### (4) 説明

GemNotifyEvent()関数とまったく同じです。GemNotifyEvent()関数を参照ください。

#### (5) コールバック関数

GemNotifyEvent()関数とまったく同じです。GemNotifyEvent()関数を参照ください。

### 3.6.3 CE 収集イベント関連ライブラリ関数

#### 3.6.3.1 DshDecodeS6F11() - S6F11 イベントレポートのデコード

##### (1) 呼出書式

###### [C, C++]

```
API int APIX DshDecodeS6F11(
    int eqid, // 装置 ID
    DSHMSG *msg, // SECS メッセージ 情報構造体のポインタ
    TCE_CONTENT *pinfo // デコードした情報を格納する構造体のポインタ
);
```

###### [.NET VB]

```
Function DshDecodeS6F11 (
    ByVal eqid As Int32,
    ByRef msg As dshdr2.DSHMSG,
    ByRef info As dsh_info.TCE_CONTENT) As Int32
```

###### [.NET C#]

```
int DshDecodeS6F11(
    int eqid,
    ref DSHMSG msg,
    ref TCE_CONTENT info );
```

##### (2) 引数

eqid

GEM 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。

msg

S6F11 の SECS メッセージ情報が格納されている構造体のポインタです。

pinfo

デコードしたイベントレポート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	正常にデコードできた。
(-1)	msg を正しくデコードできなかった。

##### (4) 説明

S6F11 メッセージに含まれるイベントレポート情報を、ユーザプログラムが処理しやすい TCE\_CONTENT 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 DshFreeTCE\_CONTENT() 関数を使って開放してください。

msg S6F11

L,3

dataid

ceid

.





### 3.6.3.2 DshMakeS6F11Response() - S6F11 の応答メッセージの生成

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshMakeS6F11Response(
    int    eqid,                // 通信対象装置 ID(0,1,2,...)
    BYTE   ackc6,              // S6F12 に設定する ACKC6 です。
    DSHMSG *msg,              // S6F12 メッセージを格納するメッセージ構造体のポインタ
    BYTE   *buff,             // S6F12 のテキスト格納バッファポインタ
    int    buff_size          // buff のバイトサイズ
);
```

##### [.NET VB]

```
Function DshMakeS6F11Response (
    ByVal ackc6 As Byte,
    ByRef msg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buff_size As Int32) As Int32
```

##### [.NET C#]

```
int DshMakeS6F11Response(
    byte ackc6,
    ref DSHMSG msg,
    byte[] buff,
    int buff_size );
```

#### (2) 引数

ackc6

S6F12 メッセージの ACKC6 です。

msg

S6F12 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

S6F12 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

S6F11 に対する S6F12 応答メッセージを msg, buff 内に作成します。

応答情報は、ackc6 を S6F12 の ACKC6 として設定します。

ackc6 の設定はユーザがイベント情報を評価した結果です。

### 3.6.3.3 DshDecodeS6F13() - S6F13 注釈付きイベントレポートのデコード

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshDecodeS6F13(
    int eqid, // 装置 ID
    DSHMSG *smsg, // SECS メッセージ 情報構造体のポインタ
    TCE_CONTENT *pinfo // デコードした情報を格納する構造体のポインタ
);
```

##### [.NET VB]

```
Function DshDecodeS6F13 (
    ByVal eqid As Int32,
    ByRef smsg As dshdr2.DSHMSG,
    ByRef info As dsh_info.TCE_CONTENT) As Int32
```

##### [.NET C#]

```
int DshDecodeS6F13(
    int eqid,
    ref DSHMSG smsg,
    ref TCE_CONTENT info );
```

#### (2) 引数

eqid

GEM 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。

smsg

S6F13 の SECS メッセージ情報が格納されている構造体のポインタです。

pinfo

デコードしたイベントレポート情報を格納する構造体のポインタです。

#### (3) 戻り値

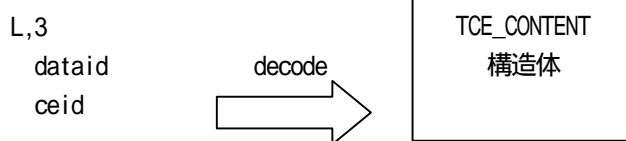
戻り値	意味
0	正常にデコードできた。
(-1)	smsg を正しくデコードできなかった。

#### (4) 説明

S6F13 メッセージに含まれるイベントレポート情報を、ユーザプログラムが処理しやすい TCE\_CONTENT 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 DshFreeTCE\_CONTENT() 関数を使って開放してください。

smsg S6F11



### 3.6.3.4 DshMakeS6F13Response() - S6F13 の応答メッセージの生成

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshMakeS6F13Response(
    int eqid, // 通信対象装置 ID(0,14,...)
    BYTE ackc6, // S6F14 に設定する ACKC6 です。
    DSHMSG *msg, // S6F14 メッセージを格納するメッセージ構造体のポインタ
    BYTE *buff, // S6F14 のテキスト格納バッファポインタ
    int buff_size // buff のバイトサイズ
);
```

##### [.NET VB]

```
Function DshMakeS6F13Response (
    ByVal ackc6 As Byte,
    ByRef msg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buff_size As Int32) As Int32
```

##### [.NET C#]

```
int DshMakeS6F13Response(
    byte ackc6,
    ref DSHMSG msg,
    byte[] buff,
    int buff_size );
```

#### (2) 引数

ackc6

S6F14 メッセージの ACKC6 です。

msg

S6F14 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

S6F14 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

S6F13 に対する S6F14 応答メッセージを msg, buff 内に作成します。

応答情報は、ackc6 を S6F14 の ACKC6 として設定します。

ackc6 の設定はユーザがイベント情報を評価した結果です。

### 3.6.3.5 DshFreeTCE\_CONTENT () 収集イベント情報構造体メモリの開放

#### (1) 呼出書式

##### [C, C++]

```
API void APIX DshFreeTCE_CONTENT(
    TCE_CONTENT *info // メモリを開放したい収集イベント情報構造体のポインタ
);
```

##### [.NET VB]

```
Sub DshFreeTCE_CONTENT (
    ByRef info As dsh_info.TCE_CONTENT)
```

##### [.NET C#]

```
void DshFreeTRP_CONTENT(
    ref TRP_CONTENT info );
```

#### (2) 引数

info

メモリを解放したい収集イベント情報構造体のポインタです。

#### (3) 戻り値

なし。

#### (4) 説明

TCE\_CONTENT 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、TCE\_CONTENT 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。

GemSendS6F11(), GemSendS6F13() API 関数の実行で得られた、あるいは GemSendS6F15 の応答メッセージで取得した収集イベント情報構造体を使用されているメモリのために使用します。

### 3.6.3.6 DshInitTCE\_LIST イベントリンク情報リストの初期設定

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshInitTCE_LIST(
    TCE_LIST *list,           // TCE_LIST 構造体のポインタ
    int count                // コマンドコード情報リストのサイズ
);
```

##### [.NET VB]

```
Sub DshInitTCE_LIST (
    ByRef list As dsh_info.TCE_LIST,
    ByVal count As Int32)
```

##### [.NET C#]

```
void DshInitTCE_LIST(
    ref TCE_LIST list,
    int count );
```

#### (2) 引数

list

イベントリンク情報リスト構造体のポインタです。このメンバーを初期設定します。

count

TCE\_LIST に含まれる CEID の最大数です。

#### (3) 戻り値

なし。

#### (4) 説明

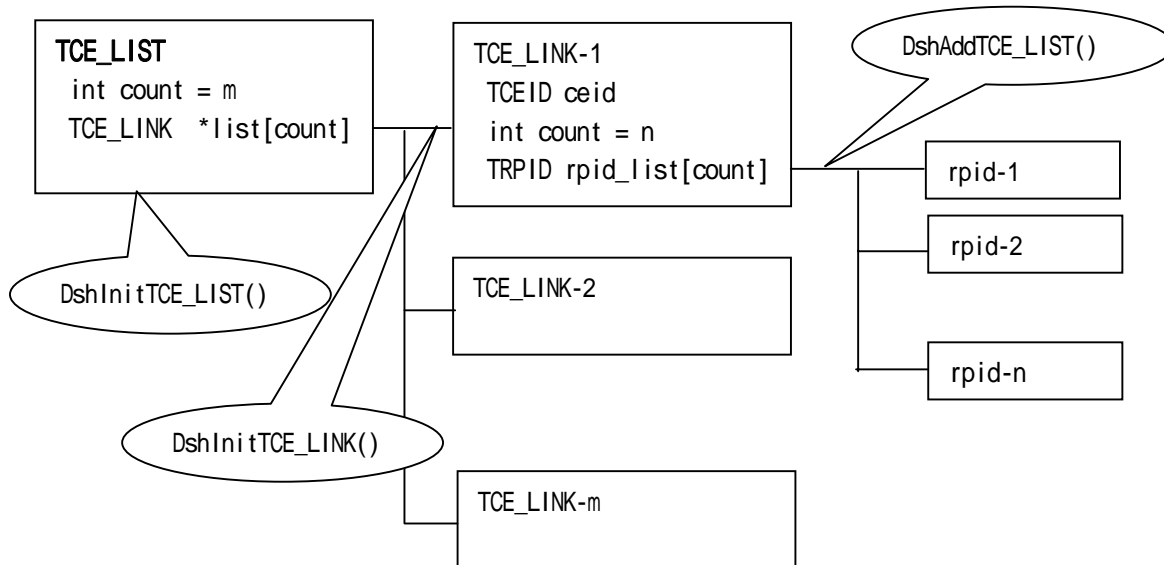
本関数は APP が S2F35 を送信する際のイベントリンク情報のためのリスト構造体を初期設定するために使用します。

最初に list 内をクリアします。そして、list 内に count 分のイベント ID に対するリンク情報を設定するための領域を確保します。

また、1 個の CEID に対するリンクレポート情報の設定は次の 2 つの関数を使ってください。

DshInitTCE\_LINK()

DshAddTCE\_LINK()



TCE\_LIST 構造体の使用後、DshFreeTCE\_LIST()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。

### 3.6.3.7 DshInitTCE\_LINK イベントレポートリンク情報の初期設定

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshInitTCE_LINK(
    TCE_LINK *list,           // TCE_LINK 構造体のポインタ
    int ce_order,           // list 内の LIST 位置
    TCEID ceid,             // TCE_LINK 内に設定する CEID
    int rp_count            // リポートリンク情報リストのサイズ
);
```

##### [.NET VB]

```
Function DshInitTCE_LINK (
    ByRef list As dsh_info.TCE_LIST,
    ByVal ce_order As Int32,
    ByVal ceid As Int32,
    ByVal rp_count As Int32) As Int32
```

##### [.NET C#]

```
int DshInitTCE_LINK(
    ref TCE_LIST list,
    int ce_order,
    uint ceid,
    int rp_count );
```

#### (2) 引数

list  
TCE\_LIST \*list, // TCE\_LIST 構造体のポインタ

ce\_order  
list 内のリスト位置を指定します。(0,1..)

ceid  
設定対象のイベント ID です。

rp\_count  
list の ce\_order 番目の CEID にリンクされる RPID の最大数です。  
(TCE\_LINK 構造体の count です。)

#### (3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	list 内の count の枠外の ce_order が指定された。

#### (4) 説明

本関数は APP が S2F35 を送信する際のイベントレポートリンク情報リスト構造体内の ce\_order 番目のリストに位置する TCE\_LINK 構造体の初期設定を行うために使用します。  
ce\_order 番目のリストに TCE\_LINK 領域用メモリを確保し、ceid と rp\_count を設定します。

### 3.6.3.8 DshAddTCE\_LINK イベントレポートリンク情報にレポート ID を追加する

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshAddCE_LINK(
    TCE_LINK *list,           // TCE_LINK 構造体のポインタ
    int ce_order,           // list 内の LIST 位置
    TRPID rpid              // TCE_LINK 内に加える RPID
);
```

##### [.NET VB]

```
Function DshAddTCE_LINK (
    ByRef list As dsh_info.TCE_LIST,
    ByVal ce_order As Int32,
    ByVal rpid As Int32) As Int32
```

##### [.NET C#]

```
int DshAddTCE_LINK(
    ref TCE_LIST list,
    int ce_order,
    uint rpid );
```

#### (2) 引数

list  
TCE\_LIST \*list, // TCE\_LIST 構造体のポインタ

ce\_order  
list 内のリスト位置を指定します。(0,1..)

rpid  
list 内の ce\_order 番目の TCE\_LINK 構造体内のリストに加えるレポート ID です。

#### (3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	list 内の count の枠外の ce_order が指定された。 または、既に DshInitTCE_LINK() で設定された rp_count 分のレポート ID が設定されてしまっている。

#### (4) 説明

本関数は APP が S2F35 を送信する際のイベントレポートリンク情報リスト構造体内の ce\_order 番目のリストに用意された TCE\_LINK 構造体内の rp\_list のリストに rpid を加えます。  
既に、rp\_count 分のレポート ID が設定されていた場合には、(-1) を返却します。



### 3.6.3.9 DshFreeTCE\_LIST () イベントリンクレポート情報リスト構造体メモリの開放

#### (1) 呼出書式

##### [C, C++]

```
API void APIX DshFreeTCE_LIST(  
    TCE_LIST *list    // メモリを開放したいイベントリンクレポート情報リスト構造体のポインタ  
);
```

##### [.NET VB]

```
Sub DshFreeTCE_LIST (  
    ByRef list As dsh_info.TCE_LIST)
```

##### [.NET C#]

```
void DshFreeTCE_LIST(  
    ref TCE_LIST list );
```

#### (2) 引数

list

メモリを解放したいイベントリンクレポート情報リスト構造体のポインタです。

#### (3) 戻り値

なし。

#### (4) 説明

TCE\_LIST 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、TCE\_LIST の内容を全て 0 で初期設定します。

list が NULL ならば、何も処理しません。

TCE\_LIST 構造体は GemSendS2F35() API 関数の実行時に使用されるものです。

### 3.6.3.10 DshEncodeS2F35() イベント定義情報を S2F35 へエンコード

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshEncodeS2F35(
    DSHMSG *msg,           // SECS メッセージ 情報構造体のポインタ
    BYTE *buffer,         // S2F35 を格納するバッファポインタ
    int buflen,           // buffer のバイトサイズ
    TCE_LIST *list        // エンコードしたいイベント ID リスト格納構造体のポインタ
);
```

##### [.NET VB]

```
Function DshEncodeS2F35 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buflen As Int32,
    ByRef list As dsh_info.TCE_LIST) As Int32
```

##### [.NET C#]

```
int DshEncodeS2F35(
    ref DSHMSG msg,
    byte[] buff,
    int buflen,
    ref TCE_LIST list );
```

#### (2) 引数

msg

エンコードした S2F35 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 S2F35 のテキストを格納するためのバッファポインタです。

buflen

buffer のバイトサイズです。

list

エンコードしたいイベント ID リストが格納されている構造体のポインタです。

list には、イベント ID とそのイベント ID にリンクされているレポート ID が格納されています。イベント ID の数が=0 の場合は、イベント定義が消去されます。また、イベント ID にリンクされるレポート ID の数が=0 の場合、そのイベント ID が消去されます。

#### (3) 戻り値

戻り値	意味
0	正常にエンコードできた。
(-1)	msg を正しくエンコードできなかった。

#### (4) 説明

TCE\_LIST 構造体に格納されているイベント定義情報を、S2F35 の SECS メッセージにエンコードします。buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。

smg S2F35

L,2  
dataid  
L,a  
L,2  
CEID  
L,c  
RPTID1  
RPTID2  
.  
RPTIDc

encode  
←



### 3.6.3.11 DshDecodeS2F35() S2F35 をイベント定義情報ヘデコード

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshDecodeS2F35(
    DSHMSG *msg, // S2F35 が格納されている SECS メッセージ 情報構造体のポインタ
    TCE_LIST *list // デコードしたイベント ID 定義情報リスト格納構造体のポインタ
);
```

##### [.NET VB]

```
Function DshDecodeS2F35 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef list As dsh_info.TCE_LIST) As Int32
```

##### [.NET C#]

```
int DshDecodeS2F35(
    ref DSHMSG msg,
    ref TCE_LIST list );
```

#### (2) 引数

msg

S2F35 メッセージが格納されているメッセージ情報構造体のポインタです。

list

デコードしたイベント ID リストの結果を格納するための構造体のポインタです。

#### (3) 戻り値

戻り値	意味
0	正常にデコードできた。
(-1)	msg を正しくデコードできなかった。

#### (4) 説明

S2F35 の SECS メッセージに含まれるイベント定義情報を list で指定される TCE\_LIST 構造体にデコードし格納します。

msg S2F35

L,2

dataid

L,a

L,2

CEID

L,c

RPTID1

RPTID2

.

RPTIDc

decode  
→



得られた list 情報の処理が終わった後は、DshFreeTCE\_LIST()関数で list 内部に割付使用されたメモリを開放することができます。

### 3.6.3.12 DshEncodeS2F37() イベント有効/無効情報の S2F37 へエンコード

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshEncodeS2F37(
    DSHMSG *msg,           // SECS メッセージ 情報構造体のポインタ
    BYTE *buffer,         // S2F37 を格納するバッファポインタ
    int buflen,           // buffer のバイトサイズ
    TCEID *list,           // エンコードしたいイベント ID リスト格納ポインタ
    int count,             // イベント ID リストに含まれる CEID の数
    int ceed               // 有効/無効(1/0)指定フラグ
);
```

##### [.NET VB]

```
Function DshEncodeS2F37 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buflen As Int32,
    ByRef list As Int32,
    ByVal count As Int32,
    ByVal ceed As Int32) As Int32
```

##### [.NET C#]

```
int DshEncodeS2F37(
    ref DSHMSG msg,
    byte[] buff,
    int buflen,
    ref uint list,
    int count,
    int ceed );
```

#### (2) 引数

msg

エンコードした S2F37 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 S2F37 のテキストを格納するためのバッファポインタです。

buflen

buffer のバイトサイズです。

list

エンコードしたいイベント ID リストが格納されている領域のポインタです。

list には、イベント ID が count で指定されただけ格納されています。

イベント ID の数、count=0 の場合は、全イベント ID が対象になります。

count

list に格納されているイベント ID の数です。

ceed

有効(Enabled) / 無効(Disabled)を 1 (有効) 0 (無効) で指定します。

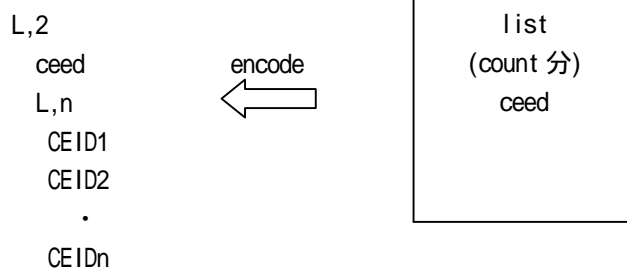
(3) 戻り値

戻り値	意味
0	正常にエンコードできた。
(-1)	msg を正しくエンコードできなかった。

(4) 説明

list に格納されているイベント ID を、S2F37 の SECS メッセージにエンコードします。  
buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。

msg S2F37



### 3.6.3.13 DshDecodeS2F37() S2F37 のイベント有効/無効情報のデコード

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshDecodeS2F37(
    DSHMSG *msg, // S2F37 が格納されている SECS メッセージ 情報構造体のポインタ
    TCEID *list // デコードしたイベント ID 定義情報リスト格納用領域のポインタ
    int *count, // メッセージに含まれていたイベント ID 数の格納用
    int *ceed // 有効/無効(1/0)格納用
);
```

##### [.NET VB]

```
Function DshDecodeS2F37 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef list As Int32,
    ByRef count As Int32,
    ByRef ceed As Int32) As Int32
```

##### [.NET C#]

```
int DshDecodeS2F37(
    ref DSHMSG msg,
    ref uint list,
    ref int count,
    ref int ceed );
```

#### (2) 引数

msg

デコードした S2F37 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。

list

デコードしたイベント ID リストの結果を格納するための構造体のポインタです。

count

S2F37 メッセージに含まれるイベント ID の数を格納します。

ceed

S2F37 メッセージで指定される CEED を格納します。

#### (3) 戻り値

戻り値	意味
0	正常にデコードできた。
(-1)	msg を正しくデコードできなかった。

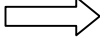
#### (4) 説明

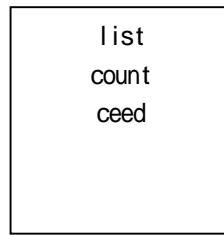
S2F37 の SECS メッセージをデコードし、含まれるイベント ID を list で指定される領域に、イベント ID の数を count に、そして CEED (有効/無効) を ceed に格納します。

list 領域のメモリは S2F37 に含まれる可能性のある最大限の数分を準備してください。

smg S2F37

L,2  
ceed  
L,n  
CEID1  
CEID2  
.  
CEIDn

decode  






### 3.6.4 ユーザ作成ライブラリ関数

#### 3.6.4.1 DshResponseS6F12() S6F12 イベントレポート送信応答メッセージ

##### (1) 呼出書式

###### [c, C++]

```
API int APIX DshResponseS6F12(
    int eqid, // 通信対象装置 ID(0,1,2,...)
    ID_TR trid, // 通信ドライバーから渡されるトランザクション ID
    TFPP_INFO * info, // プロセッサプログラム情報格納ポインタ
    int ackc6 // ACKC6 です。
);
```

###### [.NET VB]

```
Function DshResponseS6F12 (
    ByVal eqid As Int32,
    ByVal trid As Int32,
    ByRef info As dsh_info.TCE_CONTENT,
    ByVal ackc6 As Int32) As Int32
```

###### [.NET C#]

##### (2) 引数

eqid

GEM 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

通信ドライバーから渡されるトランザクション ID です。応答メッセージの送信時に必要になります。受信メッセージポーリング時に GemGetSecsMsg() 関数によって渡されたものです。

info

イベントレポート情報が格納されている構造体のポインタです。

ackc6

S6F12 メッセージに設定する ACKC6 です。

##### (3) 戻り値

戻り値	意味
0	常に 0 が戻ります。

##### (4) 説明

DSHGEM-LIB から渡された S6F11 メッセージの処理を受け付けた関数は、まず、メッセージをデコードしてイベントレポート情報を構造体に取り出します。

そして、処理が終了した後、本関数を呼び出して処理結果を S6F12 として装置に送信します。

本関数は、trid と ackc6 に情報に基づき、DshMakeS6F11Response() 関数を使って S6F12 メッセージを作成し、それを装置に送信します。

**本関数は user 作成ライブラリに属するものです。( default\_u\_s6f11.c )**

VB, C# では、dsh\_ulib クラスに属します。

### 3.6.4.2 DshResponseS6F14() S6F14 注釈付きイベントレポート送信応答メッセージ

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshResponseS6F14(
    int eqid,                // 通信対象装置 ID(0,14,...)
    ID_TR trid,              // 通信ドライバーから渡されるトランザクション ID
    TFPP_INFO * info,        // プロセッサ情報格納ポインタ
    int ackc6                 // ACKC6 です。
);
```

##### [.NET VB]

```
Function DshResponseS6F14 (
    ByVal eqid As Int32,
    ByVal trid As Int32,
    ByRef info As dsh_info.TCE_CONTENT,
    ByVal ackc6 As Int32) As Int32
```

##### [.NET C#]

```
int DshResponseS6F14(
    int eqid,
    uint trid,
    ref TCE_CONTENT info,
    int ackc6 );
```

#### (2) 引数

eqid

GEM 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

通信ドライバーから渡されるトランザクション ID です。応答メッセージの送信時に必要になります。受信メッセージポーリング時に GemGetSecsMsg() 関数によって渡されたものです。

info

注釈付きイベントレポート情報が格納されている構造体のポインタです。

ackc6

S6F14 メッセージに設定する ACKC6 です。

#### (3) 戻り値

戻り値	意味
0	常に 0 が戻ります。

#### (4) 説明

DSHGEM-LIB から渡された S6F13 メッセージの処理を受け付けた関数は、まず、メッセージをデコードして注釈付きイベントレポート情報を構造体に取り出します。

そして、処理が終了した後、本関数を呼び出して処理結果を S6F14 として装置に返信送信します。

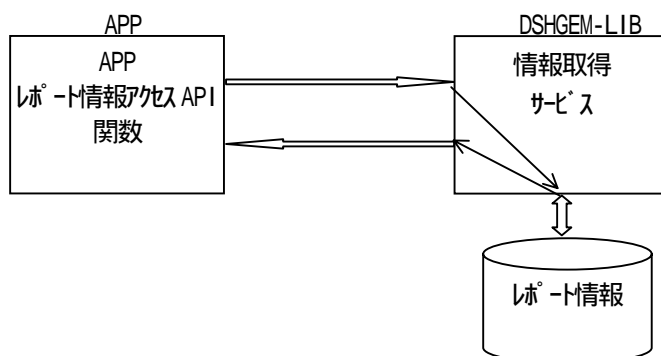
本関数は、trid と ackc6 に情報に基づき、DshMakeS6F13Response() 関数を使って S6F14 メッセージを作成し、それを装置に送信します。

**本関数は user 作成ライブラリに属するものです。( default u\_s6f13.c )**

VB, C# では、dsh\_ulib クラスに属します。

### 3.7 Report レポート情報アクセス関数

レポート情報は収集イベントにリンクされる情報であり、DSHGEM-LIB が管理します。従って、APP はこれらの情報をアクセスするために以下の DSHGEM-LIB API 関数を使用します。



#### (1) 情報アクセスと送信 API 関数

レポート情報のアクセスと装置へのメッセージ送信に関連するサービスのための API 関数名は次の一覧表のとおりです。

	API 関数名	機能
1	GemGetRpName()	指定された REPORT ID のレポート名を取得します。
2	GemGetRpVCount()	指定された REPORT ID の Enable 状態を取得します。
3	GemGetRpVid()	指定された REPORT ID にリンクされている変数 ID を取得します。
4	GemGetRpAllVid()	指定された REPORT ID にリンクされている全変数 ID を取得します。
5	GemGetRpList()	システムに登録されているレポート ID 名の一覧表を取得します。
6	GemSendS2F33()	S2F33 レポート情報を送信します。(レポート ID と、それにリンクされている VID 情報)
7	GemSendS6F19()	S6F19 個別レポート要求メッセージを送信します。

#### (2) レポート関連ライブラリ関数

APP が使用できるレポート情報処理関連ライブラリ関数として、以下の関数があります。

	API 関数名	機能
1	DshInitTRP_LIST()	TRP_LIST 構造体の初期設定をします。
2	DshInitTRP_LINK()	TRP_LINK 構造体の初期設定をします。
3	DshAddTRP_LINK()	TRP_LINK 構造体にリンク変数 ID を追加します。)
4	DshFreeTRP_LIST()	TRP_LIST 構造体内に使用されているメモリを開放します。(3.5.2 で説明します。)
5	DshEncodeS2F33()	TRP_LIST 構造体内のレポート ID リストから S2F33 メッセージをエンコードします。
6	DshDecodeS2F33()	S2F33 メッセージをデコードしレポート ID リストに設定します。
7	DshDecodeS6F20()	S6F20 に含まれる変数情報を TRP_CONTENT 構造体内にデコードします。

#### 3.7.1 使用する構造体

先に説明した 3.4.1 CE 収集イベント情報アクセス関数が使用する構造体の項を参照ください。

## 3.7.2 レポート情報アクセスと通知関数

### 3.7.2.1 GemGetRpName() レポート名取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetRpName(
    int eqid,           // 通信対象装置 ID(0,1,2,...)
    TRPID rpid,        // RPID
    char *name          // 取得した名前の格納領域o イタ
);
```

##### [.NET VB]

```
Function GemGetRpName (
    ByVal eqid As Int32,
    ByVal rpid As Int32,
    ByVal name As String) As Int32
```

##### [.NET C#]

```
int GemGetRpName(
    int eqid,
    uint rpid,
    byte[] name );
```

#### (2) 引数

eqid

GEM 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。

rpid

レポート ID です。

装置管理情報定義ファイルに登録されているレポートの ID でなければなりません。

name

取得した名前 (文字列) を格納する領域のポインタです。名前格納に十分な領域を準備してください。

bytesize

取得された名前の文字列長です。

#### (3) 戻り値

戻り値	意味
>= 0	正常に取得できた。
(-1)	rpid が正しくなかった。

#### (4) 説明

rpid で指定された RP (レポート) の名前を取得します。

正常に取得できた場合は、関数の戻り値は 取得できた名前のバイト長 になります。

rpid が未定義の場合は (-1) が戻ります。

### 3.7.2.2 GemGetRpVCount() レポートにリンクされている変数の数の取得関数

(1) 呼出書式

**[C, C++]**

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

**[.NET VB]**

```
Function GemGetRpVCount (
    ByVal eqid As Int32,
    ByVal rpid As Int32) As Int32
```

**[.NET C#]**

```
int GemGetRpVCount(
    int eqid,
    uint rpid );
```

(2) 引数

eqid

GEM 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。

rpid

レポート ID です。

装置管理情報定義ファイルに登録されているレポートの ID でなければなりません。

(3) 戻り値

戻り値	意味
>= 0	リンクされているデータ変数の数。
(-1)	rpid の値が正しくなかった。

(4) 説明

rpid で指定された RP(レポート)にリンクされている変数 ID の数を取得します。

=0 はリンクされている変数がないことを意味します。

### 3.7.2.3 GemGetRpVid() レポートにリンクされている指定順位の変数 ID 取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetRpVid(
    int eqid,           // 通信対象装置 ID(0,1,2,...)
    TRPID rpid,        // RPID
    int order,         // 順位(0,1,2.. )
    TRPID *vid         // 取得した変数 ID 格納ポインタ
);
```

##### [.NET VB]

```
Function GemGetRpVid (
    ByVal eqid As Int32,
    ByVal rpid As Int32,
    ByVal order As Int32,
    ByRef vid As Int32) As Int32
```

##### [.NET C#]

```
int GemGetRpVid(
    int eqid,
    uint rpid,
    int order,
    ref uint vid );
```

#### (2) 引数

eqid

GEM 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。

rpid

レポート ID です。

装置管理情報定義ファイルに登録されているレポートの ID でなければなりません。

order

何番目のレポート ID を取得するかを指定します。先頭の指定は=0 です。

vid

取得した変数 ID(VID)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)	rpid の値が正しくなかった。または order 番目のレポートは無かった。

#### (4) 説明

rpid で指定された RP(レポート)にリンクされている order 番目の変数 ID を取得します。

### 3.7.2.4 GemGetRpAllVid - RP 最小(Min)値取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetRpAllVid(
    int eqid,           // 通信対象装置 ID(0,1,2,...)
    TRPID rpid,        // RPID
    TRPID *vid          // リンクされている変数 vid を格納する領域のポインタ
);
```

##### [.NET VB]

```
Function GemGetRpAllVid (
    ByVal eqid As Int32,
    ByVal rpid As Int32,
    ByRef vid As Int32) As Int32
```

##### [.NET C#]

```
int GemGetRpAllVid(
    int eqid,
    uint rpid,
    ref uint vid );
```

#### (2) 引数

eqid

GEM 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。

rpid

レポート ID です。

装置管理情報定義ファイルに登録されているレポートの ID でなければなりません。

vid

リンクされている全変数 ID を格納するための領域ポインタです。

リンクされている可能性のある最大数分の領域を準備してください。

#### (3) 戻り値

戻り値	意味
>=0	取得できた変数 ID 数
(-1)	rpid の値が正しくなかった。

#### (4) 説明

rpid で指定された RP(レポート)にリンクされている全変数 ID を取得します。

戻り値 = 0 の場合、リンクされている変数がないことを意味します。

戻り値 > 0 の場合、戻り値がリンクされている変数の数で、それらの ID が vid の領域に設定されます。

### 3.7.2.5 GemGetRpVName() レポートにリンクされている指定順位の変数名取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetRpVName(
    int eqid,           // 通信対象装置 ID(0,1,2,...)
    TRPID rpid,        // RPID
    int order,         // 順位(0,1,2.. )
    char *vname        // 取得した変数名格納ポインタ
);
```

##### [.NET VB]

```
Function GemGetRpVName (
    ByVal eqid As Int32,
    ByVal rpid As Int32,
    ByVal order As Int32,
    ByVal vname As String) As Int32
```

##### [.NET C#]

```
int GemGetRpVName(
    int eqid,
    uint rpid,
    int order,
    byte[] vname );
```

#### (2) 引数

eqid

GEM 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。

rpid

レポート ID です。

装置管理情報定義ファイルに登録されているレポートの ID でなければなりません。

order

何番目の変数の名前を取得するかを指定します。先頭の指定は=0 です。

vname

取得した変数名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)	rpid の値が正しくなかった。または order 番目の変数は無かった。

#### (4) 説明

rpid で指定された RP(レポート)にリンクされている order 番目の変数の名前を取得します。



### 3.7.2.6 Gem SetRpVLink () レポートの変数リンクの設定関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemSetRpVLink (
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TRP_LIST *list          // リンク作成したい情報が格納されているリスト
);
```

##### [.NET VB]

```
Function GemSetRpVLink (
    ByVal eqid As Int32,
    ByRef list As dsh_info.TRP_LIST) As Int32
```

##### [.NET C#]

```
int GemSetRpVLink(
    int eqid,
    ref TRP_LIST list );
```

#### (2) 引数

eqid

GEM 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

リンク対象の RPID とリンクしたい変数 ID が含まれているリストのポインタです。

#### (3) 戻り値

戻り値	意味
= 0	正常に取得できた。
(-1)	rpидの値が正しくなかった。または変数 ID は無かった。

#### (4) 説明

list に含まれる 1 個以上の RP リンク情報に従って RP に対する変数のリンクを設定しなおします。

list には、リストに含まれる TRP\_LINK 構造体の数とポインタリストが含まれます。

TRP\_LINK 構造体には、1 個の RPID と、それにリンクされる変数の数と変数 ID のリストが含まれます。

### 3.7.2.7 GemGetRpList() 全登録レポート ID 取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetRpList(
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TBIN_DLIST **list       // 取得リスト格納ポインタの格納ポインタ
);
```

##### [.NET VB]

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

##### [.NET C#]

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

#### (2) 引数

eqid

GEM 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 RPID が格納されている TBIN\_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全レポート ID とその名前を TBIN\_DLIST 構造体に取り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レポート ID 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、DshFreeTBIN\_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。

TBIN\_DLIST 構造体は次のとおりです。

```
typedef struct{
    int          count;                // 取得できた ID 数
    ULONG       *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
}TBIN_DLIST;
```

### 3.7.2.8 GemSendS2F33() レポート設定送信

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemSendS2F33(
    int      eqid,                // 通信対象装置 ID(0,1,2,...)
    TRP_LIST *list,              // rpid リンク情報リスト格納構造体のポインタ
    int      *drack,             // S2F34 の DRACK の値格納用ポインタ
    int      (WINAPI *callback)(), // 実行終了時のコールバック関数
    ULONG upara                  // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function GemSendS2F33 (
    ByVal eqid As Int32,
    ByRef list As dsh_info.TRP_LIST,
    ByRef drack As Int32,
    ByVal callback As vcallback.callback_S2F33,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int GemSendS2F33(
    int eqid,
    ref TRP_LIST list,
    ref int drack,
    CallbackAck callback,
    uint upara );
```

#### (2) 引数

eqid

GEM 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

レポート ID にリンクする変数 ID リストが格納されている構造体のポインタです。

drack

応答メッセージ S2F34 に含まれる DRACK を格納する領域のポインタです。

callback

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

この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) ブロックド : 正常に送信できた。 drack に受信した応答 ack が格納されています。 (2) 非ブロックド : 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。(応答メッセージが得られなかった)

#### (4) 説明

装置に対し、list 内に設定された RPID(レポート ID)とそれにリンクされる VID(変数 ID)情報を S2F33 メッセージを使って送信します。

list 内には1個以上の RPID とそれらにリンクされる VID が格納されています。

RPID にリンクされる VID の数は0個以上です。

正常に応答メッセージを受信できた場合は、それに含まれる DRACK の値を drack 領域に返却します。

TRP\_LIST 構造体 list に CEID とそれにリンクされる RPID の設定するために以下のライブラリ関数を使用することができます。

```
DshInitTRP_LIST()
DshInitTRP_LINK()
DshAddTRP_LINK()
```

送信要求から S2F33 応答メッセージ受信までの制御は引数の S2F33 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S2F33 送信後、応答メッセージ S2F34 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、drack に S2F34 の DRACK の値が渡されます。
あり	送信要求後、S2F33 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常であった場合、drack に S2F34 の DRACK の値が渡されます。 エラーが検出された場合、(-1)が end_status にセットされます。

通信終了後、必要に応じて DshFreeTRP\_LIST()関数を使って、TRP\_LIST 内に使用されたメモリを開放してください。

```
DshFreeTRP_LIST(list);
```

#### (5) コールバック関数

[C,C++]

```
API int APIX callback(
    int eqid,                // 装置 ID
    int end_status,         // 実行結果
    int *drack,             // DRACK 値が格納されているポインタ
    ULONG upara             // 呼出時に指定したパラメータ
);
```

[.NET VB]

```
Function callback_S2F33(ByVal eqid As Integer, ByVal end_status As Integer, ByRef drack As Integer,
```

ByVal upara As Integer) As Integer

[.NET C#]

```
int CallbackAck(int eqid, int end_status, int *drack, 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。
(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

end\_status = 0 のときのみ drack の内容が有効です。

### 3.7.2.9 GemSendS6F19() 個別レポート要求メッセージ送信

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemSendS6F19(
    int      eqid,                // 通信対象装置 ID(0,1,2,...)
    TRPID    rpid,                // レポート ID
    TRP_CONTENT *rinfo,          // S6F20 のレポートデータ情報格納用構造体ポインタ
    int      (WINAPI *callback)(), // 実行終了時のコールバック関数
    ULONG    upara                // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function GemSendS6F19 (
    ByVal eqid As Int32,
    ByVal rpid As Int32,
    ByRef info As dsh_info.TRP_CONTENT,
    ByVal callback As vcallback.callback_S6F19,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int GemSendS6F19(
    int eqid,
    uint rpid,
    ref TRP_CONTENT info,
    CallbackS6F19 callback,
    uint upara );
```

#### (2) 引数

eqid

GEM 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。

rpid

レポートデータを個別要求するレポート ID です。

rinfo

応答メッセージ S6F20 に含まれるレポートデータを格納する構造体のポインタです。

callback

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

この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) ブロックド : 正常に送信できた。 rinfo に受信した応答 ack が格納されています。 (2) 非ブロックド : 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。(応答メッセージが得られなかった)

#### (4) 説明

装置に対し、rpid で指定されたレポート ID の個別レポートデータを要求するための S6F19 を送信します。正常に応答メッセージ S6F20 を受信できた場合は、それに含まれるレポートデータ情報の値を rinfo 構造体領域に返却します。

送信要求から S6F19 応答メッセージ受信までの制御は引数の S6F19 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S6F19 送信後、応答メッセージ S6F20 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、rinfo に S6F20 のレポートデータ情報の値が渡されます。
あり	送信要求後、S6F19 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常であった場合、rinfo に S6F20 のレポートデータ情報の値が渡されます。 エラーが検出された場合、(-1)が end_status にセットされます。

通信終了後、必要に応じて DshFreeTRP\_CONTENT()関数を使って、rinfo 内に使用されたメモリを開放してください。

```
DshFreeTRP_CONTENT(rinfo);
```

#### (5) コールバック関数

##### [c,C++]

```
API int APIX callback(
    int eqid,                // 装置 ID
    int end_status,         // 実行結果
    TRP_CONTENT *rinfo,     // レポートデータ情報値が格納されているポインタ
    ULONG upara             // 呼出時に指定したパラメータ
);
```

##### [.NET VB]

```
Function callback_S6F19(ByVal eqid As Integer, ByVal end_status As Integer, ByRef info As dsh_info.TRP_CONTENT, ByVal upara As Integer) As Integer
```

##### [.NET C#]

```
int CallbaS6F19(int eqid, int end_status, ref TRP_CONTENT info, uint upara);
```

end\_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。

(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

end\_status = 0 のときのみ info の内容が有効です。



### 3.7.3 レポート関連ライブラリ関数

#### 3.7.3.1 DshInitTRP\_LIST レポートリンク情報リストの初期設定

##### (1) 呼出書式

###### [C, C++]

```
API int APIX DshInitTRP_LIST(
    TRP_LIST *list,           // TRP_LIST 構造体のポインタ
    int count                // コマンドコード情報リストのサイズ
);
```

###### [.NET VB]

```
Sub DshInitTRP_LIST (
    ByRef list As dsh_info.TRP_LIST,
    ByVal count As Int32)
```

###### [.NET C#]

```
void DshInitTRP_LIST(
    ref TRP_LIST list,
    int count );
```

##### (2) 引数

list

レポートリンク情報リスト構造体のポインタです。このメンバーを初期設定します。

count

TRP\_LIST に含まれる RPID の最大数です。

##### (3) 戻り値

なし。

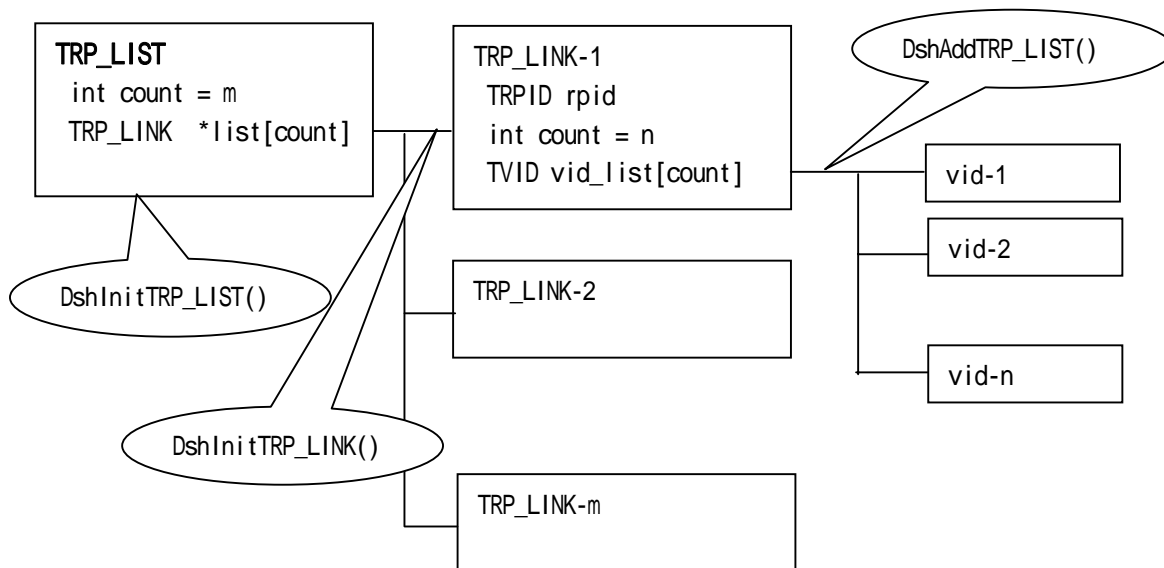
##### (4) 説明

本関数は APP が S2F33 を送信する際のレポートリンク情報のためのリスト構造体を初期設定するために使用します。

最初に list 内をクリアします。そして、list 内に count 分のレポート ID に対するリンク情報を設定するための領域を確保します。

また、1 個の RPID に対するレポート情報の設定は次の 2 つの関数を使ってください。

```
DshInitTRP_LINK()
DshAddTRP_LINK()
```



TRP\_LIST 構造体の使用後、DshFreeTRP\_LIST()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。

### 3.7.3.2 DshInitTRP\_LINK レポートリンク情報の初期設定

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshInitTRP_LINK(
    TRP_LINK *list,           // TRP_LINK 構造体のポインタ
    int rp_order,           // list 内の LIST 位置
    TRPID rpid,             // TRP_LINK 内に設定する RPID
    int v_count             // レポートリンク情報リストのサイズ
);
```

##### [.NET VB]

```
Function DshInitTRP_LINK (
    ByRef list As dsh_info.TRP_LIST,
    ByVal rp_order As Int32,
    ByVal rpid As Int32,
    ByVal rp_count As Int32) As Int32
```

##### [.NET C#]

```
int DshInitTRP_LINK(
    ref TRP_LIST list,
    int rp_order,
    uint rpid,
    int rp_count );
```

#### (2) 引数

list  
TRP\_LIST \*list, // TRP\_LIST 構造体のポインタ

rp\_order  
list 内のリスト位置を指定します。(0,1..)

rpid  
設定対象のレポート ID です。

v\_count  
list の rp\_order 番目の RPID にリンクされる VID の最大数です。  
(TRP\_LINK 構造体の count です。)

#### (3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	list 内の count の枠外の rp_order が指定された。

#### (4) 説明

本関数は APP が S2F33 を送信する際のレポートリンク情報リスト構造体内の rp\_order 番目のリストに位置する TRP\_LINK 構造体の初期設定を行うために使用します。

rp\_order 番目のリストに TRP\_LINK 領域用メモリを確保し、rpid と v\_count を設定します。

### 3.7.3.3 DshAddTRP\_LINK レポートリンク情報に変数 ID を追加する

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshAddRP_LINK(
    TRP_LINK *list,           // TRP_LINK 構造体のポインタ
    int rp_order,           // list 内の LIST 位置
    TVID vid                // TRP_LINK 内に加える VID
);
```

##### [.NET VB]

```
Function DshAddTRP_LINK (
    ByRef list As dsh_info.TRP_LIST,
    ByVal rp_order As Int32,
    ByVal vid As Int32) As Int32
```

##### [.NET C#]

```
int DshAddTRP_LINK(
    ref TRP_LIST list,
    int rp_order,
    uint vid );
```

#### (2) 引数

**list**  
TRP\_LIST \*list, // TRP\_LIST 構造体のポインタ

**rp\_order**  
list 内のリスト位置を指定します。(0,1..)

**vid**  
list 内の rp\_order 番目の TRP\_LINK 構造体内のリストに加える変数 ID です。

#### (3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	list 内の count の枠外の rp_order が指定された。 または、既に DshInitTRP_LINK() で設定された v_count 分のレポート ID が設定されてしまっている。

#### (4) 説明

本関数は APP が S2F33 を送信する際のイベントレポートリンク情報リスト構造体内の rp\_order 番目のリストに用意された TRP\_LINK 構造体内の rp\_list のリストに vid を加えます。  
既に、rp\_count 分のレポート ID が設定されていた場合には、(-1) を返却します。

### 3.7.3.4 DshFreeTRP\_LIST () レポート情報リスト構造体メモリの開放

#### (1) 呼出書式

##### [C, C++]

```
API void APIX DshFreeTRP_LIST(  
    TRP_LIST *list    // メモリを開放したいレポート情報リスト構造体のポインタ  
);
```

##### [.NET VB]

```
Sub DshFreeTRP_LIST (  
    ByRef list As dsh_info.TRP_LIST)
```

##### [.NET C#]

```
void DshFreeTRP_LIST(  
    ref TRP_LIST list );
```

#### (2) 引数

list

メモリを解放したいレポートリンク情報リスト構造体のポインタです。

#### (3) 戻り値

なし。

#### (4) 説明

TRP\_LIST 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、TRP\_LIST の内容を全て 0 で初期設定します。

list が NULL ならば、何も処理しません。

TRP\_LIST 構造体は RPngSendS2F33() API 関数の実行時に使用されるものです。

### 3.7.3.5 DshEncodeS2F33() レポート定義情報を S2F33 へエンコード

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshEncodeS2F33(
    DSHMSG *msg,           // SECS メッセージ 情報構造体のポインタ
    BYTE *buffer,         // S2F33 を格納するバッファポインタ
    int buflen,           // buffer のバイトサイズ
    TRP_LIST *list        // エンコードしたいレポート ID リスト格納構造体のポインタ
);
```

##### [.NET VB]

```
Function DshEncodeS2F33 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buflen As Int32,
    ByRef list As dsh_info.TRP_LIST) As Int32
```

##### [.NET C#]

```
int DshEncodeS2F33(
    ref DSHMSG msg,
    byte[] buff,
    int buflen,
    ref TRP_LIST list );
```

#### (2) 引数

msg

エンコードした S2F33 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 S2F33 のテキストを格納するためのバッファポインタです。

buflen

buffer のバイトサイズです。

list

エンコードしたいレポート ID リストが格納されている構造体のポインタです。

list には、レポート ID とそのレポート ID にリンクされている変数 ID が格納されています。

レポート ID の数が=0 の場合は、レポート定義が消去されることを意味し、また、レポート ID にリンクされる変数 ID の数が=0 の場合、そのレポート ID が消去されることを意味します。

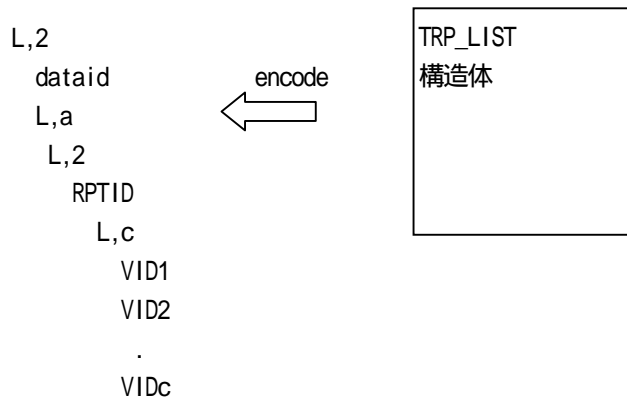
#### (3) 戻り値

戻り値	意味
0	正常にエンコードできた。
(-1)	msg に正しくエンコードできなかった。

#### (4) 説明

TRP\_LIST 構造体に格納されているレポート定義情報を、S2F33 の SECS メッセージにエンコードします。buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。

msg S2F33



### 3.7.3.6 DshDecodeS2F33() S2F33 をレポート定義情報へデコード

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshDecodeS2F33(
    DSHMSG *msg, // S2F33 が格納されている SECS メッセージ 情報構造体のポインタ
    TRP_LIST *list // デコードしたレポート ID 定義情報リスト格納構造体のポインタ
);
```

##### [.NET VB]

```
Function DshDecodeS2F33 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef list As dsh_info.TRP_LIST) As Int32
```

##### [.NET C#]

```
int DshDecodeS2F33(
    ref DSHMSG msg,
    ref TRP_LIST list );
```

#### (2) 引数

msg

S2F33 が格納されているメッセージ情報構造体のポインタです。

list

デコードしたレポート ID リストの結果を格納するための構造体のポインタです。

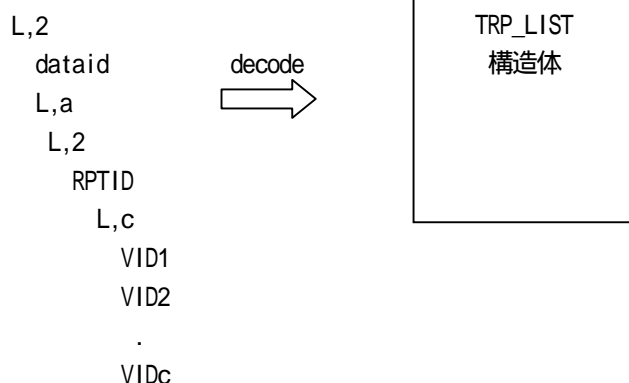
#### (3) 戻り値

戻り値	意味
0	正常にデコードできた。
(-1)	msg を正しくデコードできなかった。

#### (4) 説明

S2F33 の SECS メッセージに含まれるレポート定義情報を list で指定される TRP\_LIST 構造体にデコードし格納します。

msg S2F33



得られた list 情報の処理が終わった後は、DshFreeTRP\_LIST()関数で list 内部に割付使用されたメモリを開放することができます。



### 3.7.3.7 DshDecodeS6F20() S6F20 をレポートデータ情報ヘデコード

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshDecodeS6F20(
    int eqid, // 通信対象装置 ID(0,1,2,...)
    DSHMSG *msg, // S6F20 が格納されている SECS メッセージ 情報構造体のポインタ
    TRPID rpid, // レポート ID
    TRP_CONTENT *TRP_CONTENT // デコードしたレポート ID 定義情報リスト格納構造体のポインタ
);
```

##### [.NET VB]

```
Function DshDecodeS6F20 (
    ByVal eqid As Int32,
    ByRef msg As dshdr2.DSHMSG,
    ByVal rpid As Int32,
    ByRef info As dsh_info.TRP_CONTENT) As Int32
```

##### [.NET C#]

```
int DshDecodeS6F20(
    int eqid,
    ref DSHMSG msg,
    uint rpid,
    ref TRP_CONTENT info );
```

#### (2) 引数

eqid

GEM 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。

msg

デコードした S6F20 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。

rpid

レポートデータを個別要求したレポート ID です。

rinfo

デコードしたレポートデータ情報の結果を格納するための構造体のポインタです。

#### (3) 戻り値

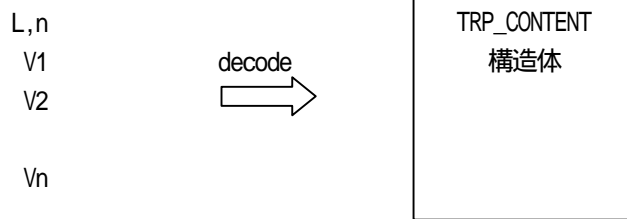
戻り値	意味
0	正常にデコードできた。
(-1)	msg を正しくデコードできなかった。

#### (4) 説明

S6F20 の SECS メッセージに含まれるレポートデータ情報を rinfo で指定される TRP\_CONTENT 構造体にデコードし格納します。

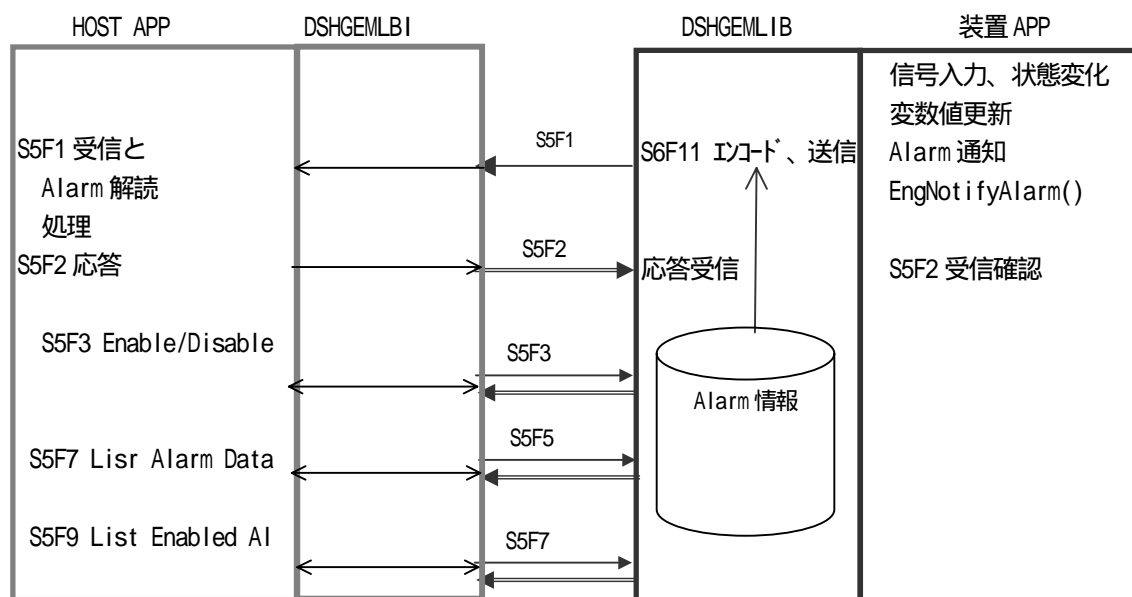
引数の中に、装置 ID がありますが、レポート ID とそれにリンクされている変数が装置によって異なるためです。また rpid を指定するのは、S6F20 にレポート ID が含まれていないためです。

msg S6F20



得られた rinfo 情報の処理が終わった後は、DshFreeTRP\_CONTENT()関数で rinfo 内部に割付使用されたメモリを開放することができます。

### 3.8 Alarm アラーム情報アクセスと通知関数



#### (1) 情報アクセスと送信 API 関数

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

	API 関数名	機能
1	GemGetAlName()	指定された ALID の収集イベント名を取得します。
2	GemGetAlEnabled()	指定された ALID の Enable 状態を取得します。
3	GemSetAlEnabled()	指定された ALID の Enable 状態を設定します。
4	GemGetAlcd()	指定された ALID に設定されている ALCD を取得します。
5	GemGetAltx()	指定された ALID に設定されている ALTX を取得します。
6	GemGetAlCeOn()	指定された ALID のアラーム発生時にリンクされている CEID を取得します。
7	GemGetAlCeOff()	指定された ALID のアラーム復旧時にリンクされている CEID を取得します。
8	GemSendS5F3()	S5F3 アラーム有効/無効設定メッセージを送信します。
9	GemSendS5F5()	S5F5 アラームリスト要求メッセージを送信します。
10	GemNotifyAlarm()	指定された ALID のアラーム通知(S5F1)をします。

#### (2) アラーム関連ライブラリ関数

	API 関数名	機能
1	DshDecodeS5F1()	S5F1 メッセージをデコードしアラーム情報を取得します。
2	DshResponseS5F2()	S5F2 応答メッセージを送信します。(ユーザライブラリ u_s5f1.c)
3	DshMakeS5F1Response()	S5F2 応答メッセージを作ります。
4	DshFreeTAL_S5F1_INFO()	アラーム情報 TAL_S5F1_INFO 構造体内のメモリを開放します。
5	DshDecodeS5F6()	S5F6 メッセージをデコードしアラームリストを取得します。
6	DshFreeTAL_S5F6_LIST()	アラーム情報 TAL_S5F6_LIST 構造体内のメモリを開放します。

#### (3) ユーザ作成ライブラリ関数

	ライブラリ関数名	機能
1	DshResponseS5F2()	S5F1 アラーム報告送信応答

### 3.8.1 使用する構造体

(1) S5F1 アラーム報告メッセージのデコード結果を格納する構造体です。

```
typedef struct{
    int      on_off;          // 有効/無効(1/0)
    TALID    alid;           // ALID
    TALCD    alcd;           // ALCD
    char     *al tx;         // ALTX
} TAL_S5F1_INFO;
```

(2) S5F6 メッセージのデコード結果を格納する構造体です。

```
typedef struct{
    int      count;          // 含まれるアラーム ID 数
    TAL_S5F1_INFO **al_list; // アラーム情報のリスト
} TAL_S5F6_LIST;
```

## 3.8.2 Alarm アラーム情報アクセス関数

### 3.8.2.1 GemGetAIName() アラーム名取得関数

#### (1) 呼出書式

##### A[C,C++]

```

PI int APIX GemGetAIName(
    int eqid,           // 通信対象装置 ID(0,1,2,...)
    TALID alid,        // ALID
    char *name         // 取得した名前の格納領域o イタ
);

```

##### [.NET VB]

```

Function GemGetAIName (
    ByVal eqid As Int32,
    ByVal alid As Int32,
    ByVal name As String) As Int32

```

##### [.NET C#]

```

int GemGetAIName(
    int eqid,
    uint alid,
    byte[] name );

```

#### (2) 引数

eqid

GEM 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。

alid

アラーム ID です。

装置管理情報定義ファイルに登録されているアラームの ID でなければなりません。

name

取得した名前（文字列）を格納する領域のポインタです。名前格納に十分な領域を準備してください。

#### (3) 戻り値

戻り値	意味
>= 0	正常に取得できた。
(-1)	alid が正しくなかった。

#### (4) 説明

alid で指定された AL(アラーム)の名前を取得します。

正常に取得できた場合は、関数の戻り値は 得られた名前のバイト長 になります。

alid が未定義の場合は(-1)が戻ります。

### 3.8.2.2 GemGetAIEnabled() アラーム有効状態取得関数

#### (1) 呼出書式

##### [C, C++]

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

##### [.NET VB]

```
Function GemGetAIEnabled (
    ByVal eqid As Int32,
    ByVal alid As Int32) As Int32
```

##### [.NET C#]

```
int GemGetAIEnabled(
    int eqid,
    uint alid );
```

#### (2) 引数

eqid

GEM 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。

alid

アラーム ID です。

装置管理情報定義ファイルに登録されているアラームの ID でなければなりません。

#### (3) 戻り値

戻り値	意味
0	有効状態 0=無効、1=有効
(-1)	alid の値が正しくなかった。

#### (4) 説明

alid で指定された AL(アラーム)の有効状態を取得します。

戻り値が =1 の場合、有効状態です。装置がアラーム通知できる状態です。

戻り値が =0 の場合、無効状態です。装置がアラーム通知できない状態です。

### 3.8.2.3 GemSetAlEnabled() アラーム有効状態設定関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemStAlnabled(
    int eqid,           // 通信対象装置 ID(0,1,2,...)
    TALID alid,        // ALID
    int on_off         // 有効 On/Off
);
```

##### [.NET VB]

```
Function GemSetAlEnabled (
    ByVal eqid As Int32,
    ByVal alid As Int32,
    ByVal on_off As Int32) As Int32
```

##### [.NET C#]

```
int GemSetAlEnabled(
    int eqid,
    uint alid,
    int on_off );
```

#### (2) 引数

eqid

GEM 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。

alid

アラーム ID です。

装置管理情報定義ファイルに登録されているアラームの ID でなければなりません。

on\_off

有効 / 無効の設定したい状態を指定します。

=1 の場合、有効状態にします。GemNotifyAlarm() によってアラーム通知できる状態です。

=0 の場合、無効状態にします。GemNotifyAlarm() によってアラーム通知できない状態です。

#### (3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	alid の値が正しくなかった。

#### (4) 説明

alid で指定された AL (アラーム) の有効状態を on\_off 値で設定します。

ここで設定した状態を GemSendS5F3() API 関数を使って装置に S5F3 メッセージで通知することができます。

### 3.8.2.4 GemGetAlcd() アラームのALCD取得関数

#### (1) 呼出書式

##### [C, C++]

```
API TALCD APIX GemGetAlcd(
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TALID alid               // ALID
);
```

##### [.NET VB]

```
Function GemGetAlcd (
    ByVal eqid As Int32,
    ByVal alid As Int32) As Int32
```

##### [.NET C#]

```
byte GemGetAlcd(
    int eqid,
    uint alid );
```

#### (2) 引数

eqid

GEM 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。

alid

アラーム ID です。

装置管理情報定義ファイルに登録されているアラームの ID でなければなりません。

#### (3) 戻り値

戻り値	意味
>= 0	指定された alid が有する ALCD アラームコード
(-1)	alid の値が正しくなかった。

#### (4) 説明

alid で指定された AL(アラーム)のアラームコード(ALCD)を取得します。



### 3.8.2.5 GemGetAltx() アラームのALTX取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetAltx(
    int eqid,           // 通信対象装置 ID(0,1,2,...)
    TALID alid         // ALID
    char *altx
);
```

##### [.NET VB]

```
Function GemGetAltx (
    ByVal eqid As Int32,
    ByVal alid As Int32,
    ByVal altx As String) As Int32
```

##### [.NET C#]

```
int GemGetAltx(
    int eqid,
    uint alid,
    byte[] altx );
```

#### (2) 引数

eqid

GEM 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。

alid

アラーム ID です。

装置管理情報定義ファイルに登録されているアラームの ID でなければなりません。

al tx

取得したアラームテキストを格納するバッファのポインタです。

41 バイト以上のサイズのバッファが必要です。

#### (3) 戻り値

戻り値	意味
= 0	正常に取得できた。
(-1)	alid の値が正しくなかった。

#### (4) 説明

alid で指定された AL(アラーム)のアラームテキスト(ALTX)を al tx のバッファに取得します。

### 3.8.2.6 GemGetAlCeOn() アラーム発生時のリンクイベント ID 取得関数

#### (1) 呼出書式

##### [C, C++]

```
API TCEID APIX GemGetAlCeOn(
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TALID alid               // ALID
);
```

##### [.NET VB]

```
Function GemGetAlCeOn (
    ByVal eqid As Int32,
    ByVal alid As Int32) As Int32
```

##### [.NET C#]

```
uint GemGetAlCeOn(
    int eqid,
    uint alid );
```

#### (2) 引数

eqid

GEM 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。

alid

アラーム ID です。

装置管理情報定義ファイルに登録されているアラームの ID でなければなりません。

#### (3) 戻り値

戻り値	意味
>= 0	取得したアラーム発生時のイベント ID
(-1)	alid の値が正しくなかった。またはリンクイベントは無かった。

#### (4) 説明

alid で指定された AL (アラーム) が発生したときに通知するためにリンクされているイベント ID (CEID) を取得します。

アラーム発生時のリンクイベントがあれば、戻り値に取得したイベント ID が返却されます。

リンクされていない場合は (-1) が返却されます。

### 3.8.2.7 GemGetAlCeOff() アラーム復旧時のリンクイベント ID 取得関数

#### (1) 呼出書式

##### [C, C++]

```
API TCEID APIX GemGetAlCeOff(
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TALID alid               // ALID
);
```

##### [.NET VB]

```
Function GemGetAlCeOff (
    ByVal eqid As Int32,
    ByVal alid As Int32) As Int32
```

##### [.NET C#]

```
uint GemGetAlCeOff(
    int eqid,
    uint alid );
```

#### (2) 引数

eqid

GEM 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。

alid

アラーム ID です。

装置管理情報定義ファイルに登録されているアラームの ID でなければなりません。

#### (3) 戻り値

戻り値	意味
>= 0	取得したアラーム復旧時のイベント ID
(-1)	alid の値が正しくなかった。またはリンクイベントは無かった。

#### (4) 説明

alid で指定された AL (アラーム) が復旧したときに通知するためにリンクされているイベント ID (CEID) を取得します。

アラーム復旧時のリンクイベントがあれば、戻り値に取得したイベント ID が返却されます。

リンクされていない場合は (-1) が返却されます。

### 3.8.2.8 GemGetAllList() 全登録アラーム ID 取得関数

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemGetAllList(
    int eqid,                // 通信対象装置 ID(0,1,2,...)
    TBIN_DLIST **list       // 取得リスト格納ポインタの格納ポインタ
);
```

##### [.NET VB]

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

##### [.NET C#]

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

#### (2) 引数

eqid

GEM 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 ALID が格納されている TBIN\_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全アラーム ID とその名前を TBIN\_DLIST 構造体に取り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アラーム ID 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、DshFreeTBIN\_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。

TBIN\_DLIST 構造体は次のとおりです。

```
typedef struct{
    int          count;                // 取得できた ID 数
    ULONG       *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
}TBIN_DLIST;
```

### 3.8.2.9 GemSendS5F3() 有効/無効アラーム送信

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemSendS5F3(
    int      eqid,           // 通信対象装置 ID(0,1,2,...)
    TALID    alid,           // 有効/無効対象 ALID
    int      aled,           // 有効/無効(1/0)の指定
    int      all_flag,       // 対象が全 ALID かどうかを指定(0=指定 alid、その他=All)
    int      *ackc5,         // S5F4 の ACKC5 格納領域のポインタ
    int      (WINAPI *callback)(), // 実行終了時のコールバック関数
    ULONG    upara           // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function GemSendS5F3 (
    ByVal eqid As Int32,
    ByVal alid As Int32,
    ByVal aled As Int32,
    ByVal all_flag As Int32,
    ByRef ackc5 As Int32,
    ByVal callback As vcallback.callback_S5F3,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int GemSendS5F3(
    int eqid,
    uint alid,
    int aled,
    int all_flag,
    ref int ackc5,
    CallbackAck callback,
    uint upara );
```

#### (2) 引数

eqid

GEM 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。

alid

有効/無効対象 ALID です。

aled

有効/無効の指定をします。1 が有効、0 が無効になります。

all\_flag

有効/無効の設定が全 ALID 対象にするかどうかを指定します。

all\_flag=0 ならば alid に指定された ALID だけが対象になり、その他の値の場合は、全 ALID が対象になります。

ackc5

応答メッセージ S5F4 に含まれる ACKC5 を格納する領域のポインタです。

callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。

ユーザは任意の関数名を指定できます。  
この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) ブロックモード：正常に送信できた。 ackc5 に受信した応答 ACKC5 が格納されています。 (2) 非ブロックモード：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。(応答メッセージが得られなかった)

### (4) 説明

alid で指定された ALID(収集イベント ID)に対し alid で指定された有効/無効を S5F3 メッセージを使って装置に送信します。

all\_flag の値が 0 以外( !=0 )の場合は全 ALID に対する有効/無効の設定になります。

all\_flag = 0 ならば、alid で指定された ALID 個別の設定になります。

正常に応答メッセージを受信できた場合は、それに含まれる ACKC5 の値を ackc5 領域に返却します。

送信要求から S5F3 応答メッセージ受信までの制御は引数の S5F3 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S5F3 送信後、応答メッセージ S5F4 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、ackc5 に S5F4 の ACKC5 の値が渡されます。
あり	送信要求後、S5F3 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常であった場合、ackc5 に S5F4 の ACKC5 の値が渡されます。 エラーが検出された場合、(-1)が end_status にセットされます。

### (5) コールバック関数

#### [C, C++]

```
API int APIX callback(
    int eqid,                // 装置 ID
    int end_status,          // 実行結果
    int *ackc5,              // ACKC5 値が格納されているポインタ
    ULONG upara              // 呼出時に指定したパラメータ
);
```

#### [.NET VB]

```
Function callback_S5F3(ByVal eqid As Integer, ByVal end_status As Integer, ByRef ackc5 As Integer,
    ByVal upara As Integer) As Integer
```

#### [.NET C#]

```
int CallbackAck(int eqid, int end_status, int *ackc5, 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。
(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

end\_status = 0 のときのみ ackc5 の内容が有効です。

### 3.8.2.10 GemSendS5F5() アラームリスト要求メッセージ送信

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemSendS5F5(
    int eqid, // 通信対象装置 ID(0,1,2,...)
    TALID *alid_list, // 要求する ECID のリスト領域のポインタ
    int count, // alid_list に含まれる ECID の数
    TAL_S5F6_LIST *al_list, // S5F6 で返されたアラームリスト情報格納用リストポインタ
    int (WINAPI *callback)(), // 実行終了時のコールバック関数
    ULONG upara // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function GemSendS5F5 (
    ByVal eqid As Int32,
    ByRef al_list As Int32,
    ByVal count As Int32,
    ByRef list As dsh_info.TAL_S5F6_LIST,
    ByVal callback As vcallback.callback_S5F5,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int GemSendS5F5(
    int eqid,
    ref uint al_list,
    int count,
    ref TAL_S5F6_LIST list,
    CallbackS5F5 callback,
    uint upara );
```

#### (2) 引数

eqid

GEM 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。

alid\_list

装置に要求したいアラームリストのアラーム ID が格納されているリストのポインタです。

count

alid\_list 内に含まれる装置アラーム ID の数です。

count=0 の場合は、全アラームが対象になります。

al\_list

alid\_list 内に指定されたアラーム ID のアラームリスト情報を格納するためのリストへのポインタです。

アラームリスト情報には ALCD (アラームコード), ALID (アラーム ID), ALTX (アラームテキスト) が与えられます。

callback

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

この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) ブロックド : 正常に送信できた。 al_list に受信したアラームリスト情報が格納されています。 (2) 非ブロックド : 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。(応答メッセージが得られなかった)

### (4) 説明

装置に対し、alid\_list 内に指定された count 分のアラーム ID のアラームリストを S5F5 メッセージを使って要求します。

正常に応答メッセージを受信できた場合は、そのメッセージをデコードし、al\_list で指定された構造体リストに格納します。

また、count = 0 の場合は、全 ALID に対するリストの要求になります。

送信要求から S5F5 応答メッセージ受信までの制御は引数の S5F5 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S5F5 送信後、応答メッセージ S5F6 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、al_list に S5F6 メッセージ内の情報がデコードされて渡されます。
あり	送信要求後、S5F5 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が 0 ならば正常で引数 al_list に S5F6 メッセージ内の情報がデコードされて渡されます。 エラーが検出された場合、(-1) が end_status にセットされます。

正常に応答メッセージを受信した場合、al\_list 構造体内にアラームリスト情報が渡されますが、ユーザ側で情報の処理を終えた後、その構造体を使用されているメモリを解放してください。

```
DshFreeTAL_S5F6_LIST(al_list);
```

### (5) コールバック関数

[C, C++]

```
API int APIX callback(
    int eqid,                // 装置 ID
    int end_status,         // 実行結果
    TAL_S5F6_LIST *al_list, // EC 名一覧リストが格納されている構造体のポインタ
    ULONG upara             // 呼出時に指定したパラメータ
);
```

[.NET VB]

```
Function callback_S5F5(ByVal eqid As Integer, ByVal end_status As Integer, ByRef list As dsh_info.TAL_S5F6_LIST, ByVal upara As Integer) As Integer
```

**[.NET C#]**

```
int CallbackS5F5(int eqid, int end_status, ref TAL_S5F6_LIST list, uint upara);
```

end\_status には以下の値が設定されます。

結果	意味
0	正常に送受信できた。
(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

end\_status = 0 のときのみ list の内容が有効です。

### 3.8.2.11 GemNotifyAlarm() アラーム通知要求

#### (1) 呼出書式

##### [C, C++]

```
API int APIX GemNotifyAlarm(
    int      eqid,           // 装置 ID
    TALID    alid,           // 通知 alid(アラーム ID)
    int      on_off,         // 発生/復旧(1/0)の指定
    int (WINAPI *AlarmCallback)(), // 実行終了時のコールバック関数
    ULONG    upara           // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function GemNotifyAlarm (
    ByVal eqid As Int32,
    ByVal alid As Int32,
    ByVal on_off As Int32,
    ByVal callback As vcallback.callback_NotifyAlarm,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int GemNotifyAlarm(
    int eqid,
    uint alid,
    int on_off,
    CallbackNotifyAlarm AlarmCallback,
    uint upara );
```

#### (2) 引数

eqid

送信する装置の ID です。

alid

ホストに通知するための ALID (アラーム ID) を指定します。

S5F1 の ALID になります。

on\_off

アラーム発生が復旧かを指定します。 発生の場合は=1、復旧の場合は=0 を指定してください。

S5F1 の ALCD の最上位ビットに反映されます。

AlarmCallback

DSHGEMLIB によるアラーム通知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することができます。

#### (3) 戻り値

戻り値	意味
-----	----

0	(1) ブロックモード：正常に送信できた。 rmsg に応答メッセージが格納されているポイントが返却される。 (2) 非ブロックモード：要求が受け付けられた。
1	alid は定義されているが Enable になっていなかった。
9	キャンセルされた。
(-1)	alid が定義されていないか、送信エラーを検出した。
(-14)	T3 タイムアウトを検出した。
その他 > 0	ブロックモードの場合、ackc5 の値が返却されます。

#### (4) 説明

ホストにアラーム (S5F1) メッセージを送信するための関数です。

DSHGEMLIB は指定された alid のメッセージを自動的に組立てた上でホストに送信します。

alid がシステム内に定義されており、報告状態が有効であるとき、ALCD, ALTX を S5F1 に設定展開した上で送信します。

また、指定 alid に CEID (イベント ID) がリンクされている場合、同時にその CEID の S6F11 メッセージも送信します。

AlarmCallback として =0 が指定されている場合は、ブロックモードになり、アラームメッセージの送信が完了してから制御が戻ってきます。

AlarmCallback が指定されている場合、要求を受け付けられた後、ただちに制御が戻ってきます。そして、アラームメッセージの送信が完了したら、コールバック関数によって完了が通知されます。

#### (5) コールバック関数

##### [c,C++]

```
API int APIX AlarmCallback(
    int end_status,           // 実行結果
    TALID alid,              // 通知した ALID
    ULONG upara              // 呼出時に指定したパラメータ
);
```

##### [.NET VB]

```
Function callback_NotifyAlarm(ByVal eqid As Integer, ByVal end_status As Integer, ByVal alid As Integer, ByVal upara As Integer) As Integer
```

##### [.NET C#]

```
int CallbackNotifyAlarm( int eqid, int end_status, uint id, 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。
1	alid は定義されているが Enable になっていなかった。
9	キャンセルされた。
(-1)	alid が定義されていないか、送信エラーを検出した。
(-14)	T3 タイムアウトを検出した。
その他 > 0	ackc5 の値が返却されます。

### 3.8.3 アラーム関連ライブラリ関数

#### 3.8.3.1 DshDecodeS5F1() - S5F1 アラームレポートのデコード

##### (1) 呼出書式

###### [C, C++]

```
API int APIX DshDecodeS5F1(
    DSHMSG *msg, // SECS メッセージ 情報構造体のポインタ
    TAL_S5F1_INFO *pinfo // デコードした情報を格納する構造体のポインタ
);
```

###### [.NET VB]

```
Function DshDecodeS5F1 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef info As dsh_info.TAL_S5F1_INFO) As Int32
```

###### [.NET C#]

```
int DshDecodeS5F1(
    ref DSHMSG msg,
    ref TAL_S5F1_INFO info );
```

##### (2) 引数

msg

S5F1 の SECS メッセージ情報が格納されている構造体のポインタです。

pinfo

デコードしたアラームレポート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	正常にデコードできた。
(-1)	msg を正しくデコードできなかった。

##### (4) 説明

S5F1 メッセージに含まれるアラームレポート情報を、ユーザプログラムが処理しやすい TAL\_S5F1\_INFO 構造体の中にデコードします。

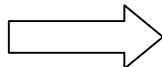
なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTAL\_S5F1\_INFO() 関数を使って開放してください。(DshResponseS5F2() を使う場合、DshResonseS5F2() が開放してくれます。)

msg S5F1

L,3

alcd  
alid  
altx

decode



### 3.8.3.2 DshMakeS5F1Response() - S5F1 の応答メッセージの生成

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshMakeS5F1Response(
    int    eqid,           // 通信対象装置 ID(0,1,2,...)
    BYTE   ackc5,         // S5F2 に設定する ACKC5 です。
    DSHMSG *msg,          // S5F2 メッセージ を格納するメッセージ 構造体のポインタ
    BYTE   *buff,        // S5F2 のテキスト格納バッファポインタ
    int    buff_size     // buff のバイトサイズ
);
```

##### [.NET VB]

##### [.NET C#]

#### (2) 引数

ackc5

S5F2 メッセージの ACKC5 です。

msg

S5F2 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

S5F2 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

S5F1 に対する S5F2 応答メッセージを msg, buff 内に作成します。

応答情報は、ackc5 を S5F2 の ACKC5 として設定します。

ackc5 の設定はユーザがアラーム情報を評価した結果です。

### 3.8.3.3 DshFreeTAL\_S5F1\_INFO () アラーム情報構造体メモリの開放

(1) 呼出書式

**[C, C++]**

```
API void APIX DshFreeTAL_S5F1_INFO(
    TAL_S5F1_INFO *info // メリを開放したいアラーム情報構造体のポインタ
);
```

**[.NET VB]**

**[.NET C#]**

(2) 引数

info

メモリを解放したいアラーム情報構造体 TAL\_S5F1\_INFO のポインタです。

(3) 戻り値

なし。

(4) 説明

TAL\_S5F1\_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、TAL\_S5F1\_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。

S5F1 メッセージのデコード関数で得られたアラーム情報構造体内に使用されているメモリを開放します。

### 3.8.3.4 DshDecodeS5F6() - S5F6 アラームリストのデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS5F6(
    DSHMSG *msg, // SECS メッセージ 情報構造体のポインタ
    TAL_S5F6_LIST *plist // デコードした情報リストを格納する構造体のポインタ
);
```

[.NET VB]

[.NET C#]

(2) 引数

msg

S5F6 の SECS メッセージ情報が格納されている構造体のポインタです。

plist

デコードしたアラームリスト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	正常にデコードできた。
(-1)	msg を正しくデコードできなかった。

(4) 説明

GemSendS5F5()関数に対する装置からの S5F6 応答メッセージに含まれるアラームリスト情報を、ユーザプログラムが処理しやすい TAL\_S5F6\_LIST 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTAL\_S5F6\_LIST()関数を使って開放してください。(DshResponseS5F6()を使う場合、DshResonseS5F6()が開放してくれます。)

msg S5F6

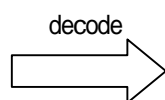
L,m

L,3

alcd

alid

altx





### 3.8.3.5 DshFreeTAL\_S5F6\_LIST () アラームリスト構造体メモリの開放

(1) 呼出書式

**[C, C++]**

```
API void APIX DshFreeTAL_S5F6_LIST(  
    TAL_S5F6_LIST *list // メリを開放したいアラームリスト構造体のポインタ  
);
```

**[.NET VB]**

**[.NET C#]**

(2) 引数

list

メモリを解放したいアラームリスト構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TAL\_S5F6\_LIST 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、TAL\_S5F6\_LIST の内容を全て0で初期設定します。

list がNULL ならば、何も処理しません。

### 3.8.4 ユーザ作成ライブラリ関数

#### 3.8.4.1 DshResponseS5F2() S5F2 アラーム報告送信応答メッセージ

##### (1) 呼出書式

###### [C, C++]

```
API int APIX DshResponseS5F2(
    int eqid,           // 通信対象装置 ID(0,16,...)
    ID_TR trid,        // DSHDR2 のトランザクション ID
    TAL_S5F1_INFO *info, // アラーム報告送信メッセージ 格納ポインタ
    int ackc5          // S5F2 応答情報格納用構造体のポインタ
);
```

###### [.NET VB]

```
Function DshResponseS5F2 (
    ByVal eqid As Int32,
    ByVal trid As Int32,
    ByRef info As dsh_info.TAL_S5F1_INFO,
    ByVal ackc5 As Int32) As Int32
```

###### [.NET C#]

```
int DshResponseS5F2(
    int eqid,
    uint trid,
    ref TAL_S5F1_INFO info,
    int ackc5 );
```

##### (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

S5F1 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

受信したアラーム報告送信 S5F1 に含まれていたレポート情報が格納されている構造体のポインタです。

ackc5

送信する応答メッセージ ack の値です。

##### (3) 戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

##### (4) 説明

アラーム報告送信メッセージ S5F1 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 ackc5 情報から S5F2 メッセージを組み立て、その後、S5F2 メッセージを送信します。

VB, C# では、dsh\_ulib クラスに属します。