DSHEng5 装置／ホスト通信エンジン・ライブラリ(GEM+GEM300)

ソフトウェア・パッケージ

DSHEng5 GEM 通信エンジン・クラス説明書

Vol－1

エンジン起動・停止、通信確立関連クラス

(EngAPI、GEM 通信確立、予約装置変数関連)

２０１９年１２月（改訂-1）

株式会社データマップ

文書番号　DSHENG5-19-30321-01

［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.	2019-06-28	初版	
2.	2019.12.16	訂正と記述追加	誤字等の訂正 説明の追加など。仕様上は変更はない。
3.			
4.			

目　次

１．はじめに

DSHEng5 GEM 通信エンジン仕様書は、Vol-1 から６までの６つの Volume に分けられています。
本仕様書は Vol 番号は１です。

本説明書では、DSHEng5 通信エンジンの起動・停止、通信確立関連クラスの機能、コンストラクタ、プロパティ、メソッドなどについて説明します。

Vol 番号	文書番号	内容
Vol-1	**DSHENG5-19-30321-00**	**エンジン起動・停止、通信確立関連クラス (EngAPI、GEM 通信確立、予約装置変数関連)**
Vol-2	DSHENG5-19-30322-00	変数情報関連クラス (EC, SV, DVVAL, CE, Report, Alarm)
Vol-3	DSHENG5-19-30323-00	プロセス情報関連クラス (PP, FPP, RECIPE, PRJ, CJ, CARRIER, SUBSTRATE)
Vol-4	DSHENG5-19-30324-00	SECS-II メッセージ送信クラス
Vol-5	DSHENG5-19-30325-00	SECS-II 通信メッセージ情報保存クラス
Vol-6	DSHENG5-19-30326-00	SECS-II 通信メッセージエンコード／デコード処理クラス

DSHEng5 通信エンジンは、DSHEng4, DSHGemLib 通信エンジンの後継ソフトウェア・ライブラリパッケージです。

各エンジンの機能などの比較は下表の通りです。

項目	DSHEng5	DSHEng4	DSHGemLib
GEM300 対応	○	○	○
装置側通信機能	○	○	○
ホスト側通信機能	○	X	○
エンジン生成言語	C#	C, C++	C, C++
クラス・ライブラリ Interface 使用	△ (DSHEng4 互換用)	○	○
APP 使用言語	C# VB.NET	C# VB.NET	C# VB.NET

（1）SEMI の規定規定では、SECS-II メッセージを、Stream(S) / Function(F) の値によって以下の３つに分類しています。

①ホスト側用
②装置側用
③ホストと装置側両用

（2）すでに開発済の DSHEng4 の APP プログラムを、DSHEng5 でも一部修正することによって動作させることができます。
（プロジェクト参照 DLL の変更を行い、再ビルドする必要があります）

１．１　DSHEng5 の動作環境

（1）OS が Windows-8.1、10 プロフェッショナル版以上の性能を有するコンピュータ
（2）Microsoft .Net Framework 4.5 以上

１．２　DSHEng5 通信エンジンプログラムの名前空間

・DSHEng5 の　namespace **DSH_ENG** になります。
・APP は、言語プログラムでは、以下のように参照できるようにしてください

```
c#       - using DSH_ENG;
VB.Net   - Imports DSH_ENG
```

２．通信エンジン・システムの構成

２．１　構成－１　APP が DSHEng５と直接インタフェース

本構成は、APP が DSHEng5 のクラスを直接使用する形態です。

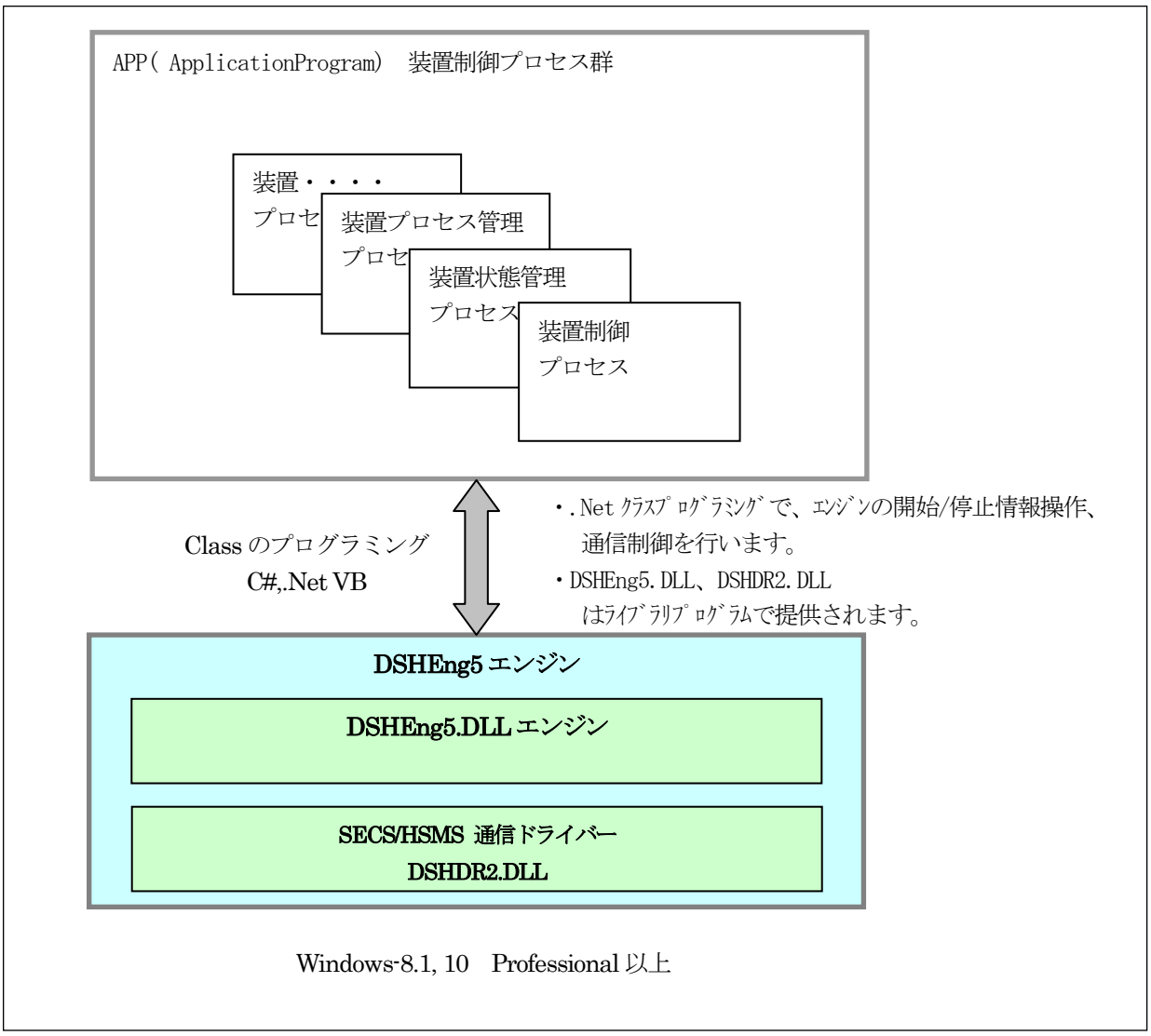


図 2-1 基本的なソフトウエア構成

２．２　構成－２　APP はクラスライブラリ(EngClass)を通してインタフェース

本構成は、従来使用されてきた DSHEng4 通信エンジンで開発されれた APP のための互換性を保つための構成です。

APP と DSHEng5 との間に、**Eng4Class ライブラリとインタフェース仕様が互換性のある Eng5Class.DLL** クラスライブラリを挟み、DSHEng5 エンジンの下で動作できるようにするための構成です。

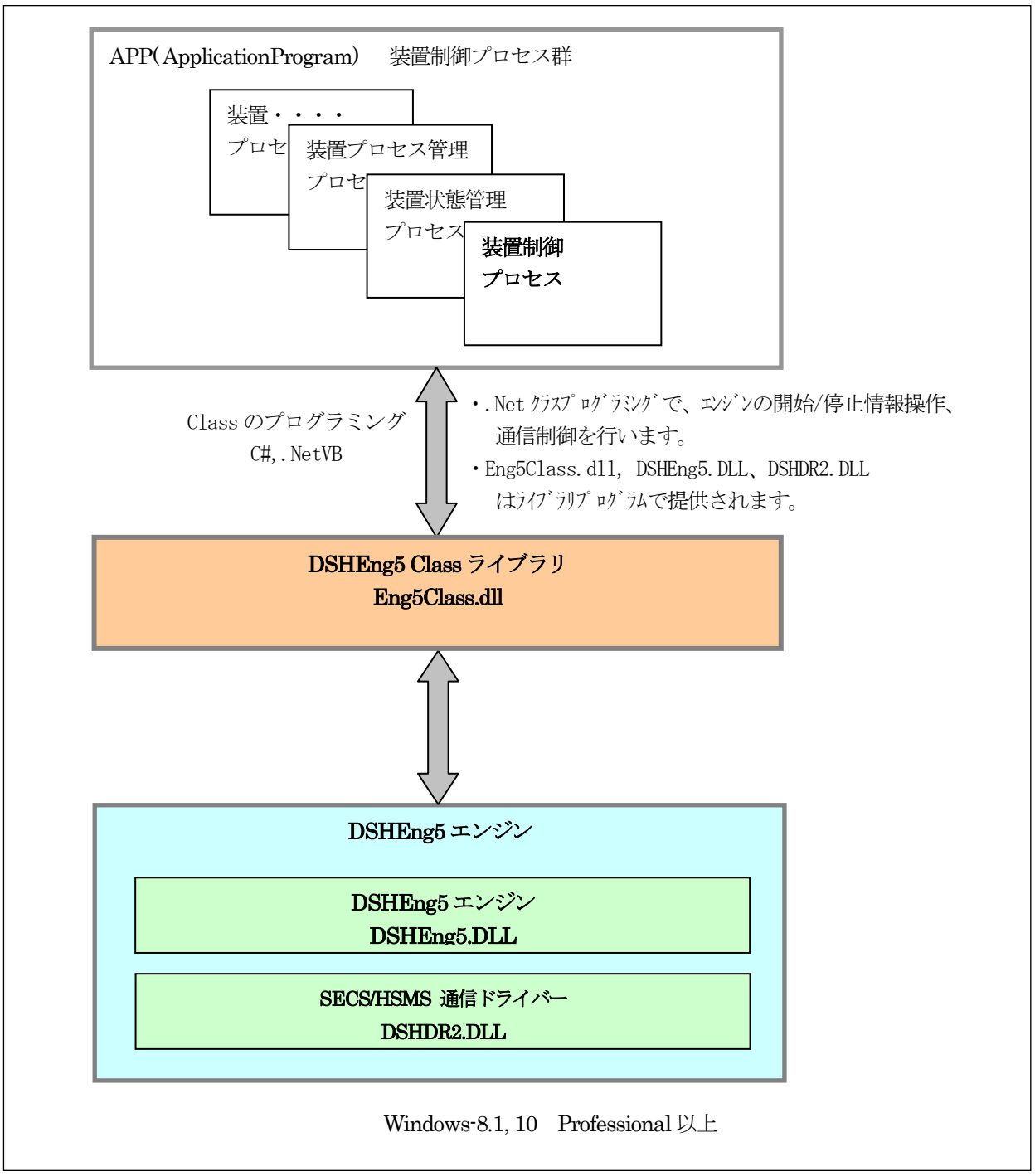


図 2-2 基本的なソフトウエア構成（Eng5Class ライブラリ使用）

３．　DSHEng5 通信エンジン・プログラムの構成

３．１　ソフトウェアの階層構造

２．１で説明した構成-1 におけるアプリケーションシステムにおけるシステムの階層構造はつぎのようになります。

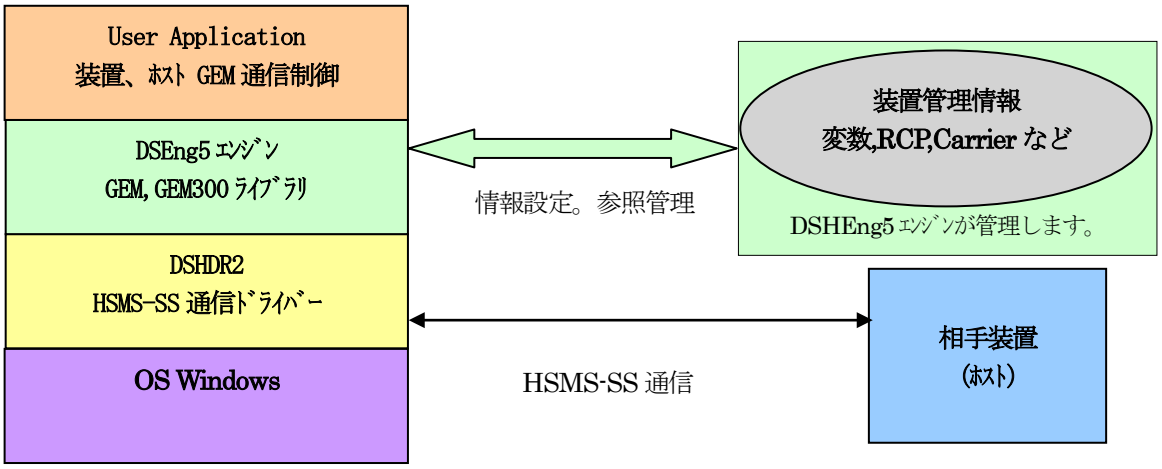


図３．１　階層構造

ユーザは、オブジェクト指向言語である Microsoft .Net(言語)を使って、GEM300 仕様に基づく情報管理と通信制御を簡単なコーディングで実現することができます。

(通常、GEM 通信関連機能の制御のために DSHEng5, DSHDR2 に対し直接的な関数の呼出を行う必要はありません。)

３．２　クラスの構成

ここでは、用途別に分けると基本的に以下の５種類になります。

	クラス
1	通信エンジン開始・停止クラス
2	変数情報管理クラス
3	変数情報保存クラス
4	SECS-II メッセージ保存情報クラス
5	SECS-II メッセージ送信・受信クラス

３．２．１　通信エンジン開始・停止クラス

	クラス名	
エンジンクラス	EngAPI	DSHEng5 通信エンジンクラス (1) 通信エンジンの開始、停止を行います。 (2) 開始時には以下の処理を行います。 ①装置起動ファイルによる基本管理情報を設定します。 ②装置変数情報管理領域と変数管理クラスを設けます。 (EC, SV, DV) ③装置変数定義ファイルによる装置変数情報の定義を行います。 ④APP の指定があれば、バックアップした変数バックアップファウルを復元します。 ⑤HSMS 通信制御関連クラスを生成し、メッセージ送受信制御を行うスレッドの起動を行います。 (3) 終了時には以下の処理を行います。 ①HSMS 通信制御関連スレッドを終了させます。 ②バックアップ処理を終了させます。 ③変数管理情報領域を解放します。 (3) 変数情報領域と管理のためのクラスを変数の種類ごとに設けます。 M_EC, M_SV など、APP が参照できるクラスを意味します。 (4) APP が必要とされる一般的な情報取得に応えるメソッドを有します。

３．２．２　変数情報管理クラス

DSHEng5 が管理する変数として各種の変数がありますが、変数別に管理するためのクラスです。

例として、SV（装置状態変数）の class_SV についてその構成を説明します。

（1）プロパティ

```
public  TV_INFO[] v_info_tab;      // ID 単位で変数情報を保存するテーブル
                                   // ( TV_INFO は 1 個の変数情報を保存するクラス)
```

（2）メソッド

```
allcate()                          // 新規登録
set()                              // クラスのプロパティの代入
get()                              // クラスのプロパティの取得
Dispose()                          // プロパティ値の廃棄
その他
```

以下、DSHEng5 が管理する変数情報管理クラスの一覧表を示します。

表-3.2.2 変数情報管理クラス一覧表

	クラス名	用途
1	class_EC	EC 装置定数情報クラス 定義登録された全 EC の情報を保存管理をします。 個別の情報の保存は TV_INFO クラスを使う。 TV_INFO には Limit 情報も含む。
2	class_SV	SV 装置状態情報クラス 定義登録された全 SV の情報を保存管理をします。 個別の情報の保存は TV_INFO クラスを使う。 TV_INFO には Limit 情報も含む。 日付時刻の更新管理も行います。
3	class_DV	DV データ値情報クラス 定義登録された全 DV の情報を保存管理します。 個別の情報の保存は TV_INFO クラスを使う。 TV_INFO には Limit 情報も含む。
4	class_Report	Report 情報クラス 定義登録された全 Report の情報を保存管理します。 個別の情報の保存は TRP_INFO クラスを使う。
5	class_CE	CE 情報クラス 定義登録された全 CE の情報を保存管理します。 個別の情報の保存は TCE_INFO クラスを使う。

6	class_Alarm	Alarm 情報クラス 定義登録された全 Alarm の情報を保存管理します。 個別の情報の保存は TAL_INFO クラスを使う。
7	class_PP	PP(Process Program)情報クラス 登録された全 PP の情報を保存管理します。 個別の情報の保存は TPP_INFO クラスを使う。
8	class_FPP	FPP(Formatted Process Program)情報クラス 登録された全 FPP の情報を保存管理します。 個別の情報の保存は TFPP_INFO クラスを使う。
9	class_RCP	RCP(Recipe)情報クラス 登録された全 RCP の情報を保存管理します。 個別の情報の保存は TRCP_INFO クラスを使う。
10	class_PRJ	PRJ(Process Job)情報クラス 登録された全 PRJ の情報を保存管理します。 個別の情報の保存は TPRJ_INFO クラスを使う。
11	class_CJ	CJ(Control Job)情報クラス 登録された全 CJ の情報を保存管理します。 個別の情報の保存は TCJ_INFO クラスを使う。
12	class_CAR	Carrier 情報クラス 登録された全 Carrier の情報を保存管理します。 個別の情報の保存は TCAR_INFO クラスを使う。
13	class_SUBST	Substrate 情報クラス 登録された全 Substrate の情報を保存管理します。 個別の情報の保存は TSUBST_INFO クラスを使う。
14	class_TRACE	TRACE 情報クラス 登録された全 Trace の情報を保存管理します。 個別の情報の保存は TTRACE_INFO クラスを使う。
15	class_Spool	Spool 情報クラス 登録された全 Spool の情報を保存管理します。 個別の情報の保存は TSPOOL_INFO クラスを使う。

３．２．３　変数情報保存クラス

DSHEng5 が管理する各種の装置変数を個別に管理するためのクラスです。

例として、EC,、SV、DV 変数で使用する TV_INFO クラスについては概略次のようになります。

（1）プロパティ

```
public  UInt32  vid ;          // ID 単位で変数情報を保存するテーブル
                               // ( TV_INFO は 1 個の変数情報を保存するクラス)
```

（2）メソッド

```
set_value()                    // 変数値の設定
get_value()                    // 変数値の取得
Dispose()                      // プロパティ値の廃棄
その他
```

以下、DSHEng5 が管理する各変数の変数単位クラスの一覧表を示します。

表-3-2-3 変数情報保存クラス一覧表

	クラス名	用途
1	TV_INFO	1 個の EC, SV または DV 変数情報を保存するためのクラスです。 変数が有する ID, プロパティの設定／取得を行うメソッドで構成されます。
2	TRP_INFO	1 個の Report 情報を保存するためのクラスです。 Report が有する ID, プロパティの設定／取得を行うメソッドで構成されます。Link される変数(EC,SV,DV)ID 情報も含む。
3	TCE_INFO	1 個の CE 情報を保存するためのクラスです。 CE が有する ID, プロパティの設定／取得を行うメソッドで構成されます。 Link される Report ID 情報も含む。
4	TAL_INFO	1 個の Alarm 情報を保存するためのクラスです。 Alarm が有する ID, プロパティの設定／取得を行うメソッドで構成されます。
5	TPP_INFO	1 個の PP 情報を保存するためのクラスです。 PP が有する ID, プロパティの設定／取得を行うメソッドで構成されます。
6	TFPP_INFO	1 個の FPP 情報を保存するためのクラスです。 FPP が有する ID, プロパティの設定／取得を行うメソッドで構成されます。
7	TRCP_INFO	1 個の Recipe 情報を保存するためのクラスです。 Recipe が有する ID, プロパティの設定／取得を行うメソッドで構成されます。
8	TPRJ_INFO	1 個の PRJ 情報を保存するためのクラスです。 PRJ が有する ID, プロパティの設定／取得を行うメソッドで構成されます。
9	TCJ_INFO	1 個の CJ 情報を保存するためのクラスです。 CJ が有する ID, プロパティの設定／取得を行うメソッドで構成されます。

10	TCAR_INFO	１個の Carrrier 情報を保存するためのクラスです。 Carrier が有する ID, プロパティの設定／取得を行うメソッドで構成されます。
11	TSUBST_INFO	１個の Substrate 情報を保存するためのクラスです。 Substrate が有する ID, プロパティの設定／取得を行うメソッドで構成されます。
12	TTRACE_INFO	１個の Trace 情報を保存するためのクラスです。 Trace が有する ID, プロパティの設定／取得を行うメソッドで構成されます。 変数 SV がトレース対象になる。（関連メッセージ ： S2F23, S6F1）
13	TLIMIT_INFO	装置変数(EC, SV, DV)値のリミット監視情報を保存するためのクラスです。 本情報は、変数情報 TV_INFO クラスのプロパティに含まれます。 （関連メッセージ ： S2F45, S2F47）
14	TSPOOL_INFO	Spool 情報の保存クラスです。 （関連メッセージ ： S2F43, S6F23）

３．２．４　SECS-II メッセージ情報保存クラス

DSHEng5 は、送信、受信メッセージに含まれる情報を処理するときに、各メッセージに応じてメッセージ情報クラス内に情報を保存します。

送信時には、クラスのインスタンスに保存された情報を SECS-II のメッセージの構造に合わせ組立（Encode）を行います。

また、受信時には、SECS-II のメッセージをクラスのインスタンス内に解読（（Decode）保存します。

表-3-2-4 メッセージ情報保存クラス一覧表

	クラス名	用途
1	TRCMD_INFO	S2F41 - HOST Command メッセージ情報を保存します。
2	TLIMIT_LIST	S2F45　- 変数リミットリストメッセージ情報を保存します。
3	TERCMD_INFO	S2F49　- 拡張リモートコマンドメッセージ情報を保存します。
4	TCACT_INFO	S3F17 - Carrier Action メッセージ情報を保存します。
5	TPORTG_INFO	S3F23 - Port Group Action メッセージ情報を保存します。
6	TPORT_INFO	S3F25 - Port Action メッセージ情報を保存します。
7	TACCESS_INFO	S3F27 - Port Access Change メッセージ情報を保存します。
8	TAL_INFO	S5F1 - アラーム通知メッセージ情報を保存します。 （3.2.2 のｱﾗｰﾑ情報保存用）と同じ）

３．２．５　SECS-II メッセージ送信、受信関連クラス

ユーザが DSHEng5 がサポートしている SECS-II メッセージを送信／受信するために使用されるクラスは下表のとおりです。

表-3-2-5 SECS-II メッセージ送信、受信クラス一覧表

	クラス名	用途
メッセージ送信・受信クラス	class_SendSxFy	SxFy の送信を行うクラスです。 x ： stream, f ： function code 例 class_SendS15F13　クラス **【参照文書】DSHENG5-19-30324-00** **DSHEng5 GEM 通信エンジン・クラス説明書 Vol-4** **SECS-II メッセージ送信クラス**
	class_SendReqSxFy	User 固有の SECS-II メッセージの送信を行うクラスです。 ユーザが作成（Encode）したメッセージを送信するために使用します。 ﾒｿｯﾄﾞ： **SendRequest(), SendRequest_wait(),SendResponse()** **【参照文書】DSHENG5-19-30324-00** **DSHEng5 GEM 通信エンジン・クラス説明書 Vol-4 － 18.1, 18.2**
	class_TPRI_PASS	APP が受信処理したいメッセージのメッセージ ID（S,F）の登録メソッドを提供します。 set_stream_func(int stream, int function(int s, int f); 【参照文書】**本説明書の 4.3 を参照**ください。
	class_MsgPassPri	受信した 1 次メッセージを APP に渡すためのクラスです。 APP は初期化時に本クラスを使って、受信メッセージ情報を受取るための イベントハンドラーを設定する必要があります。 ①EventHandler の設定(APP が使用する) void set_ev_handler(class_CALLBACK.DshMsgPollEventHandler handler)

４．通信エンジン開始・停止・管理クラス

４．１　EngAPI クラス　－　エンジン開始／停止/管理クラス

エンジンのメイン・クラスであり、開始、停止処理を行います。

エンジン開始時に使用する、予約変数の設定などのメソッドを提供します。

４．１．１　コンストラクタ

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

EngAPI クラスのインスタンスを生成します。
この後、APP は、エンジン機能（変数管理、通信制御）を使用することができます。

４．１．２　プロパティ

以下、プロパティを一覧表に示します。

４．１．２．１　エンジン起動状態、バックアップファイル関連情報

	プロパティ名	説明
1	public int activated	Engine が開始されているかどうかを示す。 0 = 停止中 1 = 実行中（開始済）
2		
3	public int bkup_flag	装置管理情報ﾊﾞｯｸｱｯﾌﾟﾌｧｲﾙから情報を復元するかどうかを指定します。 値 = 0 で復元しない，　=1 で復元する、を意味します。
4	public static string file_name public static string time_stamp public static int rec_count public static int generation	check_backup_EC()ﾒｿｯﾄﾞなどﾊﾞｯｸｱｯﾌﾟ情報の有効性の確認を行った際に取得された情報を保存します。 file_name　：有効であった最新のﾊﾞｯｸｱｯﾌﾟﾌｧｲﾙ名 time_stamp ：ﾌｧｲﾙのﾀｲﾑｽﾀﾝﾌﾟ rec_count　：ﾌｧｲﾙに保存されている情報数 generation ：有効であったﾌｧｲﾙの世代番号 0,1,2 or 3（0=最も新しい） backupfile については　３．２．６ 参照

４．１．２．２　変数 ID の最大管理登録数

最大値のデフォルト値であり、装置起動定義ファイル(¥DSHEng5¥cnf¥equip.cnf or host.cnf を使って値を変更することができます。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1	public static int max_ecid public static int max_svid public static int max_dvid public static int max_rpid public static int max_rpid public static int max_alid	EC 変数の登録可能最大 ID 数 SV　　“ DV　　“ Report　“ CE　　“ Alarm　“	256 512 512 512 512 512
2	public static int max_ppid public static int max_fppid public static int max_rcpid	PP 変数の登録可能最大 ID 数 FPP　　“ Recipe	256 256 256
3	public static int max_prjid public static int max_cjid	PRJ 変数の登録可能最大 ID 数 CJ　　“	128 128
4	public static int max_carid public static int max_substid	Carrier 変数の登録可能最大 ID 数 Substrate　“	256 256
5	public static int max_traceid	SV トレース登録可能最大 ID 数	128

４．１．２．３　変数情報管理クラス

APP は変数管理クラスから EngAPI クラス内に定義される以下の変数クラスのインスタンスを通して特定 ID の変数クラス情報を参照することができます。

変数管理クラスのインスタンスは、通信エンジンのスタート時に生成されます。(EngAPI.start()の実行による)

一般的に、変数管理クラスの APP からの参照は、変数**クラスのインスタンス名称は EngAPI.M_XXX になります。**

(例　ENGAPI.M_SV ： 装置状態変数)

SV 管理クラスのインスタンス名は M_SV ですが、APP は、この EngAPI クラスの M_SV インスタンスを通して特定 ID の SV 変数情報を参照することができます。

例えば、svid= SV_100 の変数の情報を TV_INFO class_v 内に取得する場合は次のようにプログラミングします。

```
TV_INFO class_v = new TV_INFO();
result = EngAPI.M__SV.get( SV_100, ref class_v);
```

これで、変数 SV100 のプロパティ情報を class_v に取得できます。

変数管理クラスのインスタンス名を下表に示します。

	プロパティ名	説明
1	EngAPI.M_EC	EC 変数情報管理クラスのインスタンス
2	EngAPI.M_SV	SV　　“
3	EngAPI.M_DV	DV　　“
4	EngAPI.M_RP	Report　“
5	EngAPI.M_CE	CE　　“
6	EngAPI.M_AL	Alarm　“
7	EngAPI.M_PP	PP　　“
8	EngAPI.M_FPP	FP　　“
9	EngAPI.M_RCP	RCP　　“
10	EngAPI.M_PRJ	PRJ　　“
11	EngAPI.M_CJ	CJ　　“
12	EngAPI.M_CAR	Carrier　“
13	EngAPI.M_SUBST	Substrate “
14	EngAPI.M_TRACE	Trace　　“
15	EngAPU.M_SPOOL	Spool　　“

４．１．３　メソッド

APP が使用できる EngAPI クラスのメソッドは下記一覧表のとおりです。

	メソッド名	説明
1	public int **start**()	DSHEng5 通信エンジンを開始します。
2	public void **stop**()	DSHEng5 通信エンジンを停止します。
3	public static string **get_SN**()	製品のシリアル番号を取得します。
4	public static string **get_backup_dir**()	バックアップファイルの Directory を取得します。
5	public static int **get_HSMS_state**()	HSMS-SS の接続状態を取得します。
6	public static int **get_GEM_comm_state**()	GEM 通信接続状態を取得します。
7	public static void **set_mdln_len_free**()	MDLN の文字列長を無制約設定
8	public static int **get_mdln_len_free**()	MDLN の文字列長の制約取得
9	public static void **set_softrev_len_free**()	SOFTREV の文字列長を無制約設定
10	public static int **get_softrev_len_free**()	SOFTREV の文字列長の制約取得
11	public static void **set_mdln_softrev_size**()	MDLN, SOFTREV 変数の変数値(string)のバイト長を設定します。
12	public static void **set_altx_size**()	S5F1 の ALTX の文字列長(ﾊﾞｲﾄ)を変更します。
13	public static int **get_altx_size**()	S5F1 の ALTX の文字列長（ﾊﾞｲﾄ）を取得します。
14	public static void **set_ev_handler**()	SECS-II メッセージ受信用ハンドラーの設定
15	public static void **set_S1F13_send**()	通信接続開始で 自分から S1F13 を送信するかどうかを指定します。
16	public static int **get_S1F13_send**()	通信接続開始をどちらが行うかの設定を取得します。
17	public static int **set_EC_backuop_flag()** (同様のﾒｿｯﾄﾞは SV,DV,,,)	EC 変数情報のバックアップを行うかどうかを指定します。
18	public static int **get_EC_backuop_flag()** (同様のﾒｿｯﾄﾞは SV,DV,,,)	EC 変数情報がバックアップ対象になっているかどうかを調べる。

４．１．３．１　start() - エンジン開始

DSHEng5 通信エンジンを開始します。引数、装置起動ファイル(.cnf) と通信環境定義ファイル(.def)名を付けて呼び出します。

【構文】

public int start(string equip_file, string comm_file, int bkup_flag)

【引数】

equip_file

装置起動ファイル名を指定します。
（ファイル名の拡張子が “cnf”）

comm_file

HSMS 通信ドライバーに必要な通信環境定義ファイル名を指定します。
（ファイル名の拡張子が “def”）

bkup_flag

前に保存された変数情報のバックアップファイルの内容を通信エンジン内に復元させるかどうかを指定します。(0 = 復元しない、1=復元する、の指定になります。)

【戻り値】

返却値	意　味
0	正常に開始できた。
(-1)	開始に失敗した。

【説明】

通信エンジンの開始によって、以下の処理を行います。

（1）equip_file の内容を通信エンジン内に設定します。
その内容に含まれる装置変数定義ファイル (通常 EQINFO.fil) に定義されている変数情報をエンジン内に取り込み、各変数管理領域に登録します。

（2）ログファイルの記録を開始します。
ログファイルの保存場所は装置起動ファイル内に指定されます。
名前は、equip-yyyy-mm-dd.log　 yyyy, mm, dd は年月日です。日付単位でファイルが生成されます。

（3）comm_file で指定された通信環境定義ファイルを引数にして、DSHDR2 HSMS 通信ドライバーの起動を行います。(M_EC, M_SV などのインスタンスに変数管理情報を登録します。)

（4）bkup_flag = 1 の場合、バックアップした変数情報の復元処理を行います。

（5）通信制御関連プログラム（スレッド）を起動し、SECS-II メッセージの受信/送信を可能にします。

開始が正常に終了した場合は、 =0 を返却します。

開始に失敗した場合は、=(-1)を返却します。
なお、失敗した場合、ログファイルに記録されますので、そちらを参照して原因を調査することができます。

４．１．３．２　stop()　- エンジン停止

DSHEng5 通信エンジンを停止します。
停止は、通信制御関連プログラムの停止、変数管理クラスが使用している資源を解放し、破棄します。
また、ログ・ファイルも閉じます。

【構文】

```
public void stop()
```

【引数】

なし。

【戻り値】

返却値	意　味
0	正常に終了できた。
(-1)	開始に失敗した。

【説明】

通信エンジンを停止します。
停止に伴う処理は以下の通りです。
（1）　DSHDR2 HSMS 通信ドライバーを停止します。
（2）通信制御関連プログラム（スレッド）を停止します。
（3）全変数管理クラスで使用していた資源をすべて開放し、破棄します。
（4）ログファイルを閉じる。

４．１．３．３　get_SN()　- 製品のシリアル番号情報の取得

使用中の DSHEng5 通信エンジンの製品のシリアル番号を取得します。

【構文】

```
public static string get_SN()
```

【引数】

なし。

【戻り値】

返却値	意　味
文字列	シリアル番号

【説明】

DSHEng5 通信エンジンのシリアル番号を取得します。

４．１．３．４　get_engine_type() - エンジンタイプの取得

使用中の DSHEng5 通信エンジンタイプを取得します。

【構文】

```
public static string get_eingine_type()
```

【引数】

なし。

【戻り値】

返却値	意　味
文字列	エンジンタイプです。 試用版　：″DSHEng5 Communication Engine Trial type″ 組込版　：″DSHEng5 Communication Engine Produc type″ 開発版　：″DSHEng5 Communication Engine Develop type″

【説明】

DSHEng5 通信エンジンのエンジンタイプを取得します。

- 試用版は、一定期間、評価用に使用できます。
- 組込版は、実装置用に使用されるタイプです。USB キーによるライセンスが必要になります。
- 開発版は、APP の開発専用です。ソフトウェア開発とその評価作業に使用できます。
  ソフトウェアキーライセンスが必要です。

４．１．３．５　get_engine_state() - エンジン状態の取得

使用中の通信エンジンの状態を取得します。

【構文】

```
public static int get_engine_state()
```

【引数】

なし。

【戻り値】

返却値	意　味
0	停止状態です。
1	実行状態です。

【説明】

DSHEng5 通信エンジンのシリアル番号を取得します。

４．１．３．６　get_backup_dir() - 変数バックアップファイルの保存ディレクトリの取得

変数情報のバックアップファイルの保存ディレクトリ名を取得します。

【構文】

```
public static string get_backup_dir()
```

【引数】

なし。

【戻り値】

返却値	意　味
文字列	バックアップファイル保存ディレクトリ

【説明】

DSHEng5 が管理している変数情報のバックアップファイルが保存されるディレクトリ名を取得します。

４．１．３．７　get_HSMS_state() - HSMS 通信の接続状態の取得

相手装置との HSMS-SS 通信接続状態を取得します。

【構文】

public static int get_HSMS_state ()

【引数】

なし。

【戻り値】

返却値	意　味
0	接続状態（Selection が確立）
!=0	未接続

【説明】

相手装置との HSMS 通信接続が確立しているかどうかを調べるために使用します。

４．１．３．８　get_GEM_comm_state() - GEM 通信の接続状態の取得

相手装置との GEM 通信接続状態を取得します。

【構文】

public static int get_GEM_comm_state ()

【引数】

なし。

【戻り値】

返却値	意　味
0	接続状態（GEM 通信接続が確立）
!=0	未接続

【説明】

相手装置との GEM 通信接続が確立しているかどうかを調べるために使用します。
S1F13 による通信確立のことです。

４．１．３．９　set_mdln_free() - MDLN の文字列長を無制約設定

S1F13 などに使用するデータアイテム **MDLN**(装置 Model)名の長さを制約なしに設定します。
MDLN は EC(装置定数)の定義に含まれます。

【構文】

public　static void set_mdln_len_free(int flag)

【引数】

flag

0：長さ固定（6 バイトまたは装置起動ファイルの MDLN 長の定義で決められた長さ）
1：長さ自由

【戻り値】

なし。

【説明】

通常は、MDLN の長さが決められてますが、例外的にこの長さを制限しないシステムで MDLN の長さを変更することができます。

４．１．３．10　get_mdln_free() - MDLN の文字列長の制約取得

S1F13 などに使用するデータアイテム **MDLN**(装置 Model)名の長さを取得します。

【構文】

public　static int get_mdln_len_free()

【引数】

なし。

【【戻り値】

返却値	意　味
0	制限あり（固定長）
!=0	制限なし

【説明】

MDLN の長さが制限なしかどうかの設定値を取得します。

４．１．３．11　set_softrev_free() - SOFTREV の文字列長を無制約設定

S1F13 などに使用するデータアイテム SOFTREV(Software の版番号)の長さを制約なしにします。
SOFTREV は EC(装置定数)に含まれます。

【構文】

```
public  static void set_softrev_len_free(int flag)
```

【引数】

flag

0：長さ固定（6 バイトまたは装置起動ファイルの SOFTREV 長の定義で決められた長さ）
1：長さ自由

【戻り値】

なし。

【説明】

通常は、SOFTREV の長さが決められてますが、例外的にこの長さを制限しないシステムで SOFTREV の長さを変更することができます。

４．１．３．12　get_softrev_free() - SOFTREV の文字列長の制約取得

S1F13 などに使用するデータアイテム SOFTREV(Software の版番号)の長さを取得します。

【構文】

```
public  static int get_softrev_len_free()
```

【引数】

なし。

【【戻り値】

返却値	意　味
0	制限あり（固定長）
!=0	制限なし

【説明】

SOFTREV の長さが制限なしかどうかの設定を取得します。

４．１．３．13　set_mdln_softrev_size() - MDLN, SOFTREV の長さ設定

S1F13 などに使用するデータアイテム MLDLN(装置 Model 名), SOFTREV(Software の版番号)の長さを設定します。
MDLN, SOFTREV 変数は EC(装置定数)に含まれます。

【構文】

public static void set_mdln_softrev_size(ref int m_size, ref int s_size)

【引数】

m_size

MDLN の長さ (ﾊﾞｲﾄ長)

s_size

SOFTREV の長さ (ﾊﾞｲﾄ長)

【戻り値】

なし。

【説明】

MDLN, SOFTREV の長さを設定します。長さはバイト単位です。

本設定は、MDLN, SOFTREV それぞれについて長さ制限なしの場合は、効果を持ちません。
長さ制限の設定については、下記メソッドを参照ください。

MDLN　：　set_mdln_len_free()
SOFTREV　：　set_softrev_len_free()

４．１．３．14　set_altx_size() - ALTX の文字列長約設定

S5F1 などに使用するデータアイテム、**アラーム ALTX(アラームテキスト)**の長さを設定します。

【構文】

public　static void set_altx_size(int size)

【引数】

size

設定したい ALTX のバイトサイズです。(固定)

size ＞ 0 であること。

【戻り値】

なし。

【説明】

通常は、ALTX の長さが４８バイトと決められてますが、例外的にこの長さを size で指定した長さに設定します。

S5F1 では、この設定値に合わせて ALTX が送信されます。

size は 1 以上でなければ設定されません。

４．１．３．15　get_altx_size() - ALTX の文字列サイズ取得

S5F1 などに使用するデータアイテム **ALTX**(Alarm Text)の長さを取得します。

【構文】

public　static int get_altx_size()

【引数】

なし。

【【戻り値】

返却値	意　味
現設定値	ALTX の長さの設定値

【説明】

ALTX の長さの設定値を取得します。

４．１．３．16　set_S1F13_send() - 通信確立方法の選択

GEM 通信接続開始時、Ｓ１F13 を相手装置から受信して応答をして通信確立とするかどうかを決めます。

【構文】

```
public  static void set_S1F13_send( int flag)
```

【引数】

flag

方法選択設定値

0　：当該装置、相手装置双方とも S1F13 を送信できます。

1　：相手装置から S1F13 を受信するまで待機します。

【戻り値】

なし。

【説明】

本メソッドは、classEnableComm クラスの Enable()メソッドを実行する前に設定してください。

通常は、当該装置または相手装置どちらからでも S1F13 を送信でき、それに対して、どちらかが S1f14 を応答することで通信確立とします。
ただし、相手装置が先に S1F13 を送信し、S1F14 応答を受信したときのみ通信確立とするケースがあります。
その対策として本メソッドが準備されています。

４．１．３．17　get_S1F13_send() - 通信確立の方法の取得

set_S1F13_send()で設定した通信確立の方法を取得します。

【構文】

```
public  static int get_S1F13_send()
```

【引数】

なし。

【【戻り値】

返却値	意　味
0	通常の方法（双方の装置が S1F13 を送信し、S1F14 の確認で通信確立）
1	相手装置から S1F13 を受信するまで待機します。

【説明】

set_S1F13_send()で設定した通信確立の方法を取得します。

４．１．３．18　set_EC_backup_flag() - EC 変数情報のバックアップ保存指定の設定

set_SV_backup_flag() - SV
set_DV_backup_flag() - DV
set_RP_backup_flag() - RP
set_CE_backup_flag() - CE
set_PP_backup_flag() - PP
set_FPP_backup_flag() - FPP
set_RCP_backup_flag() - RCP
set_CAR_backup_flag() - CAR
set_SUBST_backup_flag() - SUBST
set_PRJ_backup_flag() - PRJ
set_CJ_backup_flag() - CJ

【構文】

public　static int set_EC_backup_flag(int flag)
(他の変数についても同様の構文になります。)

【引数】

flag

バックアップファイル保存するかどうかを指示します。
0 ： バックアップ保存処理を行わない。
1 ：バックアップ保存処理を行います。

【戻り値】

なし。

【説明】

バックアップファイルに変数情報をバックアップするかどうかについて、個別変数単位で指定します。
メソッドは、各変数用に設けられています。

４．１．３．19　check_backup_all() - バックアップファイルの有効性の確認

現在残っている全変数のバックアップファイルが有効かどうかの確認をします。

【構文】

public static int check_backup_all(string dir, ref int vindex, ref string error)

【引数】

dir

バックアップファイルが保存されているディレクトリ

vindex

エラーを検出した変数に与えられたインデクス値の保存用
（インデクス値は、説明欄で示します。）

error

エラー発生した変数名が返されます。

【【戻り値】

返却値	意　味
0	全て有効であった。 （バックアップファイルが無い変数については正常とみなす）
＜ 0	エラーを検出した。 vindex に変数インデクス値、error にエラー内容が設定されます。

【説明】

dir で指定されたディレクトリに残っている全変数バックアップファイルについて、その有効性についてチェックします。
チェックは、各変数についてバックアップされている最新のファイルについて行います。

戻り値が =0 の場合は、存在している変数のバックアップファイルがすべて有効であることを意味します。
戻り値が =(-1)の場合は、vindex に指定された変数のバックアップファイルにエラーを検出したことを意味します。そして、error には変数の種類が返却されます。

vindex の値は、下表の変数を示します。

vindex 値	変数
0	EC
1	SV
2	DV
3	Report
4	CE
5	PP
6	FPP
7	Recipe
8	Carrier
9	Substrate
10	PRJ
11	CJ

４．１．３．20　check_EC_backup()　- EC 変数情報のバックアップファイルの有効性の確認

check_SV_backup() - SV
check_DV_backup() - DV
check_RP_backup() - RP
check_CE_backup() - CE
check_PP_backup() - PP
check_FPP_backup() - FPP
check_RCP_backup() - RCP
check_CAR_backup() - CAR
check_SUBST_backup() - SUBST
check_PRJ_backup() - PRJ
check_CJ_backup() - CJ

変数を個別にバックアップファイルが有効かどうかをチェックします。

【構文】

```
public  static int check_EC_backup ()
```

（他の変数についても同様の構文になります。）

【引数】

なし。

【【戻り値】

返却値	意　味
0	有効であった。 （バックアップファイルが無い変数については正常とみなす）
＜ 0	エラーを検出した。

【説明】

変数個別に残っている最新のバックアップファイルについて有効性の確認を行います。
メソッドは、変数個別に設けられています。

４．１．３．21　set_ev_handler()-SECS-II メッセージ受信用ハンドラーの設定

相手装置から送信されてくる SECS-II メッセージを APP 側で受け取るための受信ハンドラーを、通信エンジンに登録します。

【構文】

public static void set_ev_handler(class_CALLBACK.DshMsgPollEventHandler handler)

【引数】

handler

APP が受信メッセージを受けるためのイベントハンドラーになります。

【【戻り値】

なし。

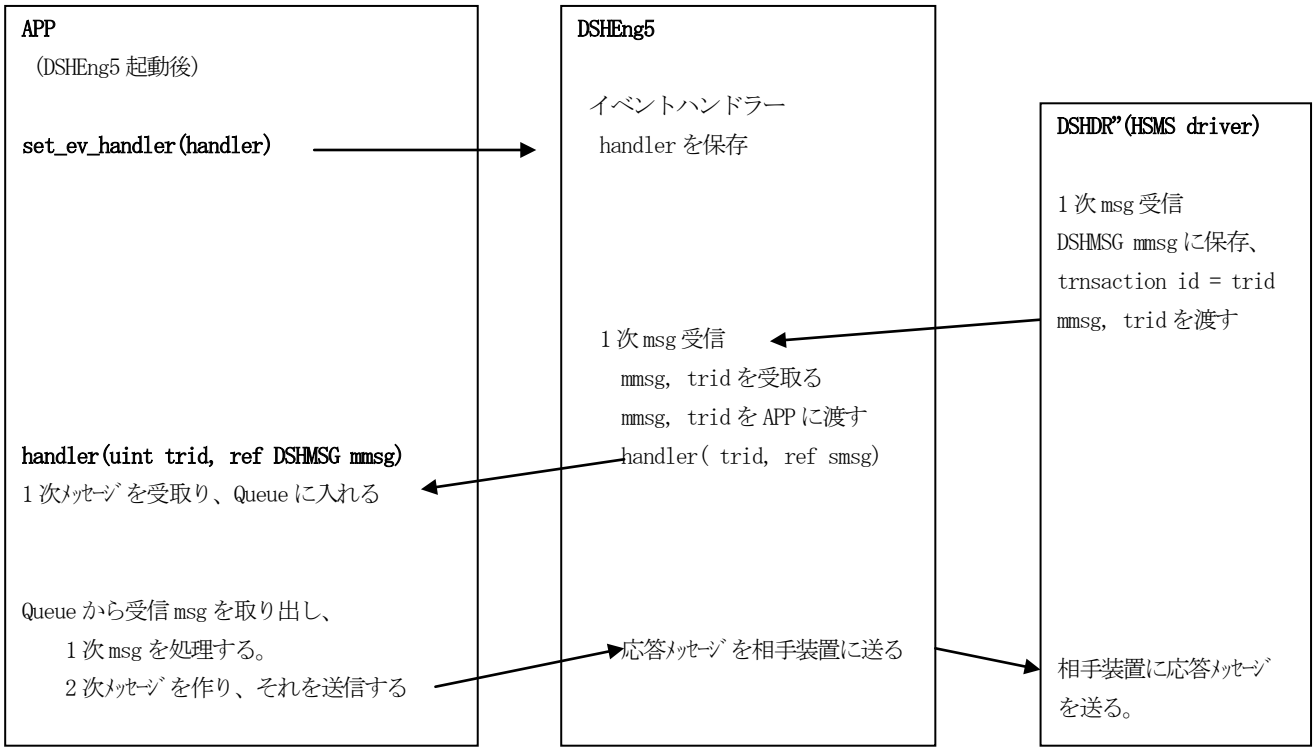
【説明】

本メソッドは、APP が相手装置から受信したメッセージを受け取るためのハンドラーを DSHEng5 に設定します。

APP は、DSHEng5 に対し、DSHEng5 が相手装置から受信したメッセージの中から APP が予め指定したメッセージだけを渡してもらうようにします。（後述する 4.3 class_TPRI_PASS クラス　を参照してください。）

メッセージを渡してもらう方法は、受信したメッセージをイベント通知ハンドラーを呼び出して渡してもらう方法です。

イベントハンドラーの役割は、概略以下のようになります。



本メソッドが、このイベント通知ハンドラーを DSHEng5 に設定するためのメソッドです。

イベント通知ハンドラーの構文は次の通りです。

［イベントハンドラーの構文と例］

```
static void poll_event_handler(uint trid, ref DSHMSG mmsg)
```

引数　　：　trid　－　通信トランザクション ID です。(DSHDR2 HSMS 通信ドライバーが発行した ID)
　　trid は、２次メッセージを応答する際に使用します。
　　mmsg　－　受信した１次メッセージ情報が保存されている構造体です。

なお、APP に渡すメッセージは、４．３で class_TPRI_PASS クラスの set_stream_func()メソッドで登録されたものが対象になります。

DSHEng5 が APP のハンドラーに受信メッセージを渡す方法は以下の通りです。

ハンドラー名	： poll_event_handler とする
HSMS ドライバーからのトランザクション ID	： uint trid － 応答メッセージ送信使用する。
メッセージ保存構造体	： DSHMSG smsg － この中にメッセージ本体がある。

```
public static class_CALLBACK.DshMsgPollEventHandler poll_event =
 new class_CALLBACK.DshMsgPollEventHandler(poll_event_handler);

static void poll_event_handler( uint trid, ref DSHMSG smsg); //  DSHEng5 の呼び出し（メッセージを渡す）
```

これによって、APP の poll_event_handler()に受信メッセージ情報が渡されます。

具体的な例については、DSHEng5 のデモプログラムの formMain.cs / vb ソースファイルを参照することができます。

４．２　class_EnableComm クラス － 通信 Enable / Disable

相手装置との GEM レベルでの通信確立(Enable)／確立解消 (Disable)を行うためのクラスです。
通信確立は、Enable()メソッドを使って行います。Enable()メソッドは、実行されると、スレッドを生成し、そのスレッドに通信確立処理を任せ、制御を APP の要求元に戻します。

スレッドによって通信確立が行われたら、Enable()メソッドの引数に与えられたコールバック関数の引数に確立処理の結果を引数にして、コールバック関数を呼び出し、APP に結果を通知します。

Enabe()による通信確立をキャンセルしたい場合は、CancelEnable()メソッドでキャンセルすることができます。

通信確立した後、その状態を解消する場合は、Disbale()メソッドを使用します。

通信確立状態は、SV(装置状態変数) の中の１つの予約変数に保存され、APP はその予約変数の値で GEM 通信確立しているかどうかを判断することができます。

なお、class_Enable クラスが APP に提供するメソッドは、すべて static メソッドです。

４．２．１　コンストラクタ

なし。

４．２．２　プロパティ

なし。

４．２．３　メソッド

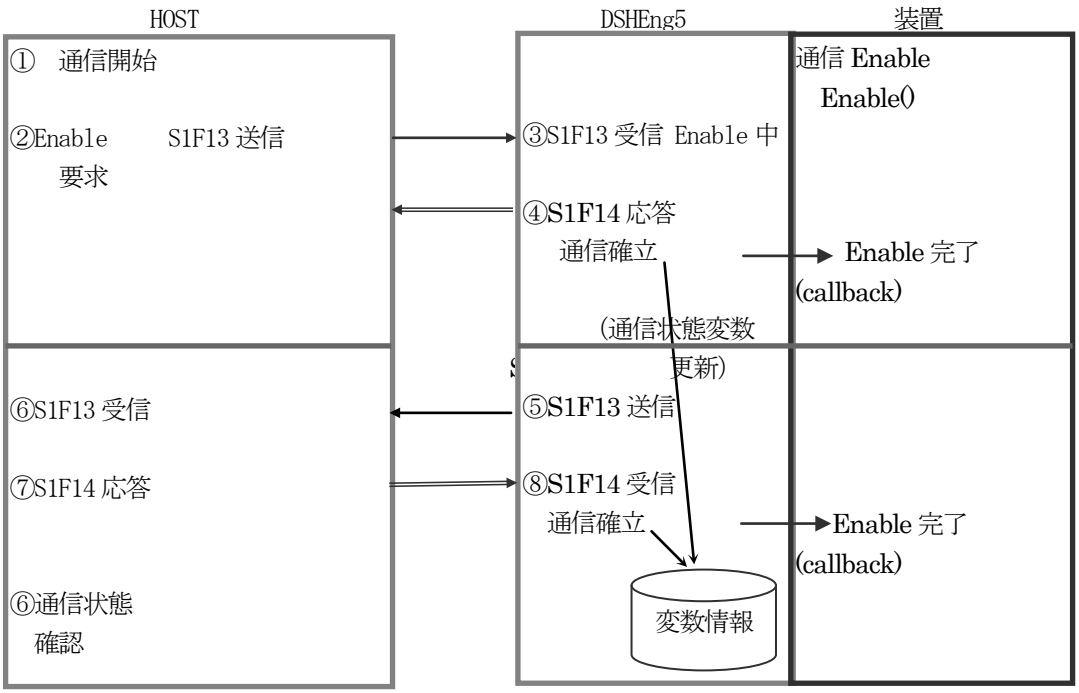
APP が使用できる classEnableComm クラスのメソッドは下記一覧表のとおりです。

	メソッド名	説明
1	public static int Enable()	GEM 通信確立の要求を行います。
2	pubic static void CancelEnable()	GEM 通信確立の要求（Enable）を取り消します。 （通信確立処理中のものを取り消します）
3	public static void_Disable()	GEM 通信確立状態を解消し、未確立状態にします。
4	public static int get_enable_busy_flag()	現在 Enable()が処理中かどうかを調べます。

４．２．３．１　Enabe()－通信確立要求

GEM 仕様の通信確立を要求します。

通信確立のための処理と状態の流れは下図の通りです。



注)　③の S1F13 はﾎｽﾄ主導、⑤の S1F13 は装置主導
どちらかの通信が成立すれば通信確立となる。

ホストと装置との間で、S1F13 とその応答 S1F14 の通信トランザクションが正常に実行されれば通信確立となります。
呼び出された後、Enable()メソッドは、通信未確立であれば、一旦、コントロールを APP に戻します。
そして、スレッドを生成し、そのスレッドが、通信を確立すべく次の処理を行います。

（1）S1F13 を送信し、S1F14 を受信し、その ack が 0 であれば、通信確立とします。
S1F13 に対し、無応答または、ack !=0 の S1F14 が受信された場合は、一定時間を置きます。そして、再び、処理を行います。

（2）相手装置からの S1F13 を受信しら、ack=0 の S1F14 を応答送信して、通信確立とします。

通信確立に成功した場合、装置状態変数(SV)の通信状態を　ST_COMMUNICATING (=5)に設定します。
その後、Enable()メソッドの引数 callback で示されるイベントハンドラーを呼出し、APP に通信確立したことを通知します。

なお、装置起動ファイル（equip.cnf or host.cf）内のコマンドに次のコマンドを設定することによって、
自身は S1F13 を発することなく、相手の装置からの S1F13 を受信し、S1F14 を送信することによって GEM 通信確立することができます。

```
S1F13_SEND        = 0   // 通信確立方法 -  0 =相手主導, 2=通常
```

【構文】

public static int Enable(class_CALLBACK.CallbackDefault callback, uint upara)

【引数】

callback

Enable 処理が終了した後、APP を呼び出すためのイベントハンドラーです。

upara

callback()の引数に付けるためのユーザパラメータ

【戻り値】

返却値	意　味
0	通信確立要求を受け付けた。
1	既に、通信確立済であった。
(-1)	既に Enable()実行中であった。

【説明】

冒頭で説明した内容の Enable 処理を行います。

４．２．３．２　CancelEnable() - Enable 通信確立処理の取り消し

先に Enable()メソッドで要求した通信確立処理の要求を取り消すために使用します。

【構文】

public static void CancelEnable()

【引数】

なし。

【戻り値】

なし。

【説明】

通信確立のための Enable()メソッドによる処理が実行中であれば、それを中止させます。

Enable()による通信確立処理は、中止した後 Enable()メソッド要求時に引数に指定された callback のイベントハンドラーを呼び出して通知します。

４．２．３．３　Disable()－通信確立の解消

もし、通信確立状態であれば、通信確立状態を解消し、装置状態変数(SV)の通信状態を ST_COMM_DISABLED (=0)に設定します。

もし、Enable()メソッドによる通信確立処理中であれば、その処理を中止します。

【構文】

```
public static void Disable()
```

【引数】

なし。

【戻り値】

なし。

【説明】

通信確立状態であれば、通信確立状態を解消し、装置状態変数(SV)の通信状態を ST_COMM_DISABLED (=0)に設定します。

通信確立のための Enable()メソッドによる処理が実行中であれば、それを中止させます。

４．２．３．４　get_enable_busy_flag()－Enable 処理状態取得

Enable()による通信確立処理中であるかどうかを調べます。

【構文】

```
public static int get_enable_busy_flag()
```

【引数】

なし。

[戻り値】

返却値	意　味
0	処理中ではない。
1	処理中である。

【説明】

Enable()による通信確立処理中であるかどうかを調べ、結果を返却します。

４．３　class_TPRI_PASS – APP へ渡す１次メッセージ登録用クラス

APP が、渡して欲しい受信した１次メッセージを登録するためのクタスです。

DSHEng5 は、本クラスの set_stream_function()メソッドによって登録された ID だけを APP に渡します。

４．１．３．21　で説明した **set_ev_handler()**メソッドによって与えられたイベントハンドラーを使って渡します。

４．３．１　コンストラクタ

なし。

４．３．２　プロパティ

なし。

４．３．３　メソッド

class_TPRI_PASS クラスのメソッドは下記一覧表のとおりです。

	メソッド名	説明
1	public static int set_stream_func()	１個のメッセージを追加登録します。
2	public static int get_stream_func()	登録されているすべてのメッセージ ID を取得します。

４．３．３．１　set_stream_func()-APP 処理１次メッセージ ID の登録

APP が処理したい 1 次メッセージの ID（stream, function）を登録します。

【構文】

public static int set_stream_func(int stream, int function)

【引数】

stream

SECS-II メッセージの Stream です。

function

SECS-II メッセージの Function です。

【戻り値】

返却値	意　味
0	正常に登録できた。
1	既に登録済であった。（エラーではありません）

【説明】

DSHEng5 通信エンジン起動直後、APP は、DSHEng5 が受信した 1 次メッセージの中で、APP 側で処理するメッセージ ID を登録します。

DSHEng5 は、1 次メッセージを受信した際、本メソッドによって登録されているメッセージであるかどうかを調べ、登録されていれば、**４．１．３．21　set_ev_handler()メソッド**で指定されたイベントハンドラーを使って受信したメッセージを APP に渡します。

４．３．３．２　get_stream_func()－APP 処理１次メッセージ ID 登録リストの取得

APP によって登録されている１次メッセージ ID（stream, function）のリストを取得します。

【構文】

public static int get_stream_func(int[] msgid_list, int max_count)

【引数】

msgid_list

　　取得したメッセージ ID を保存するリストです。

max_count

　　msgid_list リストの最大サイズです。

【戻り値】

返却値	意　味
>=0	登録されているメッセージの数

【説明】

登録されている APP に渡すメッセージ ID リストを取得します。

リスト要素の中には、以下のように stream, function の値が格納されます。

最上位バイト			最下位バイト
0	0	stream	function

５．DSHEng5 予約変数、オブジェクト関連コマンド、属性名関連クラス

５．１　class_const クラス － 変数、ＣＥ関連定数と予約変数

APP が使用できる定数の定義クラスです。

５．１．１　予約変数と参照インデクス

変数 ID の定義は、ユーザが定義します。その中の一部の変数を DSHEng5 の内部で使用します。これらの変数を予約変数と呼びます。　例えば、日付時刻を保存する SV 変数です。

DSHEng5 は、予約変数に与えられたインデクスを使って、それに対応する予約変数の ID を取得することができます。

設定は、**class_Reserved_V　クラス**のメソッドを使って行います。

５．１．１．１　EC 予約参照インデクス

（1）EC 予約変数のインデクスは下表の通りです。

ｲﾝﾃﾞｸｽ値	index	変数
0	ECX_RSV_MDLN	装置モデル名
1	ECX_RSV_SOFTREV	Sftware Revision
2	ECX_RSV_SPOOL_MAX	Spool できる最大値
3	ECX_RSV_SPOOL_OVERWRITE	Spool 上書き
4	ECX_RSV_INIT_COMMSTATE	通信状態の初期値
5	ECX_RSV_SPOOL_ENABLE	Spool Enable

（2）EC 予約変数 ID の設定は class_Reserved_V クラスの set_reserved_ECID()メソッドを使用します。
DSHEng5 を起動した後、APP が設定します。

構文

```
public static  int set_reserved_ECID(int ec_index, UInt32 ecid)
```

ec_index ： EC 予約参照インデクス
ecid　　： 設定したい ECID

（3）EC 予約変数 ID の取得は class_Reserved_V クラスの get_reserved_ECID()メソッドを使用します。

構文

```
public static int get_reserved_ECID(int ec_index, ref UInt32 ecid)
```

ec_index ： EC 予約参照インデクス
ecid　　： 取得 ECID 保存領域

５．１．１．２　SV 予約変数と参照インデクス

（1）SV 予約変数のインデクスは下表の通りです。

index 値	マクロ名	装置状態変数
0	SVX_RSV_CLOCK	ｼｽﾃﾑの日付時刻変数(DSHEng5 が値を更新する)
1	SVX_RSV_COMMUNICATING	通信状態
2	SVX_RSV_SPOOL_STATE	ｽﾌﾟｰﾙ状態
3	SVX_RSV_SPOOL_TOTAL	ｽﾌﾟｰﾙ合計
4	SVX_RSV_SPOOL_ACTUAL	実ｽﾌﾟｰﾙ数(貯えられた)
5	SVX_RSV_SPOOL_STIME	ｽﾌﾟｰﾙ開始時刻
6	SVX_RSV_SPOOL_FTIME	ｽﾌﾟｰﾙ満杯時刻
7	SVX_RSV_LIMIT_V	リミット監視対象変数 ID
8	SVX_RSV_LIMIT_DVVAL	同変数値(文字列)
9	SVX_RSV_LIMIT_ID	同リミット ID
10	SVX_RSV_LIMIT_DIR	同遷移方向

（2）SV 予約変数 ID の設定は class_Reserved_V クラスの set_reserved_SVID()メソッドを使用します。
DSHEng5 を起動した後、APP が設定します。

構文

```
public static  int set_reserved_SVID(int sv_index, UInt32 svid)
```

sv_index ： SV 予約参照インデクス
svid　　： 設定したい SVID

（3）SV 予約変数 ID の取得は class_Reserved_V クラスの get_reserved_SVID()メソッドを使用します。

構文

```
public static int get_reserved_SVID(int sv_index, ref UInt32 svid)
```

sv_index ： SV 予約参照インデクス
svid　　： 取得 SVID 保存領域

５．１．１．３　CE 予約参照インデクス

（1）CE 予約変数のインデクスは下表の通りです。

ｲﾝﾃﾞｸｽ値	index	変数
0	CEX_RSV_COMMUNICATING	通信確立時の S6F11 送信 CEID
1	CEX_RSV_SPOOL_END	Spool 終了通知用
2	CEX_RSV_LIMIT	LIMIT を超えたﾀｲﾐﾝｸﾞを通知する CEID

（2）CE 予約変数 ID の設定は class_Reserved_V クラスの set_reserved_CEID()メソッドを使用します。
DSHEng5 を起動した後、APP が設定します。

構文

```
public static  int set_reserved_CEID(int ce_index, UInt32 ceid)
```

ce_index ： CE 予約参照インデクス
ceid　　： 設定したい CEID

（3）CE 予約変数 ID の取得は class_Reserved_V クラスの get_reserved_CEID()メソッドを使用します。

構文

```
public static int get_reserved_CEID(int ce_index, ref UInt32 ceid)
```

ce_index ： CE 予約参照インデクス
ceid　　： 取得 CEID 保存領域

なお、index = CEX_RSV_LIMIT は、変数値（EC, SV, DV）が upperdb, lowerdb の限界を超えた際にホストに自動的に通知するイベント通知のための CEID を設定、取得するために使用します。
リミットイベント通知の仕組みについては、次の説明書を参照ください。

文書番号　DSHEng5-19-30310-00　「変数リミット監視機能　説明書」

５．１．１．４　RP 予約参照インデクス

（1）RP 予約変数のインデクスは下表の通りです。

ｲﾝﾃﾞｸｽ値	index	変数
0	RPX_RSV_LIMIT	LIMIT を超えたﾀｲﾐﾝｸﾞを通知する RPID

（2）RP 予約変数 ID の設定は class_Reserved_V クラスの set_reserved_RPID()メソッドを使用します。
DSHEng5 を起動した後、APP が設定します。

構文

```
public static  int set_reserved_RPID(int rp_index, UInt32 rpid)
```

rp_index ： RP 予約参照インデクス
rpid　　： 設定したい RPID

（3）RP 予約変数 ID の取得は class_Reserved_V クラスの get_reserved_RPID()メソッドを使用します。

構文

```
public static int get_reserved_RPID(int rp_index, ref UInt32 rpid)
```

rp_index ： RP 予約参照インデクス
rpid　　： 取得 RPID 保存領域

なお、index = RPX_RSV_LIMIT は、変数値（EC, SV, DV）が upperdb, lowerdb の限界を超えた際にホストに自動的に通知するイベント通知のための RPID を設定、取得するために使用します。

５．１．２　SV 通信状態の定数

GEM レベルの通信状態を表す定数について説明します。

状態定数	値	変数
ST_COMM_DISABLED	0	Disable 状態
ST_COMM_ENABLED	1	Enabled 状態
ST_NOT_COMMUNICATING	2	通信未確立状態
ST_WAIT_CRA	3	CRA 待ち状態
ST_WAIT_DELAY	4	時間待ち
ST_COMMUNICATING	5	通信確立状態

この値は、通信状態保存用 SV(装置状態)変数が有する値です。　ST_COMMUNICATING の状態は、相手装置とのメッセージ送受信ができる状態です。

通信状態保存用 SV 変数の ID は、**5.1.1.2 SV 予約変数と参照インデクス**の説明の中の、SVX_RSV_COMMUNICATING　インデクスを指定して、get_reserved_SVID()メソッドによって取得することができます。