DSHENG4　GEM 通信エンジン

ソフトウェア・パッケージ

DSHEng4Class

クラス・ライブラリ説明書

Vol-２　メッセージ通信クラス 編

２０１４年5月（改6）

株式会社データマップ

DSHENG4-09-30363-06-ClassLib-Comm

［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	2010 年 3 月	初版	
2	2010 年 7 月	send_sxfy() の overload ﾒｿｯﾄﾞ send()を追加	１次ﾒｯｾｰｼﾞ送信ｸﾗｽ DshSxSySend ｸﾗｽのﾒｿｯﾄﾞに send()ﾒｿｯﾄﾞを追加しました。
3	2011/02/07	ｸﾗｽに Dispose ﾒｿｯﾄﾞﾄｸｸﾗｽのﾄﾚｰｽﾓﾆﾀｰと表示機能を実装した。	ｸﾗｽに Dispose ﾒｿｯﾄﾞを追加した。 また、ﾃﾞﾊﾞｯｸﾞ機能強化のためﾕｰｻﾞがｸﾗｽの生成、消滅の回数を管理し、それらのﾀｲﾐﾝｸﾞをﾄﾚｰｽ表示できるようにした。 Vol-1 Part-2 21. DshDebug クラス参照
4	2011/06/03	1 次 m ｯｾｰｼﾞ送信ｸﾗｽに send_wait()ﾒｿｯﾄﾞを追加した。 send_request() send_request_wait() の説明を追加した。	1 次ﾒｯｾｰｼﾞ送信した後、その応答ﾒｯｾｰｼﾞを受信するまで要求元ﾌﾟﾛｸﾞﾗﾑをﾌﾞﾛｯｸするためのﾒｿｯﾄﾞを追加した。 S5F1, S6F11, S7Fx, S15Fx などが対象。 ﾕｰｻﾞ固有 1 次ﾒｯｾｰｼﾞの送受信のための DshEngine ｸﾗｽの send_request(), send_request_wait()ﾒｿｯﾄﾞの説明を追加した。
5	2012/06/05	S7F23,S7F24, S7F25 送信用ｸﾗｽを追加した。	ﾌｫｰﾏｯﾄ付きﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑのために使用するﾒｯｾｰｼﾞの送信のためのｸﾗｽです。18 章を追加した。
6	2014/05/30	文書番号の変更	ｸﾗｽﾗｲﾌﾞﾗﾘﾌﾟﾛｸﾞﾗﾐﾝｸﾞｶﾞｲﾄﾞの 文書番号を DSHEng4-09-30305-03 から DSHEng4-09-30365-03 に変更した。 （他のﾄﾞｷｭﾒﾝﾄと重複していた）

目次

１．はじめに

本説明書は、DSHENG-CLASS がサポートする GEM, GEM300 関連 SECS-Ⅱ通信メッセージの送受信のためのクラスについて機能、構文、メンバー（プロパティ、メソッド）とその使用方法について説明します。
本説明書は Vol-2 になります。

DSHENG-CLASS が管理する基本的な GEM,GEM300 関連情報に関するクラスの説明については Vol-1 の説明書を参照してください。

以下、本クラス・ライブラリがサポートするメッセージの一覧表を示します。

［ SECS-Ⅱメッセージ一覧表 ］

本ライブラリがサポートするメッセージの一覧表を示します。

この表に出ていないメッセージの送受信については、17. send_request(), send_request_wait()を参照してください。

	メッセージ	クラス名	機能概略
1	S1F3,4	-	S1F3 送信　Selected Equipment Status Request
		DshS1F4Response	S1F4 応答
2	S1F11,12	-	S1F11 送信　Status Variable Namelist Request
		DshS1F12Response	S1F12 応答
3	S1F15,16	-	S1F15 送信　Request OFF-LINE
		DshS1F16Response	S1F16 応答
4	S1F17,18	-	S1F17 送信　Request ON-LINE
		DshS1F18Response	S1F18 応答
5	S2F13,14	-	S2F13 送信　Equipment Constant Request
		-	(S2F14 はｴﾝｼﾞﾝが自動応答)
6	S2F15,16	-	S2F15 送信　New Equipment Constant Send
		-	(S2F14 はｴﾝｼﾞﾝが自動応答)
7	S2F23,24	-	S2F23 送信　Trace Initialize Send
		DshS2F24Response	S2F24 応答
8	S2F29,30	-	S2F29 送信　Equipmeny Constant Namelist Request
		-	(S2F30 はｴﾝｼﾞﾝが自動応答)
9	S2F31,32	-	S2F31 送信　Date and Time Set Requst
		-	(S2F32 はｴﾝｼﾞﾝが自動応答)
10	S2F33,34	-	S2F33 送信　Define Report
		-	(S2F34 はｴﾝｼﾞﾝが自動応答)
11	S2F35,36	-	S2F35 送信　Link Event Report
		-	(S2F36 はｴﾝｼﾞﾝが自動応答)
12	S2F37,38	-	S2F37 送信　Enable/Disabel Event Report
		-	(S2F38 はｴﾝｼﾞﾝが自動応答)
13	S2F41,42	-	S2F41 送信　Host Command Send
		DshS2F42Response	S2F42 応答
14	S2F43,44	-	S2F43 送信　Reset Spooling Stream and Function
		DshS2F44Response	S2F44 応答
15	S2F45,46	-	S2F45 送信　Define Variable Limit Attributes

		DshS2F46Response	S2F46 応答
16	S2F47,48	-	S2F47 送信　Variable Limit Attributes Request
		-	(S2F48 はｴﾝｼﾞﾝが自動応答)
17	S2F49,50	-	S2F49 送信　Enhanced Remote Command
		DshS2F50Response	S2F50 応答
18	S3F17,18	-	S3F17 送信　Carrier Action Request
		DshS3F18Response	S3F18 応答
19	S3F23,24	-	S3F23 送信　Port Group Action Request
		DshS3F24Response	S3F24 応答
20	S3F25,26	-	S3F25 送信　Port Action Request
		DshS3F24Response	S3F26 応答
21	S3F27,28	-	S3F27 送信　Change Access
		DshS3F28Response	S3F28 応答
22	S5F1,2	DshS5F1Send	S5F1 送信　Alarm Report Send
		-	S5F2 応答
23	S5F3,4	-	S5F3 送信　Enable/Disabel Alarm Send
		-	(S5F4 はｴﾝｼﾞﾝが自動応答)
24	S5F5,6	-	S5F5 送信　List Alarm Request
		-	(S5F6 はｴﾝｼﾞﾝが自動応答)
25	S6F1,S6F2	-	(S6F1 はｴﾝｼﾞﾝが自動応答) Trace Data Send
		-	S6F2 応答
26	S6F11,S6F12	DshS6F11Send	S6F11 送信　Event Report Send
			S6F12 応答
27	S6F15,S6F16	DshS6F15Send	S6F15 送信　Event Report Request
		-	(S6F16 はｴﾝｼﾞﾝが自動応答)
28	S6F19,S6F20	-	S6F19 送信　Indivisual Report Data
		-	(S6F20 はｴﾝｼﾞﾝが自動応答)
29	S6F23,S6F24	-	S6F23 送信　Request Spooled Data
		-	(S6F24 はｴﾝｼﾞﾝが自動応答)
30	S7F1,S7F2	DshS7F1Send	S7F1 送信　Process Program Load Inquire
		DshS7F2Response	S7F2 応答
31	S7F3,S7F4	DshS7F3Send	S7F3 送信　Process Program Send
		DshS7F4Response	S7F4 応答
32	S7F5,S7F6	DshS7F5Send	S7F5 送信　Process Program Data
		-	(S7F6 はｴﾝｼﾞﾝが自動応答)
33	S7F17,S7F18	DshS7F17Send	S7F17 送信　Delete Process Program Send
		-	(S7F18 はｴﾝｼﾞﾝが自動応答)
34	S7F19,S7F20	DshS7F19Send	S7F19 送信　Current EPPD Request
		-	(S7F20 はｴﾝｼﾞﾝが自動応答)
35	S7F23, S7F24	DshS7F23Send	S7F23 送信　Formatted Process Program Send
		DshS7F24Response	S7F24 応答
36	S7F25,S7F26	DshS7F25Send	S7F25 送信　Formatted Process Program Data
		-	(S7F26 はｴﾝｼﾞﾝが自動応答)
37	S10F1,S10F2	DshS10F1Send	S10F1 送信　Terminal Request
		DshS10F2Response	S10F2 応答
38	S10F3,S10F4	-	S10F3 送信　Treminal Display, Single
		DshS10F4Response	S10F4 応答
39	S10F5,S10F6	-	S10F5 送信　Terminal Display, Multi-Block

		DshS10F6Response	S10F6 応答
40	S14F9,S14F10	-	S14F9 送信　Create Object Request
		DshS14F10Response	S14F10 応答
41	S14F11,S14F12	-	S14F11 送信　Delete Object Request
		DshS14F12Response	S14F12 応答
42	S15F3,S15F4	DshS15F3Send	S15F3 送信　Recipe Name Space Action Request
		DshS15F4Response	S15F4 応答
43	S15F5,S15F6	DshS15F5Send	S15F5 送信　Recipe Name space Rename Request
		DshS15F6Response	S15F6 応答
44	S15F7,S15F8	DshS15F7Send	S15F7 送信　Recipe Space Request
		-	(S15F8 はｴﾝｼﾞﾝが自動応答)
45	S15F9,S15F10	DshS15F9Send	S15F9 送信　Recipe Status Request
		-	(S15F10 はｴﾝｼﾞﾝが自動応答)
46	S15F13,S15F14	DshS15F13Send	S15F13 送信　Recipe Create Request
		DshS15F14Response	S15F14 応答
47	S15F17,S15F18	DshS15F17Send	S15F17 送信　Recipe Retrieve Request
		DshS15F18Response	S15F18 応答
48	S16F5,S16F6	-	S16F5 送信　Process Job Command Request
		DshS16F6Response	S16F6 応答
49	S16F11,S16F12	-	S16F11 送信　PrJobCreateEnh
		DshS16F12Response	S16F12 応答
50	S16F15,S16F16	-	S16F15 送信　PrJobMultiCreate
		DshS16F16Response	S16F16 応答
51	S16F17,S16F18	-	S16F17 送信　PrJobDeque
		DshS16F18Response	S16F18 応答
52	S16F19,S16F20	-	S16F19 送信　PrGetAllJobs
		-	(S16F20 はｴﾝｼﾞﾝが自動応答)
53	S16F21,S16F22	-	S16F21 送信　PrGetSpace
		-	(S16F22 はｴﾝｼﾞﾝが自動応答)
54	S16F27,S16F28	-	S16F27 送信　Control Job Command Request
		DshS16F28Response	S16F28 応答

２．SECS-II メッセージ送受信の仕方

メッセージの送受信について説明します。

送信には、２種類の送信方法があります。

（１）非ブロック（ノンブロック）モード
非ブロックモードでは、ユーザプログラムは、送信クラスを使って送信依頼をしたら、すぐに戻ってきます。
戻り値=0 ならば、送信依頼が受諾されたことを意味し、エンジンは送信を開始します。送信後、応答メッセージ受信時に、send()メソッドの引数に指定された callback 関数を呼び出し、送受信終了が伝えられます。

（２）ブロックモード
ブロックモードでも非ブロックモードの場合と同様に、送信クラスを使って送信要求をします。ただし、要求した後、プログラムは応答メッセージを受信するまでそこで待機します。送信には send_wait()メソッドを使用します。

２．１　非ブロックモードの送信

２．１．１　送信

以下 S2F13 の送信を例に、送信処理の手順を説明します。

送信のためのクラスは、DshS2F13Send で、そのインスタンス名を send_class とします。

（１）送信メッセージのために用意されているクラスのインスタンスを生成します。

```
DshS2F13Send send_class = new DshS2F13Send();
```

（２）メッセージに含むべき情報をインスタンスのプロパティのメンバーに設定します。
設定は、メンバーに直接またはクラスが提供するメソッドを使って行います。
例えば、装置定数 ECID = EC_Mdln の要求を行う場合は次のように設定します。

```
send_class.add_vid(eng_id.EC_Mdln);          // EC_Mdln は class eng_id で定義されている
```

（３）メッセージを送信するためのメソッドを実行します。

```
int ei = send_class.send_s2f13(ref rsp_info213, cback_s2f13, 213)
```

ここで、引数について説明します。
①rsp_info213
は受信する応答メッセージ S2F14 の内容を保存するためのクラス DshV_ValueList のインスタンスです。次のように static 指定で静的変数として準備しておきます。

```
private static DshV_ValueList rsp_info213 = new DshV_ValueList();
```

②cback_s2f13

エンジンから終了通知を受けるためのコールバック(イベント通知)関数を指定します。

```
private static DshCallback.callback_s2f13 cback_s2f13 = new
                                        DshCallback.callback_s2f13(callback_S2F13);
```

コールバック関数は DshCallback クラスで定義されています。
callback_S2F13 が実際のハンドラーです。これについては （４）で説明します。

③最後の 213
uint 型の引数で、ユーザが使用できるタグです。要求したプログラムが終了通知を受けたときに使用することができます。ここでは、S2F13 の 213 を指定しています。

送信要求が受け付けられたかどうは、このメソッドの返却値で判断します。
返却値=0 ならば受け付けられたことを意味し、(-1)ならば受け付けられなかったことを意味します。

正常に受け付けられたら、終了イベント通知を待ちます。終了イベントは、メッセージ送信メソッドの引数 callback 関数として与えられたコールバック関数の呼び出しで通知されます。

（４）送信メソッドの引数に与える callback 関数について説明します。
（３）- ②で出てきました callback_S2F13 の例は次のようになります。

```
private static int callback_S2F13(int end_status, ref DshV_ValueList rsp_info,
                                                                    uint upara)
{
    DshLog.log("  ! send_info send_S2F13 Callback() end_status = " +
                                                    end_status.ToString() + "¥n");
    DshLog.log("    upara = " + upara.ToString() + "¥n");
    if (end_status == 0)
    {
        disp_v_info.disp_DshV_ValueList("----- EC value list -----", ref rsp_info);
    }
    return 0;
}
```

callback_S2F13 関数には４つの引数が付いてきます。それぞれ次の意味になります。

①end_status　：終了状態ｺｰﾄﾞです。　=0 で正常終了、 =(-1)で異常終了を示します。
②rsp_info　　：S2F14 の情報が保存されているインスタンスで、(3)の rsp_info213 です。
③upara　　：ユーザパラメータ（タグ）です。(3)で与えた 213 の値が戻されます。

（注）DshLog.log, DshV_ValueList は GemCsDemo デモプログラムに含まれる関数です。
ここでは説明を省略します。

この callback 関数は、static にしてください。そして、0 を返却してください。

以上、S2F13 を例に説明しましたが、送信要求に使用するクラス、送信関数の引数、コールバック関数の引数は、各メッセージごとに違います。詳しい内容については、３章以降に説明します。

２．１．２　受信１次メッセージに対する２次メッセージの応答の仕方

ユーザプログラムでの SECS-Ⅱ１次メッセージの受信と処理の手順は以下のようになります。

（１）DshEngine クラスを使って DshGemClass エンジンを起動します。
そして、次に、通信相手の装置起動を DshEquipment クラスを使って行います。

（２）装置の起動を行った後、DshEquipment クラスの start_poll()メソッドの実行によってエンジンは相手装置から送信されてくる１次メッセージの受信ポーリングを開始します。

（３）ポーリングが開始後、ポーリングによって受信したとき、start_poll()メソッドで指定したコールバック関数(callback 関数)を呼び出します。受信したメッセージ情報はコールバック関数の引数として渡されます。

（４）コールバック関数では、引数に与えられた情報（メッセージ格納構造体のポインタ、トランザクション ID）に従ってメッセージを処理します。

（５）メッセージの処理は、まず、メッセージ ID(stream, function)の値によって、処理を行うことになります。

（６）メッセージ ID 毎の処理は次のような内容になります。
①SECS-Ⅱメッセージ情報は、DSHMSG 構造体の中に保存されて与えられます。これを mmsg とします。
②DshGemClass ライブラリには、mmsg に含まれる情報の保存が必要とされる全てのメッセージに対してクラスが提供されています。すなわち、メッセージ ID によってクラス名が決まります。
③ユーザは、まず、メッセージ情報保存用のクラスのインスタンスを生成します。
④次に、そのインスタンスの decode()メソッドに引数 mmsg を付けて呼び出します。
decode()メソッドは、mmsg に含まれる情報をインスタンスのプロパティメンバーに保存してくれます。decode()が成功した場合は、=0 が返却され、失敗すれば=(-1)が返却されます。
⑤ユーザはインスタンスのプロパティに保存されたメッセージの情報の処理です。
プロパティの内容をエンジンに登録設定する処理などの処理を行います。

（７）次に、2 次メッセージの応答にの処理になります。
①DshGemClass ライブラリは、2 次メッセージ応答のためのクラスもメッセージ毎に設けています。
②クラスは、例えば、DshS3F18Response クラスは、S3F18 メッセージ送信のためのクラスです。
③これら 2 次メッセージ応答用クラスには、全て、response()メソッドが定義されており、このメソッドを使って応答メッセージを送信します。
④response()メソッドによっては、応答メッセージを作成するために必要な情報を準備して引数として与えられるものもあります。

（８）最後に、コールバック関数は、mmsg メッセージ情報に使用されているメモリを DshLib クラスの DshFreeMessage()メソッドを使って開放します。

次ページに S2F41 の受信処理サンプルを示します。

プログラムサンプル　S2F41 受信処理

```
public static void s2f41(uint trid, ref DSHMSG smsg)
{
    int hcack = 0;
    int err_count = 0;
    DshRCmd req_class = new DshRCmd();
    int ei = req_class.decode(ref smsg);                    // decode s2f41
    if (ei >= 0)
    {
        //  ここに処理
        hcack = proc_HostCommand( req_class );

        if (hcack != 0) err_count = 3;
    }
    else                                                    // decode error
    {
        hcack = 1;
    }
    DshRCmdRsp rsp_class = new DshRCmdRsp(hcack);
    for (int i = 0; i < err_count; i++)
    {
        rsp_class.add_ack(req_class.list[i].name, i + 1);
    }
    DshS2F42Response s2f42_class = new DshS2F42Response();
    s2f42_class.response(trid, ref rsp_class);              // send S2F42
}
```

２．２　ブロックモードの送信

ブロックモードでの送信は非ブロックもドと比較して、コールバックがないだけ、非常にシンプルになります。

例としてSxFyの送信するためのプログラミングについて説明します。

送信メッセージ sxfyを送信する場合のプログラミングの方法です。

（１）送信クラスのインスタンスを生成します。このとき、引数に管理情報のIDを指定することもあります。

```
DshSxFySend send_class = new DshSxFySend( ... );
```

(必要があれば、クラスのプロパティ値を設定したりします。)

（２）次に、send_wait()メソッドを使って、送信要求を行い、応答メッセージ受信まで待機します。

```
int ei = send_class.send_wait( arg1, arg2,.. );

if ( ei >= 0 ){
    〈正常終了処理〉
}
else
{
    〈エラー終了処理〉
}
```

非ブロックモードとの違いは、ブロックモードでは、受信結果を同じプログラムで直接得ることができ、引き続き受信結果の処理を行うことができます。そして、引数としてcallback関数とuparaの引数がありません。したがってcallback関数を準備する必要がありませんので、プログラミングがシンプルになります。

しかし、send_wait()したプログラムは、応答メッセージを受信するまでブロック状態になりますので、その間、そのプログラムは何もすることができません。

例えば、Windows のフォーム上のボタンなどのコンポーネントクリックのハンドラーの中で send_wait()を実行すると、そのフォームは send_wait()が終了するまでブロックされ、他のイベントが発生しても、それらに対するイベントを受付け、処理をすることができない状態になります。

したがって、実際のアプリケーションでは、フォームなどの他のイベント処理に影響を与えないようにするため、フォームとは別に、独立して非同期に処理することができるスレッドを作り、その中でブロックモードの送信を行うようにしてください。

３．装置定数(EC)関連メッセージの送受信

EC に関連する以下のメッセージがあります。

1	S2F13,14	－	S2F13 送信　Equipment Constant Request
		－	(S2F14 はｴﾝｼﾞﾝが自動応答)
2	S2F15,16	－	S2F15 送信　New Equipment Constant Send
		－	(S2F14 はｴﾝｼﾞﾝが自動応答)
3	S2F29,30	－	S2F29 送信　Equipmeny Constant Namelist Request
		－	(S2F30 はｴﾝｼﾞﾝが自動応答)

これら１次メッセージはホストが送信するメッセージであり、2 次メッセージの応答は全て DSHENG４エンジンが自動的に行いますので準備されているクラスはありません。

４．装置状態変数(SV)関連メッセージの送受信

SV に関連する以下のメッセージがあります。

1	S1F3,4	DshS1F3Send	S1F3 送信　　Selected Equipment Status Request
		－	S1F4 応答
2	S1F11,12	DshS1F11Send	S1F11 送信　　Status Variable Namelist Request
			S1F12 応答

これら１次メッセージはホストが送信するメッセージであり、2 次メッセージの応答は全て DSHENG４エンジンが自動的に行いますので準備されているクラスはありません

５．変数リミット(LIMIT)関連メッセージの送受信

以下のメッセージがあります。

1	S2F45,46	－	S2F45 送信　Define Variable Limit Attributes
		DshS2F46Response	S2F46 応答
2	S2F47,48	DshS2F47Send	S2F47 送信　Variable Limit Attributes Request
		－	(S2F48 はｴﾝｼﾞﾝが自動応答)

これら１次メッセージはホストが送信するメッセージであり、S2F46 応答メッセージ送信のクラスが準備されています。

５．１　S2F46Response クラス

S2F45 メッセージを送信し、S2F46 応答メッセージを受信するためのクラスです。

５．１．１　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public S2F46Response()	ｲﾝｽﾀﾝｽを生成します。

５．１．２　プロパティ

なし。

５．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S2F46 応答ﾒｯｾｰｼﾞを送信します。
2	**public void Dispose()**	**ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。**

５．１．３．１　response()

S2F46 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshLimitRspList rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S2F46 メッセージのための応答情報が保存されている DshLimitRspList クラスのインスタンスです。Vol 5.4 参照

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S2F45 に対する S2F46 メッセージを送信します。

応答メッセージに含める情報は rsp_class に指定した DshLimitRspList クラスのインスタンスです。
trid は、この応答メッセージに対する S2F45 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

５．１．３．２　Dispose()

本クラスが使用済になった際、クラス内で使用していて、開放すべきメモリがあれば、それを開放し、また Dispose すべきオブジェクトがあれば、それらを Dispose します。

【構文】

```
public void Dispose()
```

【引数】

なし。

【戻り値】

なし。

【説明】

生成された後、当該クラスが使用済みになった時点で、ユーザプログラムが明示的に使用していた資源を解放するためのメソッドです。

本メソッドが実行されるとシステムから本クラスに対する Finalizer は呼び出されません。

６．トレース(TRACE)関連メッセージの送受信

以下のメッセージがあります。

1	S2F23,24	－	S2F23 送信　Trace Initialize Send
		DshS2F24Response	S2F24 応答
2	S6F1,S6F2	－	(S6F1 はｴﾝｼﾞﾝが自動送信) Trace Data Send
		－	S6F2 応答

S2F23 はホストが送信し、エンジンが自動的に処理、応答します。
S6F1 トレースデータ送信はエンジンが自動送信します。
ここでは、S2F34 応答用クラスが準備されています。

６．１　DshS2F24Response クラス

S2F23 メッセージを送信し、S2F24 応答メッセージを受信するためのクラスです。

６．１．１　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS2F24Response()	ｲﾝｽﾀﾝｽを生成します。

６．１．２　プロパティ

なし。

６．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S2F24 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

６．１．３．１　response()

S2F24 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, int tiaack)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

tiaack

S2F24 メッセージのための応答 ACK です。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S2F23 に対する S2F24 メッセージを送信します。

tiaack は S2F24 に設定する ACK です。
trid は、この応答メッセージに対する S2F45 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

７．収集イベントとレポート関連メッセージの送受信

以下のメッセージがあります。

1	S2F35,36	－	S2F35 送信　Link Event Report
		－	(S2F36 はｴﾝｼﾞﾝが自動応答)
2	S2F37,38	－	S2F37 送信　Enable/Disabel Event Report
		－	(S2F38 はｴﾝｼﾞﾝが自動応答)
3	S2F33,34	－	S2F33 送信　Define Report
		－	(S2F34 はｴﾝｼﾞﾝが自動応答)
4	S6F11,S6F12	DshS6F11Send	S6F11 送信　Event Report Send
		－	S6F12 応答
5	S6F15,S6F16	－	S6F15 送信　Event Report Request
		－	(S6F16 はｴﾝｼﾞﾝが自動応答)
6	S6F19,S6F20	－	S6F19 送信　Indivisual Report Data
		－	(S6F20 はｴﾝｼﾞﾝが自動応答)

S6F11 送信のためのクラスが準備されています。
その他のホストからの１次メッセージに対してはエンジンが自動的に処理し応答します。

７．１　DshS6F11Send クラス

S6F11 メッセージを送信し、S6F12 応答メッセージを受信するためのクラスです。
イベントレポートを送信します。

７．１．１　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS6F11Send()	ｲﾝｽﾀﾝｽを生成します。
2	public DshS6F11Send(uint ceid)	ｲﾍﾞﾝﾄ ID を指定してｲﾝｽﾀﾝｽを生成します。

７．１．２　プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public uint ceid	送信するｲﾍﾞﾝﾄ ID(CEID)です。

７．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_ceid()	ｲﾍﾞﾝﾄ ID(CEID)を設定します。
2	public int send_s6f11() public int send()	S6F11 ﾒｯｾｰｼﾞを送信します。
3	public int send_wait()	S6F11 ﾒｯｾｰｼﾞを送信し、S6F12 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
4	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

７．１．３．１　set_ceid()

イベント ID を設定します。

【構文】

public void set_ceid(uint ceid)

【引数】

ceid

イベント ID (CEID) です。
このイベント ID にリンクされているレポート情報を S6F11 で送信します。

【戻り値】

なし。

【説明】

S6F11 に設定したいイベント ID を設定します。

７．１．３．２　send_s6f11(), send()

S6F11 メッセージの送信要求をします。

【構文】

```
public int send_s6f11(DshCallback.callback_s6f11 callback, uint upara)
public int send (DshCallback.callback_s6f11 callback, uint upara)
```

【引数】

ceid

送信するイベント ID です。

callback

S6F11 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
300	ceed=0(Enable でない)のため送信できなかった。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの ceid のレポート情報をエンジンから取得し、S6F11 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
エンジンからは、ceid にリンクされたレポートにリンクされている変数 ID の現在の変数値情報を取得し、それらを S6F11 に組み込んだ上で S6F11 メッセージを送信します。

要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答結果は引数 end_status に設定し、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s6f11(
    int end_status,                    // 終了状態ｺｰﾄﾞ
    uint upara                         // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了し，S6F12 の ack=0 であった。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。(ackc6 が 0 でなかった。)

７．１．３．３　send_wait()

S6F11 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

```
public int send_wait( uint ceid )
public int send_wait()
```

【引数】

ceid

送信するイベント ID です。

【戻り値】

返却値	意　味
0～255	正常に送受信した。 応答 ackc6=の値になります。
300	ceed=0(Enable でない)のため送信できなかった。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

ceid で指定された収集イベント ID の S6F11 メッセージを送信します。
実際の S6F11 送信処理までは、7.1.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S6F12 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

８．アラーム関連メッセージの送受信

以下のメッセージがあります。

1	S5F1,2	DshS5F1Send	S5F1 送信　　Alarm Report Send
		－	S5F2 応答
2	S5F3,4	－	S5F3 送信　　Enable/Disabel Alarm Send
		－	(S5F4 はｴﾝｼﾞﾝが自動応答)
3	S5F5,6	－	S5F5 送信　　List Alarm Request
		－	(S5F6 はｴﾝｼﾞﾝが自動応答)

S5F1 送信のためのクラスが準備されています。
その他のホストからの 1 次メッセージに対してはエンジンが自動的に処理し応答します。

８．１　DshS5F1Send クラス

S5F1 メッセージを送信し、S5F2 応答メッセージを受信するためのクラスです。
アラームレポートを送信します。

８．１．１　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS5F1Send()	ｲﾝｽﾀﾝｽを生成します。
2	public DshS5F1Send(uint alid)	ｱﾗｰﾑ ID を指定してｲﾝｽﾀﾝｽを生成します。

８．１．２　プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public uint alid	送信するｱﾗｰﾑ ID(ALID)です。

８．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_alid()	ｱﾗｰﾑ ID(ALID)を設定します。
2	public int send_s5f1() public int send()	S5F1 ﾒｯｾｰｼﾞを送信します。
3	public int send_wait()	S5F1 ﾒｯｾｰｼﾞを送信し、S5F2 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
4	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

８．１．３．１　set_alid()

アラーム ID を設定します。

【構文】

public void set_alid(uint alid)

【引数】

alid

アラーム ID (ALID) です。
このアラーム ID のアラーム情報を S5F1 で送信します。

【戻り値】

なし。

【説明】

S5F1 に設定したいアラーム ID を設定します。

８．１．３．２　send_s5f1(), send()

S5F1 メッセージの送信要求をします。

【構文】

```
public int send_s5f1(uint alid, DshCallback.callback_s5f1 callback, uint upara)
public int send(uint alid, DshCallback.callback_s5f1 callback, uint upara)

public int send_s5f1(DshCallback.callback_s5f1 callback, uint upara)
public int send(DshCallback.callback_s5f1 callback, uint upara)
```

【引数】

alid
　　アラーム ID です。
int alflag
　　アラーム発生／復旧の指定を行います。 1=発生、0=復旧です。
callback
　　S5F1 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
300	aled=0(Enable でない)のため送信できなかった。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの alid のアラーム情報をエンジンから取得し、S5F1 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答結果は引数 end_status に設定し、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s5f1(
    int end_status,                  // 終了状態ｺｰﾄﾞ
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

８．１．３．３　send_wait()

S5F1 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

```
public int send_wait( uint alid, int alflag )
public int send_wait( int alflag )
```

【引数】

alid

　　送信するアラーム ID です。

int alflag

　　アラーム発生／復旧の指定を行います。 1=発生、0=復旧です。

【戻り値】

返却値	意　味
0～255	正常に送受信した。 応答 ackc5 の値になります。
300	aled=0(Enable でない)のため送信できなかった。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

alid で指定された収集アラーム ID の S5F1 メッセージを送信します。
実際の S5F1 送信処理までは、8.1.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S5F2 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

９．プロセスプログラム関連メッセージの送受信

以下のメッセージとクラスがあります。

1	S7F1,S7F2	DshS7F1Send	S7F1 送信　　Process Program Load Inquire
		DshS7F2Response	S7F2 応答
2	S7F3,S7F4	DshS7F3Send	S7F3 送信　　Process Program Send
		DshS7F4Response	S7F4 応答
3	S7F5,S7F6	DshS7F5Send	S7F5 送信　　Process Program Data
		－	(S7F6 はｴﾝｼﾞﾝが自動応答)

９．１　DshS7F1Send クラス

S7F1 メッセージを送信し、S7F2 応答メッセージを受信するためのクラスです。
プロセスプログラム（PP）のロード問合せ情報を送信します。

９．１．１　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS7F1Send()	空のｲﾝｽﾀﾝｽを生成します。

９．１．２　プロパティ

なし。

９．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int send_s7f1() public int send()	S7F1 ﾒｯｾｰｼﾞを送信します。
2	public int send_wait()	S7F1 ﾒｯｾｰｼﾞを送信し、S7F2 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
3	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

９．１．３．１　send_s7f1(), send()

S7F1 メッセージの送信要求をします。

【構文】

```
public int send_S7F1(string ppid, uint length,
                     ref int ppgnt, DshCallback.callback_s7f1 callback, uint upara)
public int send(string ppid, uint length,
                     ref int ppgnt, DshCallback.callback_s7f1 callback, uint upara)
```

【引数】

ppid
　　プロセスプログラム ID です。
length
　　情報ロードに必要なメモリです。
ppgnt
　　S7F2 応答メッセージに含まれる ACK を保存します。
　　コールバック関数の引数になります。
callback
　　S7F1 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

引数に指定された ppid, length 情報から S7F1 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答メッセージの ppgnt を引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s7f1(
    int end_status,                  // 終了状態ｺｰﾄﾞ
    ref int ppgnt,                   // S7F2 に含まれる ACK です。
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

９．１．３．２　send_wait()

S7F1 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

public int send_wait(string ppid, uint length)

【引数】

ppid

　　プロセスプログラム ID です。

length

　　情報ロードに必要なメモリです。

【戻り値】

返却値	意　味
0	正常に送信し、応答 ppgnt=0 であった。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)
上記以外	S7F2 の ppgnt の値を返却します。

【説明】

引数に与えられた ppid, length について、S7F1 メッセージを送信します。
実際の S7F1 送信処理までは、9.1.3.1 で説明したとおりですが、本メソッドは送信の後、引き続き S7F2 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

９．２　DshS7F2Response クラス

S7F1 メッセージを受信した後、S7F2 応答メッセージを送信するためのクラスです。

９．２．１　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS7F2Response()	ｲﾝｽﾀﾝｽを生成します。

９．２．２　プロパティ

なし。

９．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S7F2 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

９．２．３．１　response()

S7F2 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, int ppgnt)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

ppgnt

S7F2 メッセージのための ACK です。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S7F1 に対する S7F2 メッセージを送信します。

応答メッセージに含める情報は ppgnt です。
trid は、この応答メッセージに対する S7F1 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

９．３　DshS7F3Send クラス

S7F3 メッセージを送信し、S7F4 応答メッセージを受信するためのクラスです。
プロセスプログラム（PP）情報を送信します。

９．３．１　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS7F3Send()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshS7F3Send(ref DshPP pclass)	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報の DshPP ｸﾗｽのｲﾝｽﾀﾝｽを指定してｲﾝｽﾀﾝｽを生成します。

９．３．２　プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public DshPP pp_class	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報(ppid, ppbody)の保存ｸﾗｽ DshPP のｲﾝｽﾀﾝｽです。

９．３．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_ppinfo()	引数に与えられたﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ情報を DshPP ｸﾗｽのｲﾝｽﾀﾝｽを使って設定します。
2	public int send_s7f3() public int send()	S7F3 ﾒｯｾｰｼﾞを送信します。
3	public int send_wait()	S7F3 ﾒｯｾｰｼﾞを送信し、S7F4 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
3	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

９．３．３．１　set_ppinfo()

ユーザが準備したDshPPクラスのインスタンス infoの内容を当該インスタンスのpp_classに設定します。

【構文】

public void set_ppinfo(ref DshPP info)

【引数】

info

プロパティpp_classに設定したいプロセスプログラム情報が保存されているDshPPのインスタンスです。
ユーザが準備します。

【戻り値】

なし。

【説明】

infoの引数のプロセスプログラムの内容を当該インスタンスのpp_classにコピーします。

９．３．３．２　send_s7f3(), send()

S7F3 メッセージの送信要求をします。

【構文】

```
public int send_S7F3( ref int ackc7, DshCallback.callback_s7f3 callback, uint upara)
public int send ( ref int ackc7, DshCallback.callback_s7f3 callback, uint upara)
public int send ( string ppid, ref int ackc7, DshCallback.callback_s7f3 callback, uint upara)
```

【引数】

ppid
　　プロセス・プログラム ID です。

ackc7
　　S7F4 応答メッセージに含まれる ACK を保存します。
　　コールバック関数の引数になります。

callback
　　S7F3 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。

upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの pp_class に保存されているプロセスプログラム情報から S7F3 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
ppid が指定された場合は、エンジンから PP 情報を pp_class プロパティに取得した上で送信します。
送信後、受信した応答メッセージの ackc7 を引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s7f3(
    int end_status,                  // 終了状態ｺｰﾄﾞ
    ref int ackc7,                   // S7F4 に含まれる ACK です。
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

９．３．３．３　send_wait()

S7F3 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

```
public int send_wait(string ppid)
public int send_wait()
```

【引数】

ppid

　　プロセスプログラム ID です。

【戻り値】

返却値	意　味
0	正常に送信し、応答 ackc7=0 であった。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)
上記以外	S7F4 の ackc7 の値を返却します。

【説明】

引数に ppid が与えられた場合には、エンジンから PP 情報を pp_class 内に取得します。
引数に与えられない場合は、予め、pp_class の ppid, ppbody プロパティに設定されていなければなりません。
実際の S7F3 送信処理までは、9.3.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S7F4 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

９．４　DshS7F4Response クラス

S7F3 メッセージを受信した後、S7F4 応答メッセージを送信するためのクラスです。

９．４．１　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS7F4Response()	ｲﾝｽﾀﾝｽを生成します。

９．４．２　プロパティ

なし。

９．４．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S7F4 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

９．４．３．１　response()

S7F4 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, int ackc7)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

ackc7

S7F4 メッセージに設定する応答 ACK です。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S7F3 に対する S7F4 メッセージを送信します。

応答メッセージに含める情報は ackc7 です。
trid は、この応答メッセージに対する S7F3 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

９．５　DshS7F5Send クラス

S7F5 メッセージを送信し、S7F6 応答メッセージを受信するためのクラスです。
相手装置にプロセスプログラム（PP）情報の応答を要求します。

９．５．１　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS7F5Send()	ｲﾝｽﾀﾝｽを生成します。
2	public DshS7F5Send(string ppid)	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID を指定してｲﾝｽﾀﾝｽを生成します。

９．５．２　プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string ppid	送信するﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID(PPID)です。

９．５．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_ppid()	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID(PPID)を設定します。
2	public int send_s7f5() public int send()	S7F5 ﾒｯｾｰｼﾞを送信します。
3	public int send_wait()	
4	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

９．５．３．１　set_ppid()

プロセスプログラム ID を設定します。

【構文】

public void set_ppid(string ppid)

【引数】

ppid

プロセスプログラム ID (PPID) です。
このプロセスプログラム ID の情報を S7F5 で要求します。

【戻り値】

なし。

【説明】

S7F5 に設定したいプロセスプログラム ID を設定します。

９．５．３．２　send_s7f5(), send()

S7F5 メッセージの送信要求をします。

【構文】

```
public int send_S7F5(string ppid, ref DshPP rspinfo,
                              DshCallback.callback_s7f5 callback, uint upara)
public int send(string ppid, ref DshPP rspinfo,
                              DshCallback.callback_s7f5 callback, uint upara)
public int send_S7F5( ref DshPP rspinfo,
                              DshCallback.callback_s7f5 callback, uint upara)
public int send( ref DshPP rspinfo,
                              DshCallback.callback_s7f5 callback, uint upara)
```

【引数】

rspinfo

S7F6 応答メッセージに含まれるプロセスプログラム情報保存用です。コールバック関数の引数になります。

callback

S7F5 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの ppid のプロセスプログラム情報の応答を S7F5 で相手装置に要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答メッセージに含まれるプロセスプログラム情報格納用インスタンス rspinfo に保存し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s7f5(
    int end_status,                  // 終了状態ｺｰﾄﾞ
    ref DshPP rspinfo,               // S7F6 に含まれる PP 情報です。 Vol-1 11.1 参照
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

９．５．３．３　send_wait()

S7F5 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

```
public int send_wait(string ppid, ref DshPP rspinfo)
public int send_wait(ref DshPP rspinfo)
```

【引数】

ppid

プロセスプログラム ID です。

rspinfo

S7F6 応答メッセージに含まれるプロセスプログラム情報保存するためのクラスのインスタンスです。

【戻り値】

返却値	意　味
0	正常に送信し、S7F6 を正常に受信した。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数に ppid が与えられない場合は、予め、プロパティ ppid を設定しておく必要があります。
実際の S7F5 送信処理までは、9.5.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S7F6 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

正常に受信できた場合は、rspinfo 内に PP 情報を設定します。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

10. レシピ関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S15F3,S15F4	DshS15F3Send	S15F3 送信　Recipe Name Space Action Request
		DshS15F4Response	S15F4 応答
2	S15F5,S15F6	DshS15F5Send	S15F5 送信　Recipe Namespace Rename Request
		DshS15F6Response	S15F6 応答
3	S15F7,S15F8	DshS15F7Send	S15F7 送信　Recipe Space Request
		-	(S15F8 はｴﾝｼﾞﾝが自動応答)
4	S15F9,S15F10	DshS15F9Send	S15F9 送信　Recipe Status Request
		-	(S15F10 はｴﾝｼﾞﾝが自動応答)
5	S15F13,S15F14	DshS15F13Send	S15F13 送信　Recipe Create Request
		DshS15F14Response	S15F14 応答
6	S15F17,S15F18	DshS15F17Send	S15F17 送信　Recipe Retrieve Request
		DshS15F18Response	S15F18 応答

10. 1 DshS15F3Send クラス

S15F3 メッセージを送信し、S15F4 応答メッセージを受信するためのクラスです。
レシピ名スペースのアクション要求を行います。

10. 1. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS15F3Send()	空のｲﾝｽﾀﾝｽを生成します。

10. 1. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string rcpid	ﾚｼﾋﾟ ID です。
2	public int rmnscmd	ｱｸｼｮﾝｺｰﾄﾞです。

10．1．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_info()	引数に与えられたﾚｼﾋﾟ ID とｱｸｼｮﾝｺｰﾄﾞを設定します。
2	public int send_s15f3() public int send()	S15F3 ﾒｯｾｰｼﾞを送信します。
3	public int send_wait()	S15F3 ﾒｯｾｰｼﾞを送信し、S15F4 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
4	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

10．1．3．1　set_info()

【構文】

public void set_info(string rcpid, int rmnscmd)

【引数】

rcpid

　　アクション要求対象レシピ ID です。

rmnscmd

　　アクションコードです。

【戻り値】

なし。

【説明】

レシピ ID とアクションコードを当該インスタンスの rcpid, rmnscmd に設定します。

10．1．3．2　send_s15f3(), send()

S15F3 メッセージの送信要求をします。

【構文】

```
public int send_S15F3(string rcpid, int rmnscmd,
            ref DshS15Rsp rspinfo, DshCallback.callback_s15f3 callback, uint upara)
public int send(string rcpid, int rmnscmd,
            ref DshS15Rsp rspinfo, DshCallback.callback_s15f3 callback, uint upara)
public int send_S15F3(ref DshS15Rsp rspinfo, DshCallback.callback_s15f3 callback, uint upara)
public int send(ref DshS15Rsp rspinfo, DshCallback.callback_s15f3 callback, uint upara)
```

【引数】

rcpid
　　レシピ ID です。

rmnscmd
　　アクションコードです。

rspinfo
　　S15F4 応答メッセージに含まれる情報を保存します。コールバック関数の引数になります。

callback
　　S15F3 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。

upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの rcpid, rmnscmd から S15F3 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答メッセージ情報を保存した rspinfo 引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s15f3(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    ref DshS15Rsp rspinfo,          // S15F4 に含まれる応答です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

10．1．3．3　send_wait()

S15F3 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

public int send_wait(string rcpid, int rmnscmd, ref DshS15Rsp rspinfo)
public int send_wait(ref DshS15Rsp rspinfo)

【引数】

rcpid
　　レシピ ID です。
rmnscmd
　　アクションコードです。
rspinfo
　　S15F4 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送信し、応答情報を rspinfo に設定します。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

rcpid, rmnscmd が引数に与えられない場合は、予め、本クラスのレシピ ID,アクションコードを、それぞれ rcpid, rmnscmd プロパティに設定しておかなければなりません。
実際の S15F3 送信処理までは、10.1.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S15F4 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

10. 2 DshS15F4Response クラス

S15F3 メッセージを受信した後、S15F4 応答メッセージを送信するためのクラスです。

10. 2. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS15F4Response()	ｲﾝｽﾀﾝｽを生成します。

10. 2. 2 プロパティ

なし。

10．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S15F4 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

10．２．３．１　response()

S15F4 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS15Rsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S15F4 メッセージに含める応答情報のインスタンスです。
DshS15Rsp クラスについては Vol-1 の 12.5 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S15F3 に対する S15F4 メッセージを送信します。
DshS15RSP クラスのインスタンス rsp_class 内の応答情報から S15F4 を生成します。
trid は、この応答メッセージに対する S15F3 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

10. 3　DshS15F5Send クラス

S15F5 メッセージを送信し、S15F6 応答メッセージを受信するためのクラスです。
S15F5 は、レシピ名の新しい名前への変更要求を行います。

10. 3. 1　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS15F5Send()	空のｲﾝｽﾀﾝｽを生成します。

10. 3. 2　プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string rcpid	ﾚｼﾋﾟ ID です。(現ﾚｼﾋﾟ名)
2	public int new_rcpid	名前を変えたい新しいﾚｼﾋﾟ ID です。

10．3．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_info()	引数に与えられたﾚｼﾋﾟ ID とｱｸｼｮﾝｺｰﾄﾞを設定します。
2	public int send_s15f5() public int send()	S15F5 ﾒｯｾｰｼﾞを送信します。
3	public int send_wait()	S15F5 ﾒｯｾｰｼﾞを送信し、S15F6 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
4	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

10．3．3．1　set_info()

【構文】

public void set_info(string rcpid, string new_rcp)

【引数】

rcpid

　　現レシピ ID（名前）です。

new_rcp

　　新しいレシピ ID（名前）です。

【戻り値】

なし。

【説明】

現レシピ ID と新レシピ名を rcpid, new_rcpid に設定します。

10．３．３．２　send_s15f5(), send()

S15F5 メッセージの送信要求をします。

【構文】

```
public int send_S15F5(string rcpid, string new_rcp,
                      ref DshS15Rsp rsp_info, DshCallback.callback_s15f5 callback, uint upara)
public int send(string rcpid, string new_rcp,
                      ref DshS15Rsp rsp_info, DshCallback.callback_s15f5 callback, uint upara)

public int send_S15F5(ref DshS15Rsp rsp_info, DshCallback.callback_s15f5 callback, uint upara)
public int send(ref DshS15Rsp rsp_info, DshCallback.callback_s15f5 callback, uint upara)
```

【引数】

rcpid
　　現レシピ ID です。
new_rcp
　　新レシピ ID です。
rspinfo
　　S15F6 応答メッセージに含まれる情報を保存します。コールバック関数の引数になります。
callback
　　S15F5 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの rcpid と new_rcpid から S15F5 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答メッセージ情報を保存した rspinfo 引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s15f5(
    int end_status,                  // 終了状態ｺｰﾄﾞ
    ref DshS15Rsp rspinfo,           // S15F6 に含まれる応答です。
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。

-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

10．３．３．３　send_wait()

S15F5 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

```
public int send_wait(string rcpid, string new_rcp, ref DshS15Rsp rsp_info)
public int send_wait(ref DshS15Rsp rsp_info)
```

【引数】

rcpid
　　レシピ ID です。
new_rcp
　　新レシピ ID です。
rspinfo
　　S15F6 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送信し、応答情報を rspinfo に設定します。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

rcpid, new_rcp が引数に与えられない場合は、予め、本クラスのレシピ ID,新レシピ ID を、それぞれ rcpid, new_rcpid プロパティに設定しておかなければなりません。
実際の S15F5 送信処理までは、10.3.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S15F6 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

10. 4 DshS15F6Response クラス

S15F5 メッセージを受信した後、S15F6 応答メッセージを送信するためのクラスです。

10. 4. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS15F6Response()	ｲﾝｽﾀﾝｽを生成します。

10. 4. 2 プロパティ

なし。

10．４．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S15F6 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

10．４．３．１　response()

S15F6 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS15Rsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S15F6 メッセージに含める応答情報のインスタンスです。
DshS15Rsp クラスについては Vol-1 の 12.5 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S15F5 に対する S15F6 メッセージを送信します。
DshS15RSP クラスのインスタンス rsp_class 内の応答情報から S15F6 を生成します。
trid は、この応答メッセージに対する S15F5 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

10. 5　DshS15F7Send クラス

S15F7 メッセージを送信し、S15F8 応答メッセージを受信するためのクラスです。
S15F7 は、指定されたオブジェクトスペックの記憶装置内での記憶容量を取得するメッセージです。

10. 5. 1　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS15F7Send()	空のｲﾝｽﾀﾝｽを生成します。

10. 5. 2　プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string objspec	ｵﾌﾞｼﾞｪｸﾄｽﾍﾟｯｸです。

10．５．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_objspec()	ｵﾌﾞｼﾞｪｸﾄｽﾍﾟｯｸを設定します。
2	public int send_s15f7() public int send()	S15F7 ﾒｯｾｰｼﾞを送信します。
3	public int send_wait()	S15F7 ﾒｯｾｰｼﾞを送信し、S15F8 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
4	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

10．５．３．１　set_objspec()

【構文】

public void set_objspec(string objspec)

【引数】

objspec

オブジェクトスペックです。

【戻り値】

なし。

【説明】

オブジェクトスペックを設定します。

10．５．３．２　send_s15f7(), send()

S15F7 メッセージの送信要求をします。

【構文】

```
public int send_S15F7(string objspec,
                      ref DshS15Rsp rsp_info, DshCallback.callback_s15f7 callback, uint upara)
public int send (string objspec,
                      ref DshS15Rsp rsp_info, DshCallback.callback_s15f7 callback, uint upara)
public int send_S15F7(ref DshS15Rsp rsp_info, DshCallback.callback_s15f7 callback, uint upara)
public int send (ref DshS15Rsp rsp_info, DshCallback.callback_s15f7 callback, uint upara)
```

【引数】

objspec
　　オブジェクトスペックです。

rspinfo
　　S15F8 応答メッセージに含まれる情報を保存します。コールバック関数の引数になります。

callback
　　S15F7 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。

upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの objspec から S15F7 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答メッセージ情報を保存した rspinfo 引数にして、callback 関数を呼び出します。
rspinfo の中の rmspace の値が記憶容量になります。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s15f7(
    int end_status,                  // 終了状態ｺｰﾄﾞ
    ref DshS15Rsp rspinfo,           // S15F8 に含まれる応答です。
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

10．５．３．３　send_wait()

S15F7 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

public int send_wait (string objspec, ref DshS15Rsp rsp_info)
public int send_wait(ref DshS15Rsp rsp_info)

【引数】

objspec
　　レシピ ID です。
rspinfo
　　S15F8 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送信し、応答情報を rspinfo に設定します。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

objspec が引数に与えられない場合は、予め、本クラスの objspec プロパティにレシピ ID を設定しておかなければなりません。
実際の S15F7 送信処理までは、10.5.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S15F8 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

10. 6 DshS15F9Send クラス

S15F9 メッセージを送信し、S15F10 応答メッセージを受信するためのクラスです。
S15F9 は、レシピの状態(state)とバージョン情報の取得を行います。

10. 6. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS15F9Send()	空のｲﾝｽﾀﾝｽを生成します。

10. 6. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string rcpid	ﾚｼﾋﾟ ID です。

10．6．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_rcpid()	ﾚｼﾋﾟ ID を設定します。
2	public int send_s15f9() public int send()	S15F9 ﾒｯｾｰｼﾞを送信します。
3	public int send_wait()	S15F9 ﾒｯｾｰｼﾞを送信し、S15F10 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
4	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

10．6．3．1　set_rcpid()

【構文】

public void set_rcpid(string rcpid)

【引数】

rcpid

レシピ ID です。

【戻り値】

なし。

【説明】

レシピ ID をプロパティの rcpid に設定します。

10．６．３．２　send_s15f9(), send()

S15F9 メッセージの送信要求をします。

【構文】

```
public int send_S15F9(string rcpid,
                      ref DshS15Rsp rsp_info, DshCallback.callback_s15f9 callback, uint upara)
public int send (string rcpid,
                      ref DshS15Rsp rsp_info, DshCallback.callback_s15f9 callback, uint upara)
public int send_S15F9(ref DshS15Rsp rsp_info, DshCallback.callback_s15f9 callback, uint upara)
public int send(ref DshS15Rsp rsp_info, DshCallback.callback_s15f9 callback, uint upara)
```

【引数】

rcpid
　　レシピ ID です。
rspinfo
　　S15F10 応答メッセージに含まれる情報を保存します。コールバック関数の引数になります。
callback
　　S15F9 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの rcpid から S15F9 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答メッセージ情報を保存した rspinfo 引数にして、callback 関数を呼び出します。
状態値とバージョン情報は、rspinfo の rcpstate と rcpver プロパティに設定されます。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s15f9(
    int end_status,                  // 終了状態ｺｰﾄﾞ
    ref DshS15Rsp rspinfo,           // S15F10 に含まれる応答です。
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

10．６．３．３　send_wait()

S15F9 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

public int send_wait (string rcpid, ref DshS15Rsp rsp_info)
public int send_wait(ref DshS15Rsp rsp_info)

【引数】

rcpid
　　レシピ ID です。
rspinfo
　　S15F10 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送信し、応答情報を rspinfo に設定します。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

rcpid が引数に与えられない場合は、予め、本クラスの rcpid プロパティにレシピ ID を設定しておかなければなりません。
実際の S15F9 送信処理までは、10.6.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S15F10 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

10. 7 DshS15F13Send クラス

S15F13 メッセージを送信し、S15F14 応答メッセージを受信するためのクラスです。
S15F13 は、レシピ情報を送信するためのメッセージです。

10. 7. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS15F13Send()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshS15F13Send(int update_flag, ref DshRecipe rclass)	生成/修正ﾌﾗｸﾞとﾚｼﾋﾟ情報を保存している DshRecipe ｸﾗｽのｲﾝｽﾀﾝｽを指定してｲﾝｽﾀﾝｽを生成します。
3	public DshS15F13Send(string rcpid)	ﾚｼﾋﾟ ID を指定してｲﾝｽﾀﾝｽを生成します。 指定されたﾚｼﾋﾟ ID の情報をｴﾝｼﾞﾝから取り出した上で S15F13 を送信します。
4	public DshS15F13Send(int update_flag, string rcpid)	生成/修正ﾌﾗｸﾞとﾚｼﾋﾟ ID を指定してｲﾝｽﾀﾝｽを生成します。 指定されたﾚｼﾋﾟ ID の情報をｴﾝｼﾞﾝから取り出した上で S15F13 を送信します。

10. 7. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public int update_flag	生成/修正ﾌﾗｸﾞです。 0=生成、1=修正を意味します。
2	public DshRecipe rcp_class	ﾚｼﾋﾟ情報が保存されている DshRecipe ｸﾗｽのｲﾝｽﾀﾝｽです。

10．7．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_update_flag()	生成/修正ﾌﾗｸﾞを設定します。 0=生成、1=修正を意味します。
2	public void set_recipe()	DshRecipe ｸﾗｽのｲﾝｽﾀﾝｽを設定します。
3	public int send_s15f13() public int send()	S15F13 ﾒｯｾｰｼﾞを送信します。
4	public int send_wait()	S15F13 ﾒｯｾｰｼﾞを送信し、S15F14 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
5	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

10．７．３．１　set_update_flag()

【構文】
public void set_update_flag(int flag)

【引数】
flag
　　生成/修正フラグを指定します。(0=生成、1=修正を意味します)

【戻り値】
なし。

【説明】
flag を update_flag に設定します。

10．７．３．２　set_recipe()

【構文】
public void set_recipe(ref DshRecipe rclass)

【引数】
rclass
　　レシピ情報が保存されている DshRecipe クラスのインスタンスです。

【戻り値】
なし。

【説明】
rclass のレシピ情報を rcp_class に設定します。(コピーします。)

10．7．3．3　send_s15f13(), send()

S15F13 メッセージの送信要求をします。

【構文】

```
public int send_s15f13( ref DshS15Rsp rsp_info,
                                    DshCallback.callback_s15f13 callback, uint upara)
public int send( ref DshS15Rsp rsp_info,
                                    DshCallback.callback_s15f13 callback, uint upara)
public int send( string rcpid, ref DshS15Rsp rsp_info,
                                    DshCallback.callback_s15f13 callback, uint upara)
public int send( int update_flag, string rcpid, ref DshS15Rsp rsp_info,
                                    DshCallback.callback_s15f13 callback, uint upara)
```

【引数】

rcpid
　　ﾚｼﾋﾟ ID です。
update_flag
　　生成/修正ﾌﾗｸﾞです。
rspinfo
　　S15F14 応答メッセージに含まれる情報を保存します。コールバック関数の引数になります。
callback
　　S15F13 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

引数で rcpid が指定された場合は、レシピ情報をエンジンから rcp_class に取り出します。
当該インスタンスの rcp_class のレシピ情報から S15F13 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答メッセージ情報を保存した rspinfo 引数にして、callback 関数を呼び出します。
状態値とバージョン情報は、rspinfo の rcpstate と rcpver プロパティに設定されます。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s15f13(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    ref DshS15Rsp rspinfo,          // S15F14 に含まれる応答です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

10．７．３．３　send_wait()

S15F13 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

```
public int send_wait (int update_flag, string rcpid, ref DshS15Rsp rsp_info)
public int send_wait (string rcpid, ref DshS15Rsp rsp_info)
public int send_wait(ref DshS15Rsp rsp_info)
```

【引数】

rcpid
　　レシピ ID です。
update_flag
　　生成/修正ﾌﾗｸﾞです。
rspinfo
　　S15F14 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送信し、応答情報を rspinfo に設定します。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数で rcpid が指定された場合は、レシピ情報をエンジンから rcp_class に取り出します。
rcpid が引数に与えられない場合は、予め、本クラスの rcp_class プロパティにレシピ情報を設定しておかなければなりません。
実際の S15F13 送信処理までは、10.7.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S15F14 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

10. 8 DshS15F14Response クラス

S15F13 メッセージを受信した後、S15F14 応答メッセージを送信するためのクラスです。

10. 8. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS15F14Response()	ｲﾝｽﾀﾝｽを生成します。

10. 8. 2 プロパティ

なし。

10．８．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S15F14 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

10．８．３．１　response()

S15F14 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS15Rsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S15F14 メッセージに含める応答情報のインスタンスです。
DshS15Rsp クラスについては Vol-1 の 12.5 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S15F13 に対する S15F14 メッセージを送信します。
DshS15RSP クラスのインスタンス rsp_class 内の応答情報から S15F14 を生成します。
trid は、この応答メッセージに対する S15F13 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

10．８．３．３　Dispose()

本クラスが使用済になった際、クラス内で使用していて、開放すべきメモリがあれば、それを開放し、またDisposeすべきオブジェクトがあれば、それらをDisposeします。

【構文】

```
public void Dispose()
```

【引数】

なし。

【戻り値】

なし。

【説明】

生成された後、当該クラスが使用済みになった時点で、ユーザプログラムが明示的に使用していた資源を解放するためのメソッドです。

本メソッドが実行されるとシステムから本クラスに対するFinalizerは呼び出されません。

10. 9 DshS15F17Send クラス

S15F17 メッセージを送信し、S15F18 応答メッセージを受信するためのクラスです。
S15F17 は、レシピ検索要求を送信するためのメッセージです。

10. 9. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS15F17Send()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshS15F17Send(string rcpid, int rcpseccode)	ﾚｼﾋﾟ ID とﾚｼﾋﾟｾｸｼｮﾝを指定してｲﾝｽﾀﾝｽを生成します。

10. 9. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string rcpid	検索対象ﾚｼﾋﾟ ID です。
2	public int rcpseccode	検索対象のﾚｼﾋﾟｾｸｼｮﾝｺｰﾄﾞです。

10．9．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_info()	ﾚｼﾋﾟ ID とﾚｼﾋﾟｾｸｼｮﾝｺｰﾄﾞを設定します。
2	public int send_s15f17() public int send()	S15F17 ﾒｯｾｰｼﾞを送信します。
3	public int send_wait()	S15F17 ﾒｯｾｰｼﾞを送信し、S15F18 を受信します。 ﾌﾟﾛｸﾞﾗﾑは S15F18 受信までﾌﾞﾛｯｸされます。
4	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。

10．9．3．1　set_info()

【構文】

public void set_info(string rcpid, int rcpseccode)

【引数】

rcpid

検索するレシピ ID です。

rcpseccode

検索するレシピセクションコードです。

【戻り値】

なし。

【説明】

レシピ ID とレシピセクションコードをプロパティの rcpid, rcpseccode に設定します。

10．9．3．2　send_s15f17(), send()

S15F17 メッセージの送信要求をします。

【構文】

```
public int send_S15F17(string rcpid, int rcpseccode,
            ref DshRcpS15F18Rsp rsp_info, DshCallback.callback_s15f17 callback, uint upara)
public int send(string rcpid, int rcpseccode,
            ref DshRcpS15F18Rsp rsp_info, DshCallback.callback_s15f17 callback, uint upara)
public int send_S15F17(
            ref DshRcpS15F18Rsp rsp_info, DshCallback.callback_s15f17 callback, uint upara)
public int send(
            ref DshRcpS15F18Rsp rsp_info, DshCallback.callback_s15f17 callback, uint upara)
```

【引数】

rcpid
　　検索するレシピ ID です。
rcpseccode
　　検索するレシピセクションコードです。
rspinfo
　　S15F18 応答メッセージに含まれる情報を保存します。コールバック関数の引数になります。
callback
　　S15F17 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの rcpid と rcpseccode の情報から S15F17 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答メッセージ情報を保存した rspinfo 引数にして、callback 関数を呼び出します。
rspinfo に検索結果情報が保存されます。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s15f17(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    ref DshS15F18Rsp rspinfo,       // S15F18 に含まれる応答です。  Vol-1 12.6 参照
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-18	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

10．9．3．3　send_wait()

S15F17 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

public int send_wait(string rcpid, int rcpseccode, ref DshRcpS15F18Rsp rsp_info)
public int send_wait(ref DshRcpS15F18Rsp rsp_info)

【引数】

rcpid
　　検索するレシピ ID です。
rcpseccode
　　検索するレシピセクションコードです。
rspinfo
　　S15F18 応答メッセージに含まれる情報を保存します。コールバック関数の引数になります。

【戻り値】

返却値	意　味
0	正常に送信し、応答情報を rspinfo に設定します。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数で rcpid、rcpseccode が引数に与えられない場合は予め本クラスのプロパティにレシピ ID とセクションコードを設定しておかなければなりません。
実際の S15F17 送信処理までは、10.9.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S15F18 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

10. 10 DshS15F18Response クラス

S15F17 メッセージを受信した後、S15F18 応答メッセージを送信するためのクラスです。

10. 10. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS15F18Response()	ｲﾝｽﾀﾝｽを生成します。

10. 10. 2 プロパティ

なし。

10. 10. 3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S15F18 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

10. 10. 3．1　response()

S15F18 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS15F18Rsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S15F18 メッセージに含める応答情報のインスタンスです。
DshS15F18Rsp クラスについては Vol-1 の 12.6 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S15F17 に対する S15F18 メッセージを送信します。
DshS15RSP クラスのインスタンス rsp_class 内の応答情報から S15F18 を生成します。
trid は、この応答メッセージに対する S15F17 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

11. プロセス・ジョブ関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S16F5,S16F6	-	S16F5 送信　Process Job Command Request
		DshS16F6Response	S16F6 応答
2	S16F11,S16F12	-	S16F11 送信　PrJobCreateEnh
		DshS16F12Response	S16F12 応答
3	S16F15,S16F16	-	S16F15 送信　PrJobMultiCreate
		DshS16F16Response	S16F16 応答
4	S16F17,S16F18	-	S16F17 送信　PrJobDeque
		DshS16F18Response	S16F18 応答
5	S16F19,S16F20	-	S16F19 送信　PrGetAllJobs
		-	(S16F20 はｴﾝｼﾞﾝが自動応答)
6	S16F21,S16F22	-	S16F21 送信　PrGetSpace
		-	(S16F22 はｴﾝｼﾞﾝが自動応答)

1 次メッセージは全てホストから送信されます。
応答メッセージ送信用のクラスが準備されています。

11. 1 DshS16F6Response クラス

S16F5 メッセージを受信した後、S16F6 応答メッセージを送信するためのクラスです。

11. 1. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS16F6Response()	ｲﾝｽﾀﾝｽを生成します。

11. 1. 2 プロパティ

なし。

11．1．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S16F6 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

11．1．3．1　response()

S16F6 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS16Rsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S16F6 メッセージに含める応答情報のインスタンスです。
DshS16Rsp クラスについては Vol-1 の 12.5 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S16F5 に対する S16F6 メッセージを送信します。
DshS16Rsp クラスのインスタンス rsp_class 内の応答情報から S16F6 を生成します。
trid は、この応答メッセージに対する S16F5 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

11. 2　DshS16F12Response クラス

S16F11 メッセージを受信した後、S16F12 応答メッセージを送信するためのクラスです。

11. 2. 1　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS16F12Response()	ｲﾝｽﾀﾝｽを生成します。

11. 2. 2　プロパティ

なし。

11. 2. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S16F12 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

11. 2. 3. 1 response()

S16F12 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS16Rsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S16F12 メッセージに含める応答情報のインスタンスです。
DshS16Rsp クラスについては Vol-1 の 12.5 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S16F11 に対する S16F12 メッセージを送信します。
DshS16Rsp クラスのインスタンス rsp_class 内の応答情報から S16F12 を生成します。
trid は、この応答メッセージに対する S16F11 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

11. 3　DshS16F16Response クラス

S16F15 メッセージを受信した後、S16F16 応答メッセージを送信するためのクラスです。

11. 3. 1　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS16F16Response()	ｲﾝｽﾀﾝｽを生成します。

11. 3. 2　プロパティ

なし。

11．３．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S16F16 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

11．３．３．１　response()

S16F16 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS16MultiPrjRsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S16F16 メッセージに含める応答情報のインスタンスです。
DshS16MultiPrjRsp クラスについては Vol-1 の 13.6 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S16F15 に対する S16F16 メッセージを送信します。
DshS16MultiPrjRsp クラスのインスタンス rsp_class 内の応答情報から S16F16 を生成します。
trid は、この応答メッセージに対する S16F15 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

11. 4　DshS16F18Response クラス

S16F17 メッセージを受信した後、S16F18 応答メッセージを送信するためのクラスです。

11. 4. 1　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS16F18Response()	ｲﾝｽﾀﾝｽを生成します。

11. 4. 2　プロパティ

なし。

11．４．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S16F18 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

11．４．３．１　response()

S16F18 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS16MultiPrjRsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S16F18 メッセージに含める応答情報のインスタンスです。
DshS16MultiPrjRsp クラスについては Vol-1 の 13.6 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S16F17 に対する S16F18 メッセージを送信します。
DshS16MultiPrjRsp クラスのインスタンス rsp_class 内の応答情報から S16F18 を生成します。
trid は、この応答メッセージに対する S16F17 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

12. コントロール・ジョブ関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S14F9,S14F10	-	S14F9 送信　Create Object Request
		DshS14F10Response	S14F10 応答
2	S14F11,S14F12	-	S14F11 送信　Delete Object Request
		DshS14F12Response	S14F12 応答
3	S16F27,S16F28	-	S16F27 送信　Control Job Command Request
		DshS16F28Response	S16F28 応答

1 次メッセージは全てホストから送信されます。
応答メッセージ送信用のクラスが準備されています。

12. 1　DshS14F10Response クラス

S14F9 メッセージを受信した後、S14F10 応答メッセージを送信するためのクラスです。

12. 1. 1　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS14F10Response()	ｲﾝｽﾀﾝｽを生成します。

12. 1. 2　プロパティ

なし。

12. 1. 3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S14F10 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

12. 1. 3．1　response()

S14F10 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS14Rsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S14F10 メッセージに含める応答情報のインスタンスです。
DshS14Rsp クラスについては Vol-1 の 12.5 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S14F9 に対する S14F10 メッセージを送信します。
DshS16Rsp クラスのインスタンス rsp_class 内の応答情報から S14F10 を生成します。
trid は、この応答メッセージに対する S14F9 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

12. 2 DshS14F12Response クラス

S14F11 メッセージを受信した後、S14F12 応答メッセージを送信するためのクラスです。

12. 2. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS14F12Response()	ｲﾝｽﾀﾝｽを生成します。

12. 2. 2 プロパティ

なし。

12. 2. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S14F12 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

12. 2. 3. 1 response()

S14F12 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS14Rsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S14F12 メッセージに含める応答情報のインスタンスです。
DshS14Rsp クラスについては Vol-1 の 12.5 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S14F11 に対する S14F12 メッセージを送信します。
DshS16Rsp クラスのインスタンス rsp_class 内の応答情報から S14F12 を生成します。
trid は、この応答メッセージに対する S14F11 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

12. 3 DshS16F28Response クラス

S16F27 メッセージを受信した後、S16F28 応答メッセージを送信するためのクラスです。

12. 3. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS16F28Response()	ｲﾝｽﾀﾝｽを生成します。

12. 3. 2 プロパティ

なし。

12. 3. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S16F28 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

12. 3. 3. 1 response()

S16F28 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshS16F27Rsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S16F28 メッセージに含める応答情報のインスタンスです。
DshS16F27Rsp クラスについては Vol-1 の 14.15 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S16F27 に対する S16F28 メッセージを送信します。
DshS16F27Rsp クラスのインスタンス rsp_class 内の応答情報から S16F28 を生成します。
trid は、この応答メッセージに対する S16F27 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

13. スプール関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S2F43,44	-	S2F43 送信　Reset Spooling Stream and Function
		DshS2F44Response	S2F44 応答
2	S6F23,S6F24	-	S6F23 送信　Request Spooled Data
		-	(S6F24 はｴﾝｼﾞﾝが自動応答)

S2F43 に対する応答メッセージ送信用クラスが準備されています。

13. 1 DshS2F44Response クラス

S2F43 メッセージを受信した後、S2F44 応答メッセージを送信するためのクラスです。

13. 1．1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS2F44Response()	ｲﾝｽﾀﾝｽを生成します。

13. 1．2 プロパティ

なし。

13．1．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S2F44 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

13．1．3．1　response()

S2F44 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshSpoolRsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S2F44 メッセージに含める応答情報のインスタンスです。
DshSpoolRsp クラスについては Vol-1 の 15.5 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S2F43 に対する S2F44 メッセージを送信します。
DshSpoolRsp クラスのインスタンス rsp_class 内の応答情報から S2F44 を生成します。
trid は、この応答メッセージに対する S2F43 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

14. ホストコマンド、キャリアアクション関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S2F41,42	-	S2F41 送信　Host Command Send
		DshS2F42Response	S2F42 応答
2	S2F49,50	-	S2F49 送信　Enhanced Remote Command
		DshS2F50Response	S2F50 応答
3	S3F17,18	-	S3F17 送信　Carrier Action Request
		DshS3F18Response	S3F18 応答
4	S3F23,24	-	S3F23 送信　Port Group Action Request
		DshS3F24Response	S3F24 応答
5	S3F25,26	-	S3F25 送信　Port Action Request
		DshS3F26Response	S3F26 応答
6	S3F27,28	-	S3F27 送信　Change Access
		DshS3F28Response	S3F28 応答

1 次メッセージは全てホストから送信されます。
応答メッセージ送信用のクラスが準備されています。

14. 1　DshS2F42Response クラス

S2F41 メッセージを受信した後、S2F42 応答メッセージを送信するためのクラスです。

14. 1．1　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS2F42Response()	ｲﾝｽﾀﾝｽを生成します。

14. 1．2　プロパティ

なし。

14．1．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S2F42 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

14．1．3．1　response()

S2F42 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshRCmdRsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S2F42 メッセージに含める応答情報のインスタンスです。
DshRCmdRsp クラスについては Vol-1 の 16.10 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S2F41 に対する S2F42 メッセージを送信します。
DshRCmdRsp クラスのインスタンス rsp_class 内の応答情報から S2F42 を生成します。
trid は、この応答メッセージに対する S2F41 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

14. 2 DshS2F50Response クラス

S2F49 メッセージを受信した後、S2F50 応答メッセージを送信するためのクラスです。

14. 2．1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS2F50Response()	ｲﾝｽﾀﾝｽを生成します。

14. 2．2 プロパティ

なし。

14. 2. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S2F50 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

14. 2. 3. 1 response()

S2F50 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshRCmdRsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S2F50 メッセージに含める応答情報のインスタンスです。
DshRCmdRsp クラスについては Vol-1 の 16.10 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S2F49 に対する S2F50 メッセージを送信します。
DshRCmdRsp クラスのインスタンス rsp_class 内の応答情報から S2F50 を生成します。
trid は、この応答メッセージに対する S2F49 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

14. 3 DshS3F18Response クラス

S3F17 メッセージを受信した後、S3F18 応答メッセージを送信するためのクラスです。

14. 3．1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS3F18Response()	ｲﾝｽﾀﾝｽを生成します。

14. 3．2 プロパティ

なし。

14. 3. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S3F18 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

14. 3. 3. 1 response()

S3F18 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshCarActionRsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した 1 次メッセージとの対応を取ります。

rsp_class

S3F18 メッセージに含める応答情報のインスタンスです。
DshCarActionRsp クラスについては Vol-1 の 16.12 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S3F17 に対する S3F18 メッセージを送信します。
DshCarActionRsp クラスのインスタンス rsp_class 内の応答情報から S3F18 を生成します。
trid は、この応答メッセージに対する S3F17 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

14. 4 DshS3F24Response クラス

S3F23 メッセージを受信した後、S3F24 応答メッセージを送信するためのクラスです。

14. 4. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS3F24Response()	ｲﾝｽﾀﾝｽを生成します。

14. 4. 2 プロパティ

なし。

14．４．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S3F24 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

14．４．３．１　response()

S3F24 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshPortGroupActionRsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S3F24 メッセージに含める応答情報のインスタンスです。
DshPortGroupActionRsp クラスについては Vol-1 の 16.13 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S3F23 に対する S3F24 メッセージを送信します。
DshPortGroupActionRsp クラスのインスタンス rsp_class 内の応答情報から S3F24 を生成します。
trid は、この応答メッセージに対する S3F23 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

14. 5　DshS3F26Response クラス

S3F25 メッセージを受信した後、S3F26 応答メッセージを送信するためのクラスです。

14. 5．1　コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS3F26Response()	ｲﾝｽﾀﾝｽを生成します。

14. 5．2　プロパティ

なし。

14. 5. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S3F26 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

14. 5. 3. 1 response()

S3F26 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshPortActionRsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した 1 次メッセージとの対応を取ります。

rsp_class

S3F26 メッセージに含める応答情報のインスタンスです。
DshPortActionRsp クラスについては Vol-1 の 16.13 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S3F25 に対する S3F26 メッセージを送信します。
DshPortActionRsp クラスのインスタンス rsp_class 内の応答情報から S3F26 を生成します。
trid は、この応答メッセージに対する S3F25 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

14. 6 DshS3F28Response クラス

S3F27 メッセージを受信した後、S3F28 応答メッセージを送信するためのクラスです。

14. 6. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS3F28Response()	ｲﾝｽﾀﾝｽを生成します。

14. 6. 2 プロパティ

なし。

14．６．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S3F28 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

14．６．３．１　response()

S3F28 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, ref DshPortAccessRsp rsp_class)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

rsp_class

S3F28 メッセージに含める応答情報のインスタンスです。
DshPortAccessRsp クラスについては Vol-1 の 16.15 を参照してください。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S3F27 に対する S3F28 メッセージを送信します。
DshPortAccessRsp クラスのインスタンス rsp_class 内の応答情報から S3F28 を生成します。
trid は、この応答メッセージに対する S3F27 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

15. 端末表示関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S10F1,S10F2	DshS10F1Send	S10F1 送信　Terminal Request
		-	S10F2 応答
2	S10F3,S10F4	-	S10F3 送信　Treminal Display, Single
		DshS10F4Response	S10F4 応答
3	S10F5,S10F6	-	S10F5 送信　Terminal Display, Multi-Block
		DshS10F6Response	S10F6 応答

S10F1 メッセージの送信クラスと S10F3,S10F5 の応答メッセージ送信クラスが準備されています。

15. 1 DshS10F1Send クラス

S10F1 メッセージを送信し、S10F2 応答メッセージを受信するためのクラスです。

15. 1．1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS10F1Send()	ｲﾝｽﾀﾝｽを生成します。
2	public DshS10F1Send(DshTermMsg info)	表示情報保存ｸﾗｽ DshTermMsg のｲﾝｽﾀﾝｽを指定してｲﾝｽﾀﾝｽを生成します。

15. 1．2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public DshTermMsg t_info	端末表示情報を保存するｸﾗｽです。 Vol-1 17.1 参照

15. 1．3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int send_s10f1() public int send()	S10F1 ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

15．1．3．1　send_s10f1(), send()

S10F1 メッセージの送信要求をします。

【構文】

```
public int send_s10f1(ref int ackc10,
                              DshCallback.callback_s10f1 callback, uint upara)
public int send(ref int ackc10,
                              DshCallback.callback_s10f1 callback, uint upara)
```

【引数】

ackc10

S10F2 応答メッセージの ACK 情報保存用です。コールバック関数の引数になります。

callback

S10F1 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの t_info に保存されている端末表示情報から S10F1 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答メッセージの ackc10 を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s10f1(
    int end_status,                  // 終了状態ｺｰﾄﾞ
    ref int ackc10,                  // S10F2 に含まれる ackc10 です。
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

15．１．３．２　send_wait()

S10F1 メッセージの送信要求をし,引き続き応答メッセージも受信します。

【構文】

public int send_wait(ref int ackc10)

【引数】

ackc10

S10F2 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

当該インスタンスの t_info に保存されている端末表示情報から S10F1 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。

本メソッドは送信の後、引き続き S10F2 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。ei=0 の場合は、受信した応答 ACK が ackc10 に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

15. 2 DshS10F4Response クラス

S10F3 メッセージを送信し、S10F4 応答メッセージを受信するためのクラスです。

15. 2. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS10F4Response()	ｲﾝｽﾀﾝｽを生成します。

15. 2. 2 プロパティ

なし。

15．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S10F4 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

15．２．３．１　response()

S10F4 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, int ackc10)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

ackc10

S10F4 メッセージのための応答 ACK です。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S10F3 に対する S10F4 メッセージを送信します。

ackc10 は S10F4 に設定する ACK です。
trid は、この応答メッセージに対する S2F45 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

15. 3 DshS10F6Response クラス

S10F5 メッセージを送信し、S10F6 応答メッセージを受信するためのクラスです。

15. 3. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS10F6Response()	ｲﾝｽﾀﾝｽを生成します。

15. 3. 2 プロパティ

なし。

15．３．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S10F6 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

15．３．３．１　response()

S10F6 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, int ackc10)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

ackc10

S10F6 メッセージのための応答 ACK です。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S10F5 に対する S10F6 メッセージを送信します。

ackc10 は S10F6 に設定する ACK です。
trid は、この応答メッセージに対する S10F5 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

16. オンライン／オフライン、日付時刻設定メッセージ送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S1F15,16	-	S1F15 送信　Request OFF-LINE
		DshS1F16Response	S1F16 応答
2	S1F17,18	-	S1F17 送信　Request ON-LINE
		DshS1F18Response	S1F18 応答
3	S2F31,32	-	S2F31 送信　Date and Time Set Requst
		-	(S2F32 はｴﾝｼﾞﾝが自動応答)

ホストからのオンライン、オフライン要求メッセージに対する応答メッセージ送信クラスが準備されています。

16. 1 DshS1F16Response クラス

S1F15 メッセージを送信し、S1F16 応答メッセージを受信するためのクラスです。

16. 1．1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS1F16Response()	ｲﾝｽﾀﾝｽを生成します。

16. 1．2 プロパティ

なし。

16．1．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S1F16 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

16．1．3．1　response()

S1F16 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, int oflack)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

oflack

S1F16 メッセージのための応答 ACK です。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S1F15 に対する S1F16 メッセージを送信します。

oflack は S1F16 に設定する ACK です。
trid は、この応答メッセージに対する S1F15 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

16. 2 DshS1F18Response クラス

S1F17 メッセージを送信し、S1F18 応答メッセージを受信するためのクラスです。

16. 2. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS1F18Response()	ｲﾝｽﾀﾝｽを生成します。

16. 2. 2 プロパティ

なし。

16．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S1F18 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

16．２．３．１　response()

S1F18 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, int onlack)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

onlack

S1F18 メッセージのための応答 ACK です。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S1F17 に対する S1F18 メッセージを送信します。

onlack は S1F18 に設定する ACK です。
trid は、この応答メッセージに対する S1F17 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

17. ユーザ固有メッセージの送受信

ユーザ固有メッセージの送受信のために次の３つのメソッドが DshEngine に準備されています。

- send_request() メソッド　　(非ブロックモードでの送受信)
- send_request_wait()メソッド(ブロックモードでの送受信)
- send_response()メソッド

これらは、static なメソッドになっていますので、DshEngine.send_request(..)のように呼び出してください。

17. 1　send_request()

GEM でサポートされていない、あるいは、エンジンで標準サポートされていないユーザ独自の 1 次メッセージの送信を行います。

構文】

```
public static int send_request(ref DSHMSG smsg, ref DSHMSG rmsg,
                               DshCallback.callback_send_request callback, uint upara)
```

【引数】

smsg

SECS-Ⅱ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。
メッセージの組立ては、ユーザが行います。

rmsg

応答 2 次メッセージ情報を格納するための構造体のポインタです。

callback

send_request()が終了したときに呼び出されるコールバック関数(イベントハンドラー)です。

upara

ユーザが callback で指定した関数が呼び出された際に、引数で渡して欲しいデータです。

【戻り値】

返却値	意　味
0	正常に要求が受付された。
＜ 0	send_request 要求に失敗した。

【説明】

本メソッドはユーザが組み立てた smsg に保存されている 1 次メッセージの送信を行います。そして、受信した応答メッセージを rmsg に格納し、callback でユーザに報せます。

本メソッドは、static の関数として準備されていますので、インスタンスを生成しないで次のコーディングで使用できます。

```
DshEngine.send_request(.... )
```

send_request()メソッドがエンジンに受け付けられた場合、返却値 0 で戻ります。受け入れられなかった場合は(-1)が返却されます。

そして、送信でき、応答メッセージを受信できたら、callback によって指定された関数を呼び出します。その際、送信結果と受信メッセージを引数として与えます。

ユーザは、本メソッドを実行する前に、smsg 構造体内に 1 次メッセージを組立てセットしなければなりません。ストリーム、ファンクション、そしてテキストなどです。詳しくは、プログラミング例を参考にしてください。

【終了通知関数】

send_request()メソッドに対する callback の書式は、DshCallback クラスに次のように定義されています。

```
public delegate int callback_send_request(ref DSHMSG rmsg, int end_status, uint upara);
```

rmsg : 応答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。
send_request()メソッドの引数で与えられた構造体のポインタです。
end_status : send_request()の処理結果です。 0 であれば正常に終了です。
(-1)であればエラー終了です。
upara : send_request()メソッドで与えられた upara の値が渡されます。

【例】 S7F5 を送信し、S7F6 を受信する処理

```
private void send_s7f5( string ppid )
    int ei = 0;
    DSHMSG smsg = new DSHMSG();                         // 送信用構造体
    DSHMSG rmsg = new DSHMSG();                         // 受信用構造体
    IntPtr buff = Marshal.AllocCoTaskMem(1024);                 // ﾒｯｾｰｼﾞ Text 用ﾊﾞｯﾌｧ
    smsg.wbit = 1;                                              // Wait bit=1
    smsg.stream = 7;                                    // S7
    smsg.function = 5;                                  // F5
    while (true)
    {
        smsg.buffer = buff;                                     // buff ptr 設定
        smsg.length = 1024;                                     // buff size 設定

        HSMS.D_InitItemPut(ref smsg);                           // smsg 構造体の put のための初期化

        ei = HSMS.D_PutItem(ref smsg, HSMS.ICODE_L, IntPtr.Zero, 1);     // L-1 ｾｯﾄ
        if (ei < 0) break;

        ei = HSMS.D_PutItem(ref smsg, HSMS.ICODE_A, .ppid, ppid.Length); // PPID  ｾｯﾄ
        break;
    }
    if (ei < 0)
    {
        DshLog.log("  !! Message setup error¥n");               // 組立てｴﾗｰ
        return;
    }
    ei = DshEngine.send_request(ref smsg, ref rmsg, cback_request_s7f5, 705); // 送信
    if (ei < 0)
    {
        DshLog.log("  !! send_request() error¥n");
    }
    Marshal.FreeCoTaskMem(buff);                        // buff ﾒﾓﾘ開放
}
```

送受信完了で呼び出される callback 関数

```
private static int callback_send_request_s7f5(int end_status, ref DSHMSG rmsg, uint upara)
{
    string ppid ="";
    string ppbody ="";
    IntPtr ptr = Marshal.AllocCoTaskMem( 1024 );                          // dataitem 値取得用ﾊﾞｯﾌｧ

    DshLog.log("  ! send_request callback() end_status = " + end_status.ToString() + "¥r¥n");
    if (end_status == 0)
    {
        formid.fm.OutLog(" APP S" + rmsg.stream.ToString() + "F" + rmsg.function.ToString() + " rcvd");
        formid.fm.OutLog("     length=" + rmsg.length.ToString());
        HSMS.D_InitItemGet(ref rmsg);                                     // rmsg 初期化
        int n = 0;
        int ei = 0;
        while( true )
        {
            n = HSMS.D_GetItem(ref rmsg, HSMS.ICODE_L, IntPtr.Zero, 0);    / L-2
            if ( n != 2 )
            {
                ei = (-1); break;
            }
            n = HSMS.D_GetItem( ref rmsg, HSMS.ICODE_A, ptr, 1024 );       // PPID
            if ( n < 0 )
            {
                ei = (-1); break;
            }
            ppid = DshLib.DshPtrToString(ptr, 1024, n);
            n = HSMS.D_GetItem(ref rmsg, HSMS.ICODE_A, ptr, 1024);         // PPBODY
            if (n < 0)
            {
                ei = (-1); break;
            }
            ppbody = DshLib.DshPtrToString(ptr, 1024, n);
            break;        }
        if (ei == 0)
        {
            DshLog.log("    ppid     = " + ppid + "¥n");
            DshLog.log("    ppbody   = " + ppbody + "¥n");
        }
        else
        {
            DshLog.log("  !! Message format error" + "¥n");
        }
        break;
    }
    Marshal.FreeCoTaskMem(ptr);                                           // ptr ﾒﾓﾘ開放
    return 0;
}
//----- callback 用 instance
private static DshCallback.callback_send_request cback_request_s7f5 =
                              new DshCallback.callback_send_request(callback_send_request_s7f5);
```

17. 2　send_request_wait()

GEM でサポートされていない、あるいは、エンジンで標準サポートされていないユーザ独自の 1 次メッセージをブロックモードで送信を行います。

構文】

```
public static int send_request_wait(ref DSHMSG smsg, ref DSHMSG rmsg)
```

【引数】

smsg

SECS-Ⅱ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。
メッセージの組立ては、ユーザが行います。

rmsg

応答 2 次メッセージ情報を格納するための構造体のポインタです。

【戻り値】

返却値	意　味
0	正常に要求が受付された。
＜ 0	send_request_wait 送受信に失敗した。

【説明】

本メソッドはユーザが組み立てた smsg に保存されている 1 次メッセージの送信を行います。そして、受信した応答メッセージを rmsg に格納します。

本メソッドは、send_request()との違いは、send_request()は非ブロックモードの送受信であり、send_request_wait()は、ブロックモードでの送受信になります。
ブロックモードでは、応答メッセージの受信までプログラムはブロック（待ち）状態になります。

本メソッドは、static の関数として準備されていますので、インスタンスを生成しないで次のコーディングで使用できます。

```
DshEngine.send_request_wait(.... )
```

send_request_wait()メソッドが正常に完了したときは返却値 0 で戻ります。異常を検出した場合は負の値(＜ 0)が返却されます。

ユーザは、本メソッドを実行する前に、smsg 構造体内に 1 次メッセージを組立てセットしなければなりません。ストリーム、ファンクション、そしてテキストなどです。詳しくは、次ページのプログラミング例を参考にしてください。

それから、受信メッセージの処理が終了したら、rmsg の中にメッセージ格納用に使用されたメモリの開放を必ず実行してください。実行しないとメモリリークが発生します。

DshEngine.dsh_free_buffer() メソッドを使って開放します。

```
DshEngine.dsh_free_buffer( ref rmsg );   // rmsg 内のﾊﾞｯﾌｧﾒﾓﾘを開放する。
```

【例】　S7F5 を送信し、S7F6 を受信する処理

```
private void send_s7f5_wait( string ppid )
     int ei = 0;
     DSHMSG smsg = new DSHMSG();                          // 送信用構造体
     DSHMSG rmsg = new DSHMSG();                          // 受信用構造体
     IntPtr buff = Marshal.AllocCoTaskMem(1024);                  // ﾒｯｾｰｼﾞ Text 用ﾊﾞｯﾌｧ
     smsg.wbit = 1;                                               // Wait bit=1
     smsg.stream = 7;                                     // S7
     smsg.function = 5;                                   // F5
     while (true){
         smsg.buffer = buff;                                      // buff ptr 設定
         smsg.length = 1024;                                      // buff size 設定
         HSMS.D_InitItemPut(ref smsg);                            // smsg 構造体の put のための初期化
         ei = HSMS.D_PutItem(ref smsg, HSMS.ICODE_L, IntPtr.Zero, 1);     // L-1 ｾｯﾄ
         if (ei < 0) break;
         ei = HSMS.D_PutItem(ref smsg, HSMS.ICODE_A, .ppid, ppid.Length); // PPID  ｾｯﾄ
         break;
     }
     if (ei < 0){
         DshLog.log("  !! Message setup error¥n");                        // 組立てｴﾗｰ
         return;
     }
     ei = DshEngine.send_request_wait(ref smsg, ref rmsg);                // 送受信
     if (ei < 0){
         DshLog.log("  !! send_request_wait() error¥n");
     }else{
        IntPtr ptr = Marshal.AllocCoTaskMem(1024);
        HSMS.D_InitItemGet(ref rmsg);                                     // rmsg 初期化
        int n = 0; int ei = 0;
        while( true ){
            n = HSMS.D_GetItem(ref rmsg, HSMS.ICODE_L, IntPtr.Zero, 0);    // L-2
            if ( n != 2 ){
                ei = (-1); break;
            }
            n = HSMS.D_GetItem( ref rmsg, HSMS.ICODE_A, ptr, 1024 );       // PPID
            if ( n < 0 ){
                ei = (-1); break;
            }
            ppid = DshLib.DshPtrToString(ptr, 1024, n);
            n = HSMS.D_GetItem(ref rmsg, HSMS.ICODE_A, ptr, 1024);         // PPBODY
            if (n < 0){
                ei = (-1); break;
            }
            ppbody = DshLib.DshPtrToString(ptr, 1024, n);
            break;
        }
        if (ei == 0) {
            DshLog.log("    ppid     = " + ppid + "¥n");
            DshLog.log("    ppbody   = " + ppbody + "¥n");
        }else{
            DshLog.log("  !! Message format error" + "¥n");
        }
        DshEngine.dsh_free_msg_buffer( ref rmsg );  // !!  これを必ず実行すること。
    }
    Marshal.FreeCoTaskMem(ptr);                                           // ptr ﾒﾓﾘ開放
    Marshal.FreeCoTaskMem(buff);                                          // buff ﾒﾓﾘ開放
}
```

17. 3　send_response()

GEM でサポートされていない、あるいは、エンジンで標準サポートされていないユーザ独自の 2 次メッセージの応答送信を行います。

構文】

```
public static int send_response(uint trid, ref DSHMSG rmsg)
```

【引数】

rrmsg

SECS-Ⅱ応答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。
メッセージの組立ては、ユーザが行います。

【戻り値】

返却値	意　味
0	正常に要求が受付された。
＜ 0	send_response 要求に失敗した。

【説明】

本メソッドはユーザが組み立てた rmsg に保存されている 2 次メッセージの送信を行います。

本メソッドは、static の関数として準備されていますので、インスタンスを生成いないで次のコーディングで使用できます。

```
DshEngine.send_response(.... )
```

send_response()メソッドがエンジンに受け付けられた場合、返却値 0 で戻ります。受け入れられなかった場合は(-1)が返却されます。

ユーザは、本メソッドを実行する前に、rmsg 構造体内に 2 次メッセージを組立てセットしなければなりません。ストリーム、ファンクション、そしてテキストなどです。詳しくは、プログラミング例を参考にしてください。

本メソッドは、エンジンに応答メッセージの送信を要求し、それが受付されてから後、送信が終了しても特に通知はありません。

【例】 S10F1 を受信した後、S10F2 を send_response を使って送信します。

```
public static void s10f1(int eqid, uint trid, ref DSHMSG smsg)
{

   // <ここで、S10F1 の処理を行う。
   // 以下、S10F2 ﾒｯｾｰｼﾞを準備し、応答送信します。

    int ackc10 = 0;

    DSHMSG rmsg = new DSHMSG();                                    // 2 ｼﾞﾒｯｾｰｼﾞ情報格納構造体
    rmsg.stream = 10;                                              // S10
    rmsg.function = 2;                                             // F2
    rmsg.wbit = 0;                                                 // w-bit = 0
    IntPtr buff = Marshal.AllocCoTaskMem(128);                     // text 用ﾊﾞｯﾌｧﾒﾓﾘ確保
    rmsg.buffer = buff;
    rmsg.length = 128;
    HSMS.D_InitItemPut(ref rmsg);                                  // rmsg 構造体初期化
    HSMS.D_PutItem(ref rmsg, HSMS.ICODE_B, ref ackc10, 1);                 // ackc10 を設定

    DshEngine.send_response(trid, ref rmsg);            // 応答送信

    Marshal.FreeCoTaskMem(buff);                                   //text 用ﾊﾞｯﾌｧﾒﾓﾘ開放
    return;
}
```

18. フォーマット付きプロセスプログラム関連メッセージの送受信

以下のメッセージとクラスがあります。

1	S7F23,S7F24	DshS7F23Send	S7F23 送信　　Formatted Process Program Send
		DshS7F24Response	S7F24 応答
2	S7F25,S7F26	DshS7F25Send	S7F25 送信　　Formatted Process Program Data
		－	(S7F26 はｴﾝｼﾞﾝが自動応答するので準備していません。)

18. 1 DshS7F23Send クラス

S7F23 メッセージを送信し、S7F24 応答メッセージを受信するためのクラスです。
S7F23 はフォーマット付きプロセスプログラム (FPP) 情報を送信するメッセージです。

18. 1. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS7F23Send()	空のｲﾝｽﾀﾝｽを生成します。
2	public DshS7F23Send(ref DshFPP pclass)	FPP 情報の DshFPP ｸﾗｽのｲﾝｽﾀﾝｽを指定してｲﾝｽﾀﾝｽを生成します。

18. 1. 2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public DshFPP fpp_class	FPP 情報(fppid,mdln,..)の保存ｸﾗｽ DshFPP のｲﾝｽﾀﾝｽです。

18．1．3　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_ppinfo()	引数に与えられた FPP 情報を DshFPP ｸﾗｽのｲﾝｽﾀﾝｽを使って設定します。
2	public int send_s7f23() public int send()	S7F23 ﾒｯｾｰｼﾞを送信し、S7F24 応答ﾒｯｾｰｼﾞを受信します。応答はｺｰﾙﾊﾞｯｸで受け取ります。
3	public int send_wait()	S7F23 ﾒｯｾｰｼﾞを送信し、S7F24 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
3	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

18．1．3．1　set_ppinfo()

ユーザが準備したDshTraceのクラスのインスタンスの内容を当該インスタンスのfpp_classに設定します。

【構文】

public void set_ppinfo(ref DshFPP info)

【引数】

info

プロパティ fpp_class に設定したい FPP 情報が保存されている DshFPP のインスタンスです。
ユーザが準備します。

【戻り値】

なし。

【説明】

info の引数の FPP の内容を当該インスタンスの fpp_class にコピーします。

18．１．３．２　send_s7f23(), send()

S7F23 メッセージの送信要求をします。

【構文】

```
public int send_S7F23( ref int ackc7, DshCallback.callback_s7f23 callback, uint upara)
public int send ( ref int ackc7, DshCallback.callback_s7f23 callback, uint upara)
public int send ( string fppid, ref int ackc7, DshCallback.callback_s7f23 callback, uint upara)
```

【引数】

fppid
　　FPPID です。
ackc7
　　S7F24 応答メッセージに含まれる ACK を保存します。
　　コールバック関数の引数になります。
callback
　　S7F23 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの fpp_class に保存されている FPP 情報から S7F23 メッセージを生成し、それを相手装置に送信するようエンジンに要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
fppid が指定された場合は､エンジンから FPP 情報を fpp_class プロパティに取得した上で送信します。
送信後、受信した応答メッセージの ackc7 を引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s7f23(
    int end_status,                  // 終了状態ｺｰﾄﾞ
    ref int ackc7,                   // S7F24 に含まれる ACK です。
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

18. 1. 3. 3 send_wait()

S7F23 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

```
public int send_wait(string fppid)
public int send_wait()
```

【引数】

fppid

FPPID です。

【戻り値】

返却値	意　味
0	正常に送信し、応答 ackc7=0 であった。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)
上記以外	S7F24 の ackc7 の値を返却します。

【説明】

引数に fppid が与えられた場合には、エンジンから FPP 情報を fpp_class 内に取得します。
引数に与えられない場合は、予め、fpp_class の fppid, mdln, softrev, プロセスコマンド情報がプロパティに設定されていなければなりません。
実際の S7F23 送信処理までは、9.3.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S7F24 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

18. 2 DshS7F24Response クラス

S7F23 メッセージを受信した後、S7F24 応答メッセージを送信するためのクラスです。

18. 2. 1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS7F24Response()	ｲﾝｽﾀﾝｽを生成します。

18. 2. 2 プロパティ

なし。

18．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int response()	S7F24 応答ﾒｯｾｰｼﾞを送信します。
2	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽのDispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

18．２．３．１　response()

S7F24 メッセージの応答送信要求をします。

【構文】

public int response(uint trid, int ackc7)

【引数】

trid

エンジンからポーリング時に与えられたトランザクション ID です。
処理した１次メッセージとの対応を取ります。

ackc7

S7F24 メッセージに設定する応答 ACK です。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。(ﾄﾗﾝｻﾞｸｼｮﾝ ID が正しくなかった。)

【説明】

受信した S7F23 に対する S7F24 メッセージを送信します。

応答メッセージに含める情報は ackc7 です。
trid は、この応答メッセージに対する S7F23 受信時にエンジンから与えられたトランザクション ID です。
trid がエンジンの管理の中に見つからなかった場合には、(-1)を返却します。
要求が受け入れられたら 0 を返却します。この後、応答メッセージ送信結果は特に通知されません。

18. 3 DshS7F25Send クラス

S7F25 メッセージを送信し、S7F26 応答メッセージを受信するためのクラスです。
相手装置にフォーマット付きプロセスプログラム（FPP）情報の応答を要求します。

18. 3．1 コンストラクタ

オーバーロードの一覧を示します。

	名前	説明
1	public DshS7F25Send()	ｲﾝｽﾀﾝｽを生成します。
2	public DshS7F25Send(string fppid)	FPPID を指定してｲﾝｽﾀﾝｽを生成します。

18. 3．2 プロパティ

プロパティ一覧表に示します。

	名前	説明
1	public string fppid	送信する FPPID です。

18．３．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public void set_fppid()	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID(FPPID)を設定します。
2	public int send_s7f25() public int send()	S7F25 ﾒｯｾｰｼﾞを送信します。 応答はｺｰﾙﾊﾞｯｸ関数で渡されます。
3	public int send_wait()	S7F25 ﾒｯｾｰｼﾞを送信し、S7F26 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。
4	public void Dispose()	ｸﾗｽ内部で使用された資源(ﾒﾓﾘ)をｼｽﾃﾑに返却します。 他のｸﾗｽをﾌﾟﾛﾊﾟﾃｨにしている場合、そのｸﾗｽの Dispose()も行います。 5.1.3.2 と同様です。そちらを参照ください。

18．３．３．１　set_fppid()

FPPID を設定します。

【構文】

public void set_fppid(string fppid)

【引数】

fppid

フォーマット付きプロセスプログラム ID (FPPID) です。

【戻り値】

なし。

【説明】

S7F25 に設定したいプロセスプログラム ID を設定します。

18．３．３．２　send_s7f25(), send()

S7F25 メッセージの送信要求をします。

【構文】

```
public int send_S7F25(string fppid, ref DshFPP rspinfo,
                              DshCallback.callback_s7f25 callback, uint upara)
public int send(string fppid, ref DshFPP rspinfo,
                              DshCallback.callback_s7f25 callback, uint upara)
public int send_S7F25( ref DshFPP rspinfo,
                              DshCallback.callback_s7f25 callback, uint upara)
public int send( ref DshFPP rspinfo,
                              DshCallback.callback_s7f25 callback, uint upara)
```

【引数】

rspinfo

S7F26 応答メッセージに含まれる FPP 情報保存用です。コールバック関数の引数になります。

callback

S7F25 送信後、2 次メッセージ受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

当該インスタンスの fppid のフォーマット付きプロセスプログラム情報の応答を S7F25 で相手装置に要求します。
要求がエンジンによって受け入れられたときは、0 を返却します。受け入れられなかった場合は(-1)を返却します。
送信後、受信した応答メッセージに含まれるプロセスプログラム情報格納用インスタンス rspinfo に保存し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_s7f25(
    int end_status,                  // 終了状態ｺｰﾄﾞ
    ref DshFPP rspinfo,              // S7F26 に含まれる FPP 情報です。 Vol-1 11.1 参照
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

18．３．３．３　send_wait()

S7F25 メッセージを送信し、引き続き応答メッセージを受信も行います。

【構文】

```
public int send_wait(string fppid, ref DshFPP rspinfo)
public int send_wait(ref DshFPP rspinfo)
```

【引数】

fppid

FPPID です。

rspinfo

S7F26 応答メッセージに含まれるフォーマット付きプロセスプログラム情報保存するためのクラスのインスタンスです。

【戻り値】

返却値	意　味
0	正常に送信し、S7F26 を正常に受信した。
(-1)	送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数に fppid が与えられない場合は、予め、プロパティ fppid を設定しておく必要があります。
実際の S7F25 送信処理までは、9.5.3.2 で説明したとおりですが、本メソッドは送信の後、引き続き S7F26 応答メッセージを待機し、受信が終了したら、上の【戻り値】に示した値を返却します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

正常に受信できた場合は、rspinfo 内に FPP 情報を設定します。

送信エラー、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。