DshGemMsgPro　GEM メッセージ・エンコード/デコード

ソフトウェア・ライブラリ

LIB 関数説明書

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

Vol-１／２

・変数(EC、SV、DVVAL)関連
・レポート、収集イベント(CE)関連
・アラーム関連
・プロセス・プログラム(PP、FPP)関連
・レシピ関連
・プロセス・ジョブ関連
・コントロール・ジョブ関連

２０１３年９月

株式会社データマップ

文書番号　DshGemMsgPro-13-50421-00

［取り扱い注意］

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

【改訂履歴】

番号	改訂日付	項 目	概 略
1.	2013 年 9 月	初版	

目　次

１．概要

本説明書は、SEMI GEM モデルに準拠する SECS-II メッセージのエンコード、デコード処理を行うために使用する DshGemMsgPro(以下、**GEM-PRO** と呼びます) ライブラリの付属ライブラリ関数について説明します。

これら付属ライブラリ関数は、ユーザが GEM-PRO を使って、メッセージのエンコード/デコード処理するために使用する構造体へのデータ情報の設定／取得のための使用されるものが主になります。

これらの関数は、GEM-PRO (DshGemMsgPro.dll) プログラムに含まれています。
各関数は、**Dsh**Xxxxx、**dsh_**xxxx のように名前の頭が Dsh または dsh_から始まる関数名になります。

本説明書では、関数を次の２つの種類に分けて説明します。

２章　エンコード/デコード API 関数の引数に使用される構造体の操作関連関数
３章　それ以外にユーザが使用する関数

１．１　関連ドキュメント

GEM-PRO に関する参照ドキュメントは以下の通りです。

GEM-PRO ドキュメント一覧表

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

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

１．２　C, C++言語の関数呼出し規約について

関数の説明の中で、関数のプロトタイプが、以下のように表現されていますが、これらの関数呼出し規約上の意味は次の通りです。

（1）**API**

#define API　　__declspec(dllexport)

__declspec　は、Microsoft 社固有のキーワードであり、関数名を DLL のエクスポートことを意味します。

（2）**APIX**

#define APIX　　__stdcall

__stdcall は、関数のスタック上への引数の渡し方を決めるキーワードです。

１．３　デモプログラムについて

GEM-PRO がサポートする全 GEM メッセージの Encode/Decode 機能を確認するためにデモプログラムが準備されています。

デモプログラムは、GEM-PRO　API 関数、LIB 関数の具体的なプログラミングの方法を具体的に理解する参考のために提供されます。

プログラム言語として、 C, C#, VB.Net の３種類のものが準備されています。

ユーザプログラミング用に提供される言語別のファイル名は以下の通りです。

	言語	API 関数	LIB 関数	定数、構造体定義
1	C	DshGemMsgProApi.h	DshGemMsgProLib.h	DshGemMsgProApi.h
2	c#	DshGemMsgProApi.cs	DshGemMsgProLib.cs	DshGemMsgProApi.cs
3	VB.Net	DshGemMsgProApi.vb	DshGemMsgProLib.vb	DshGemMsgProApi.vb

２．エンコード／デコードに使用する構造体の操作関数

GEM-PRO では、メッセージのエンコードを行う際、メッセージを構築するために必要な情報をそのメッセージのために用意された構造体内に詰め、そして、その構造体のポインタを API 関数の引数として渡すケースがあります。

本章では、メッセージを構築する際に使用する構造体内に、必要な情報を設定操作するために使用する関数について説明します。（これを GEM-PRO の **LIB 関数**と呼びます。）

.Net プログラム言語による場合、これらライブラリ関数が属する NameSpace と Class は以下のようになります。

名前空間　：DshGemPro
クラス名　：LIB

また、LIB 関数で使用する、定数、構造体のクラス名は、 **INFO** になります。

全 LIB 関数が static 関数になっています。したがって **LIB 関数のクラスのインスタンスの生成の必要はありません**。

LIB 関数の DshInitXXXX()を呼び出す場合は、**DshGemPro.LIB.DshInitXXXX()**のようにコーディングしてください。
（なお、 デモプログラムでは、Application の NameSpace は DshGemPro になっています。）

API 関数のインタフェース情報を構造体に設定する手順の中で使用する LIB 関数には、基本的に次の３種類のものがあり、次の順に実行します。

①構造体の初期設定を行う。	DshInitTXXXX_INFO()
②構造体のメンバーに情報を設定する。（複数回行うことがある）	DshPutTXXXX_INFO()
③構造体内で確保して使用したメモリを解放する。	DshFreeTXXXX_INFO()

表-2.1 LIB Encode/Decode 関連構造体操作関数一覧表

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTVID_LIST()	TVID_LIST 構造体の初期設定	S1F3, S2F13 (SV, EC)
	DshPutTVID_LIST()	同　変数値設定	
	DshFreeTVID_LIST()	同　内部使用ﾒﾓﾘの解放	
2	DshInitTV_VALUE_LIST()	TV_VALUE_LIST 構造体の初期設定	S1F4 (SV) S2F13 , S2F15 (EC)
	DshPutTV_VALUE_LIST()	同　変数値設定	
	DshFreeTV_VALUE_LIST()	同　内部使用ﾒﾓﾘの解放	
3	DshInitTSV_NAME_LIST(	TSV_NAME_LIST 構造体の初期設定	S1F12 (SV)
	DshPutTSV_NANE_LIST()	同　NAME 情報設定	
	DshFreeTSV_NAME_LIST()	同　内部使用ﾒﾓﾘの解放	
4	DshInitTTRACE_INFO()	TTRACE_INFO 構造体の初期設定	S2F23 (SV)
	DshPutTTRACE_INFO()	同　ﾄﾚｰｽ条件設定	
	DshFreeTTRACE_INFO()	同　内部使用ﾒﾓﾘの解放	
5	DshInitTEC_NAME_LIST()	TEC_NAMELIST 構造体の初期設定	S2F29 (EC)
	DshPutTEC_NAME_LIST()	同　NAME 情報設定	
	DshFreeTEC_NAME_LIST()	同　内部使用ﾒﾓﾘの解放	
6	DshInitTRP_LIST()	TRP_LIST 構造体の初期設定	S2F33 (REPORT)
	DshInitTRP_LINK	TRP_LINK 構造体の初期設定	
	DshPutTRP_LINK()	同　ﾘﾝｸ情報設定 (変数 ID)	
	DshFreeTRP_LIST()	同　内部使用ﾒﾓﾘの解放	
7	DshInitTCE_LIST()	TCE_LIST 構造体の初期設定	S2F35 (CE)
	DshPutTCE_LINK()	同　ﾘﾝｸ情報設定 (ﾚﾎﾟｰﾄ ID)	
	DshFreeTCE_LIST()	同　内部使用ﾒﾓﾘの解放	
8	DshInitTRCMD_INFO()	TRCMD_INFO 構造体の初期設定	S2F41
	DshPutTRCMD_INFO()	同　ﾎｽﾄｺﾏﾝﾄﾞ情報設定	
	DshFreeTRCMD_INFO()	同　内部使用ﾒﾓﾘの解放	
9	DshInitTRCMD_HERR_INFO()	TRCMD_HERR_INFO 構造体の初期設定	S2F42
	DshPutTRCMD_HERR_PARA()	同　ｴﾗｰﾊﾟﾗﾒｰﾀ設定	
	DshFreeTRCMD_HERR_INFO()	同　内部使用ﾒﾓﾘの解放	
10	DshInitTSPOOL_INFO()	TSPOOL_INFO 構造体の初期設定	S2F43 (SPOOL)
	DshPutTSPOOL_INFO()	同 stream を 1 個設定	
	DshFreeTSPOOL_INFO()	同 内部使用ﾒﾓﾘの解放	
	DshInitTSTRE_INFO()	TSTRE_INFO 構造体の初期設定	
	DshPutTSTRE_INFO()	同 function を 1 個設定	
	DshFreeTSTRE_INFO()	同 内部使用ﾒﾓﾘの解放	
11	DshInitTSPOOL_ERR_INFO()	TSPOOL_ERR_INFO 構造体の初期設定	S2F44
	DshPutTSPOOL_ERR_INFO()	同 stream1 個分の設定	
	DshFreeTSPOOL_ERR_INFO()	同 内部使用ﾒﾓﾘの解放	
	DshInitTSTRE_ERR_INFO()	TSTRE_ERR_INFO 構造体の初期設定	
	DshPutTSTRE_ERR_INFO()	同 function 1 個分の設定	
	DshFreeTSTRE_ERR_INFO()	同 内部使用ﾒﾓﾘの解放	

12	DshInitTLIMIT_LIST()	TLIMIT_LIST 構造体の初期設定	S2F45 (LIMIT)
	DshPutTLIMIT_LIST()	同　1 個の TLIMIT_INFO の設定	
	DshFreeTLIMIT_LIST()	同 内部使用ﾒﾓﾘの解放	
	DshInitTLIMIT_INFO()	TLIMIT_INFO 構造体の初期設定	
	DshPutTLIMIT_INFO()	同　ﾘﾐｯﾄ値の設定	
	DshFreeTLIMIT_INFO()	同 内部使用ﾒﾓﾘの解放	
13	DshInitTLIMIT_RSP_LIST()	TLIMIT_RSP_LIST 構造体の初期設定	S2F48 (LIMIT)
	DshPutTLIMIT_RSP_LIST()	同　TLIMIT_RSP_NFO の設定	
	DshFreeTLIMIT_RSP_LIST()	同 内部使用ﾒﾓﾘの解放	
	DshInitTLIMIT_RSP_INFO()	TLIMIT_RSP_INFO 構造体の初期設定	
	DshPutTLIMIT_RSP_INFO()	同　ﾘﾐｯﾄ値の設定	
	DshFreeTLIMIT_RSP_INFO()	同 内部使用ﾒﾓﾘの解放	
14	DshInitTERCMD_INFO()	TERCMD_INFO 構造体の初期設定	S2F49 (ERC)
	DshPutTERCMD_INFO()	同　拡張ﾘﾓｰﾄｺﾏﾝﾄﾞ情報設定	
	DshFreeTERCMD_INFO()	同　内部使用ﾒﾓﾘの解放	
15	DshInitTERCMD_HERR_INFO()	TERCMD_HERR_INFO 構造体の初期設定	S2F50
	DshPutTERCMD_HERR_PARA()	同　ｴﾗｰﾊﾟﾗﾒｰﾀ設定	
	DshFreeTERCMD_HERR_INFO()	同　内部使用ﾒﾓﾘの解放	
16	DshInitTCACT_INFO()	TCACT_INFO 構造体の初期設定	S3F17
	DshPutTCACT_INFO()	同　1 個の TCACT_PARA の設定	
	DshFreeTCACT_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTCACT_PARA()	TCACT_PARA 構造体の初期設定	
	DshPutTCACT_CONTENT()	ｷｬﾘｱｺﾝﾃﾝﾄの設定　(slot 情報)	
	DshFreeTCACT_PARA()	同　内部使用ﾒﾓﾘの解放	
17	DshInitTCACT_ERR_INFO()	TCACT_ERR_INFO 構造体の初期設定	S3F18
	DshPutTCACT_ERR_INFO()	同　1 個のｴﾗｰ情報を設定	S3F24
	DshFreeTCACT_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	S3F26
18	DshInitTPORTG_INFO()	TPORTG_INFO 構造体の初期設定	S3F23
	DshPutTPORG_INFO()	同　1 個のﾎﾟｰﾄｸﾞﾙｰﾌﾟの設定	
	DshFreeTPORTG_INFO()	同　内部使用ﾒﾓﾘの解放	
19	DshInitTPORT_INFO()	TPORT_INFO 構造体の初期設定	S3F25
	DshPutTPORT_INFO()	同　1 個のﾎﾟｰﾄの設定	
	DshFreeTPORT_INFO()	同　内部使用ﾒﾓﾘの解放	
20	DshInitTACCESS_INFO()	TACCESS_INFO 構造体の初期設定	S3F27
	DshPutTACCESS_INFO()	同　1 個のﾎﾟｰﾄの設定	
	DshFreeTACCESS_INFO()	同　内部使用ﾒﾓﾘの解放	
21	DshInitTACCESS_ERR_INFO()	TACCESS_ERR_INFO 構造体の初期設定	S3F28
	DshPutTACCESS_ERR_INFO()	同　1 個のｴﾗｰ情報を設定	
	DshFreeTACCESS_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	
22	DshInitTAL_S5F1_INFO()	TAL_S5F1_INFO 構造体の初期設定	S5F1
	DshFreeTAL_S5F1_INFO()	同　内部使用ﾒﾓﾘの解放	S5F6
23	DshInitTAL_S5F6_LIST()	TAL_S5F6_LIST 構造体の初期設定	S5F6
	DshPutTAL_S5F6_LIST()	同 TAL_S5F1_INFO を設定	
	DshFreeTAL_S5F6_LIST()	同　内部使用ﾒﾓﾘの解放	

24	DshInitTTRACE_DATA()	TTRACE_DATA 構造体の初期設定	S6F1
	DshPutTTRACE_DATA()	同　1 個の TTRACE_SV(ﾄﾚｰｽﾃﾞｰﾀ) の設定	
	DshFreeTTRACE_DATA()	同　内部使用ﾒﾓﾘの解放	
	DshInitTTRACE_SV()	TTRACE_SV 構造体の初期設定	
	DshFreeTTRACE_SV()	同　内部使用ﾒﾓﾘの解放	
25	DshInitTS6F11_CE_INFO()	TS6F11_CE_INFO 構造体の初期設定	S6F11 S6F16 S6F20
	DshPutTS6F11_CE_INFO()	同　1 個のﾚﾎﾟｰﾄ情報の設定	
	DshFreeTS6F11_CE_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTS6F11_RP_INFO()	TS6F11_RP_INFO 構造体の初期設定	
	DshPutTS6F11_RP_INFO()	同 1 個の変数値を設定	
	DshFreeTS6F11_RP_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTS6F11_V_INFO()	TS6F11_V_INFO の構造体の初期設定	
	DshFreeTS6F11_V_INFO()	同　内部使用ﾒﾓﾘの解放	
26	DshInitTPPID_LIST()	TPPID_LIST 構造体の初期設定	S7F17 S7F19
	DshPutTPPID_LIST()	同　PPID を 1 個設定	
	DshFreeTPPID_LIST()	同　内部使用ﾒﾓﾘの解放	
27	DshInitTS7F23_INFO()	TS7F23_INFO 構造体の初期設定	S7F23 S7F26
	DshPutTS7F23_INFO()	同　TFPP_CCODE を 1 個設定	
	DshFreeTS7F23_INFO	同　内部使用ﾒﾓﾘの解放	
	DshInitTFPP_CCODE()	TFPP_CCODE 構造体の初期設定	
	DshPutTFPP_CCODE()	同　ﾊﾟﾗﾒｰﾀを 1 個設定	
	DshFreeTFPP_CCODE()	同　内部使用ﾒﾓﾘの解放	
28	DshInitTPP_PVS_LIST()	TPP_PVS_LIST 構造体の初期設定	S7F27
	DshPutTPP_PVS_LIST()	同　妥当性情報を設定	
	DshFreeTPP_PVS_LIST()	同　内部使用ﾒﾓﾘの解放	
29	DshInitTTERMTEXT_INFO()	TTERMTEXT_INFO 構造体の初期設定	S10F5
	DshPutTTERMTEXT_INFO()	同　文字列を設定	
	DshFreeTTERMTEXT_INFO()	同　内部使用ﾒﾓﾘの解放	

30	DshInitTCJ_INFO()	TCJ_INFO 構造体の初期設定	S14F9
	DshPutTCJ_ATTR_INFO()	同　属性情報を設定	S14F11
	DshFreeTCJ_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTCJ_TEXT_INFO()	TCJ_TEXT_INFO 構造体の初期設定	
	DshPutTCJ_TEXT_INFO()	同　文字列を設定	
	DshFreeTCJ_TEXT_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitVOID_LIST()	TVOID_LIST 構造体の初期設定	
	DshPutVOID_LIST()	同　TVOID_LIST に属性情報構造体を設定	
	DshInitTMTRL_OUT_STAT()	TMTRL_OUT_STAT 構造体の初期設定	
	DshPutTMTRL_OUT_STAT()	同　status を 1 個設定	
	DshFreeTVOID_LIST_TMTRL_OUT_STAT()	同　内部使用ﾒﾓﾘの解放	
	DshInitTMTRL_OUT_SPEC()	TMTRL_OUT_SPEC 構造体の初期設定	
	DshPutTMTRL_OUT_SPECSrc	同 source ｷｬﾘｱの slotid を設定	
	DshPutTMTRL_OUT_SPECDst	同 destination ｷｬﾘｱの slotid を設定	
	DshFreeTVOID_LIST_TMTRL_OUT_SPEC()	同　内部使用ﾒﾓﾘの解放	
	DshInitTCTRL_SPEC()	TCTRL_SPEC 構造体の初期設定	
	DshPutTCTRL_RULE()	同　ｺﾝﾄﾛｰﾙ・ﾙｰﾙ情報を設定	
	DshPutTOUT_RULE()	同　ｺﾝﾄﾛｰﾙ・ｽﾍﾟｯｸ情報を設定	
	DshFreeTVOID_LIST_TCTRL_SPEC()	同　内部使用ﾒﾓﾘの解放	
	DshInitTPAUSE_EVENT()	TPAUSE_EVENT 構造体の初期設定	
	DshPutTPAUSE_EVENT()	同　ｲﾍﾞﾝﾄ ID を設定	
	DshFreeTPAUSE_EVENT()	同　内部使用ﾒﾓﾘの解放	
	DshInitTPRJ_STATE_LIST()	TPRJ_STATE_LIST 構造体の初期設定	
	DshPutTPRJ_STATE_LIST()	同　process job status を設定	
	DshFreeTPRJ_STATE_LIST()	同　内部使用ﾒﾓﾘの解放	
31	DshInitTRCP_ACT_INFO()	TRCP_ACT_INFO 構造体の初期設定	S15F3
	DshFreeTRCP_ACT_INFO()	同　内部使用ﾒﾓﾘの解放	
32	DshInitTRCP_RENAME_INFO()	TRCP_RENAME_INFO 構造体の初期設定	S15F5
	DshFreeTRCP_RENAME_INFO()	同　内部使用ﾒﾓﾘの解放	
33	DshInitTRCP_S15F8_INFO()	TRCP_S15F8_INFO 構造体の初期設定	S15F8
	DshPutTRCP_S15F8_INFO()	同　ｴﾗｰ情報の設定	
	DshFreeTRCP_S15F8_INFO()	同　内部使用ﾒﾓﾘの解放	
34	DshInitTRCP_S15F10_INFO()	TRCP_S15F10_INFO 構造体の初期設定	S15F10
	DshPutTRCP_S15F10_INFO()	同　ｴﾗｰ情報の設定	
	DshFreeTRCP_S15F10_INFO()	同　内部使用ﾒﾓﾘの解放	
35	DshInitTRCP_INFO()	TRCP_INFO 構造体の初期設定	S15F13
	DshPutTRCP_PARA()	同　ﾊﾟﾗﾒｰﾀ情報の設定	S16F11
	DshFreeTRCP_INFO()	同　内部使用ﾒﾓﾘの解放	S16F15
36	DshInitTRCP_ERR_INFO()	TRCP_NFO 構造体の初期設定	S15F14
	DshPutTRCP_ERR_INFO()	同　ｴﾗｰ情報の設定	
	DshFreeTRCP_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	
37	DshInitTRCP_RETRIEVE_INFO()	TRCP_RETRIEVE_INFO 構造体の初期設定	S15F17
	DshFreeTRCP_RETRIEVE_INFO()	同　内部使用ﾒﾓﾘの解放	

38	DshInitTRCP_S15F18_INFO()	TRCP_S15F18_INFO 構造体の初期設定	S15F18
	DshPutTRCP_S15F18_M_SECNM_INFO()	同　ｾｸｼｮﾝ情報を設定	
	DshPutTRCP_M_SECNM_ATTR()	同　ｾｸｼｮﾝ属性情報を設定	
	DshPutTRCP_S15F18_ERR()	同　ｴﾗｰ情報の設定	
	DshFreeTRCP_S15F18_INFO()	同　内部使用ﾒﾓﾘの解放	
39	DshInitTPRJ_CMD_INFO()	TPRJ_CMD_INFO 構造体の初期設定	S16F5
	DshPutTPRJ_CMD_INFO()	同　ｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ情報の設定	
	DshFreeTPRJ_CMD_INFO()	同　内部使用ﾒﾓﾘの解放	
40	DshInitTPRJ_CMD_ERR_INFO()	TPRJ_CMD_ERR_INFO 構造体の初期設定	S16F6
	DshPutTPRJ_CMD_ERR_INFO()	同　ｴﾗｰ情報の設定	
	DshFreeTPRJ_CMD_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	
41	DshInitTPRJ_INFO()	TPRJ_INFO 構造体の初期設定	S16F11 S16F15
	DshPutPrjRcpInfo()	同　ﾚｼﾋﾟ情報の設定	
	DshPutPrjCarInfo()	同　ｷｬﾘｱ情報の設定	
	DshPutPrjMid()	同　MID の設定	
	DshPutPrjPauseCeid()	同　PAUSE 時 CEID 設定	
	DshFreeTPRJ_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTCAR_INFO()	TCAR_INFO 構造体の初期設定	
	DshFreeTCAR_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTCAR_SLOT_INFO()	TCAR_SLOT_INFO 構造体の初期設定	
	DshPutTCAR_SLOT_INFO()	同　ｽﾛｯﾄ ID 設定	
	DshFreeTSLOT_INFO()	同　内部使用ﾒﾓﾘの解放	
42	DshInitTPRJ_LIST()	TPRJ_LIST 構造体の初期設定	S16F15
	DshPutTPRJ_LIST()	同　TPRJ_INFO を設定	
	DshFreeTPRJ_INFO()	同　内部使用ﾒﾓﾘの解放	
43	DshInitTPRJ_ERR_INFO()	TPRJ_ERR_INFO 構造体の初期設定	S16F12 S16F16 S16F18
	DshPutTPRJ_ERR_INFO()	同　ｴﾗｰ情報の設定	
	DshFreeTPRJ_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshPutTPRJ_ERR_PRJID()	同　ｴﾗｰ対象 PRJID 設定(S16F15 のみ)	
44	DshInitTPRJ_DEQ_INFO()	TPRJ_DEQ_INFO 構造体の初期設定	S16F17
	DshPutTPRJ_DEQ_INFO()	同　PRJID を設定	
	DshFreeTPRJ_DEQ_INFO()	同　内部使用ﾒﾓﾘの解放	
45	DshInitTPRJ_STATE_LIST()	TPRJ_STATE_LIST 構造体の初期設定	S1620
	DshPutTPRJ_STATE_LIST()	同　PRJID と state を設定	
	DshFreeTPRJ_STATE_LIST()	同　内部使用ﾒﾓﾘの解放	
46	DshInitTCJ_CMD_INFO()	TCJ_CMD_INFO 構造体の初期設定	S16F27
	DshPutTCJ_CMD_INFO()	同　ｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ情報の設定	
	DshFreeTCJ_CMD_INFO()	同　内部使用ﾒﾓﾘの解放	
47	DshInitTCJ_CMD_ERR_INFO()	TCJ_CMD_ERR_INFO 構造体の初期設定	S16F28
	DshFreeTCJ_CMD_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	
48	DshInitTCAR_INFO()	TCAR_INFO 構造体の初期設定	S16F11 S16F15
	DshFreeTCAR_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTCAR_SLOT_INFO()	TCAR_SLOT_INFO 構造体の初期設定	
	DshPutTCAR_SLOT_INFO()	同　ｽﾛｯﾄ ID 設定	
	DshFreeTSLOT_INFO()	同　内部使用ﾒﾓﾘの解放	

表-2.2 LIB その他関数一覧表

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

２．１　変数（EC,SV,DVVAL）関連関数

以下の関数があります。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTVID_LIST()	TVID_LIST 構造体の初期設定	S1F3, S2F13
	DshPutTVID_LIST()	同　変数値設定	
	DshFreeTVID_LIST()	同　内部使用ﾒﾓﾘの解放	
2	DshInitTV_VALUE_LIST()	TV_VALUE_LIST 構造体の初期設定	S1F4 (SV) S2F14, S2F15 (EC)
	DshPutTV_VALUE_LIST()	同　変数値設定	
	DshFreeTV_VALUE_LIST()	同　内部使用ﾒﾓﾘの解放	
3	DshInitTSV_NAME_LIST(	TSV_NAME_LIST 構造体の初期設定	S1F12 (SV)
	DshPutTSV_NANE_LIST()	同　NAME 情報設定	
	DshFreeTSV_NAME_LIST()	同　内部使用ﾒﾓﾘの解放	
4	DshInitTTRACE_INFO()	TTRACE_INFO 構造体の初期設定	S2F23 (SV)
	DshPutTTRACE_INFO()	同　ﾄﾚｰｽ条件設定	
	DshFreeTTRACE_INFO()	同　内部使用ﾒﾓﾘの解放	
5	DshInitTEC_NAME_LIST()	TEC_NAMELIST 構造体の初期設定	S2F29 (EC)
	DshPutTEC_NAME_LIST()	同　NAME 情報設定	
	DshFreeTEC_NAME_LIST()	同　内部使用ﾒﾓﾘの解放	
6	DshInitTLIMIT_LIST	TLIMIT_LIST 構造体の初期設定	S2F45 (LIMIT)
	DshPutTLIMIT_LIST	同　1 個の TLIMIT_INFO の設定	
	DshFreeTLIMIT_LIST	同 内部使用ﾒﾓﾘの解放	
	DshInitTLIMIT_INFO	TLIMIT_INFO 構造体の初期設定	
	DshPutTLIMIT_INFO	同　ﾘﾐｯﾄ値の設定	
	DshFreeTLIMIT_INFO	同 内部使用ﾒﾓﾘの解放	
	DshInitTLIMIT_RSP_LIST	TLIMIT_RSP_LIST 構造体の初期設定	S2F48 (LIMIT)
	DshPutTLIMIT_RSP_LIST	同　TLIMIT_RSP_NFO の設定	
	DshFreeTLIMIT_RSP_LIST	同 内部使用ﾒﾓﾘの解放	
	DshInitTLIMIT_RSP_INFO	TLIMIT_RSP_INFO 構造体の初期設定	
	DshPutTLIMIT_RSP_INFO	同　ﾘﾐｯﾄ値の設定	
	DshFreeTLIMIT_RSP_INFO	同 内部使用ﾒﾓﾘの解放	

２．１．１　DshInitTVID_LIST () – 変数 ID リスト構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTVID_LIST (
      TVID_LIST    *list,          // TVID_LIST 変数 ID 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
      int           count          // list に含む VID の数
);
```

[VB.Net]

```
Sub DshInitTVID_LIST (
        ByRef list As TVID_LIST,
        count      As Integer  )
```

[C#]

```
void DshInitTVID_LIST(
        ref TVID_LIST list,
        int       count );
```

（2）引数

list

TVID_LIST 構造体のポインタです。

count

list に格納する変数 ID の数です。(VID の数)

（3）戻り値

なし。

（4）説明

本関数は、TVID_LIST 構造体を使用するメッセージ、S1F3、 S2F13 のメッセージを作るときに使用することができます。構造体内に含める情報は、複数の変数 ID になります。

DshInitTVID_LIST()関数は TVID_LIST 構造体を初期化するための関数です。

TVID_LIST 構造体に count 分の変数 ID を格納するための初期設定処理を行います。

個々の変数 ID の設定には、DshPutTVID_LIST()関数を使用します。

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

（5）構造体

```
typedef struct{
        int         count;
        int         max_count;
        TVID        *list;
} TVID_LIST;
```

２．１．２　DshPutTVID_LIST () – 置変数 ID の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTVID_LIST (
      TVID_LIST *list,          // TVID_LIST 変数 ID ﾘｽﾄ格納構造体のﾎﾟｲﾝﾀ
      TVID          vid         // 追加する変数の ID
);
```

[VB.Net]

```
Function DshPutTVID_LIST (
      ByRef   list As TVID_LIST,
      vid As    Integer) As Integer
```

[C#]

```
int DshPutTVID_LIST(
      ref TVID_LIST list,
      uint    vid);
```

（2）引数

list

TVID_LIST 構造体のポインタです。

vid

list 内に追加する変数 ID です。

（3）戻り値

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

（4）説明

先に説明した DshInitTVID_LIST()関数で初期設定された list 内に 1 個の変数 ID を追加します。
追加によって、本関数が呼び出される順番に ID が構造体内の list 配列内に保存されます。

DshInitTVID_LIST()関数で設定した max_count 分だけの変数 ID を加えることがします。

max_count 分を超える数の変数情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．１．３　DshFreeTVID_LIST() – 変数 ID リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTVID_LIST(
        TVID_LIST *list              // ﾒﾓﾘを開放したい変数 IDﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTVID_LIST (
        ByRef list  As TVID_LIST)
```

[C#]

```
void DshFreeTVID_LIST(
        ref TVID_LIST  list );
```

（2）引数

list

メモリを解放したい変数 ID リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TVID_LIST 構造体内で変数 ID リストに使用されているメモリを全て解放します。
開放した後、TVID_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

２．１．４　DshInitTV_VALUE_LIST () – 変数情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTV_VALUE_LIST (
        TV_VALUE_LIST *list,          // TV_VALUE_LIST 変数値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int           count                   // list に含むVID(=ECID)の数
);
```

[VB.Net]

```
Sub DshInitTV_VALUE_LIST (
        ByRef list As TV_VALUE_LIST,
        count      As Integer  )
```

[C#]

```
void DshInitTV_VALUE_LIST(
        ref TV_VALUE_LIST list,
        int       count );
```

（2）引数

list

TV_VALUE_LIST 構造体のポインタです。

count

list に格納する変数情報の数です。(VID の数)

（3）戻り値

なし。

（4）説明

本関数は、TV_VALUE_LIST 構造体を使用するメッセージ、S1F4、 S2F14 ならびに S2F15 のメッセージを作るときに使用します。構造体内に含める情報は、複数の変数の ID とその値になります。

DshInitTV_VALUE_LIST()関数は TV_VALUE_LIST 構造体を初期化するための関数です。
TV_VALUE_COUNT に count 分の変数値情報を格納するための初期設定処理を行います。

変数値の設定には、DshPutTV_VALUE_LIST()関数を使用します。
構造体の使用が済んだら DshFreeTV_VALUE_LIST()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        int         count;
        TV_VALUE    **vv_list;
} TV_VALUE_LIST;
```

```
typedef struct{
        TVID        vid;
        int         format;
        int         asize;
        void        *value;
} TV_VALUE;
```

２．１．５　DshPutTV_VALUE_LIST () – 装置変数情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTV_VALUE_LIST (
        TV_VALUE_LIST *list,        // TV_VALUE_LIST 変数値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        TVID          vid,          // 追加する変数の ID
        int           fmt,          // 変数ﾃﾞｰﾀのﾌｫｰﾏｯﾄ
        int           asize,        // 変数ﾃﾞｰﾀの配列ｻｲｽﾞ
        void          *value        // 変数値が格納されている領域のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Function DshPutTV_VALUE_LIST (
        ByRef   list As TV_VALUE_LIST,
        vid As   Integer,
        fmt As   Integer,
        asize As Integer,
        value As Integer) As Integer
```

[C#]

```
int DshPutTV_VALUE_LIST(
        ref TV_VALUE_LIST list,
        uint   vid,
        int    fmt,
        int    asize,
        IntPtr value );
```

（2）引数

list

TV_VALUE_LIST 構造体のポインタです。

vid

list 内に追加する変数 ID です。

fmt

変数のデータフォーマットです。

asize

変数データの配列サイズです。

value

変数値が格納されている領域のポインタです。

（3）戻り値

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

（4）説明

先に説明した DshInitTV_VALUE_LIST()関数で初期設定された list 内に 1 個の変数データ情報を追加します。

追加する情報は引数に指定されている変数 ID とデータ値です。

追加によって、本関数が呼び出される順番に ID と値が構造体内の list 配列内に保存されます。

DshInitTV_VALUE_LIST()関数で設定した count 分だけの変数情報を加えることがします。
（count で指定した分の変数情報を設定してください。）

count 分を超える数の変数情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．１．６　DshFreeTV_VALUE_LIST() – 変数情報リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTV_VALUE_LIST(
        TV_VALUE_LIST *list              // ﾒﾓﾘを開放したい変数情報ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTV_VALUE_LIST (
        ByRef list  As TV_VALUE_LIST)
```

[C#]

```
void DshFreeTV_VALUE_LIST(
        ref TV_VALUE_LIST  list );
```

（2）引数

list

メモリを解放したい変数情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TV_VALUE_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TV_VALUE_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

２．１．７　DshInitTSV_NAME_LIST () – SV　装置状態変数名リスト構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTV_VALUE_LIST (
        TSV_NAME_LIST *list,          // TSV_NAME_LIST 変数値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int           count           // list に含むVID(=ECID)の数
);
```

[VB.Net]

```
Sub DshInitTSV_NAME_LIST (
        ByRef  list  As TSV_NAME_LIST,
        count  As Integer)
```

[C#]

```
void DshInitTSV_NAME_LIST(
        ref   TSV_NAME_LIST  list,
        int   count );
```

（2）引数

list

TSV_NAME_LIST 構造体のポインタです。

count

list に格納する変数情報の数です。(VID の数)

（3）戻り値

なし。

（4）説明

本関数は、TSV_NAME_LIST 構造体を使用するメッセージ、S1F12 メッセージを作るときに使用します。構造体内に含める情報は、変数の数と、各変数の名前と物理単位(Units)です。
list で指定された TSV_NAME_LIST 構造体内に、count 分の名前情報を保存できるように初期設定します。

名前情報を設定するためには、DshPutTSV_NAME_LIST()関数を使用します。

TSV_NAME_LIST 構造体の使用が済んだら、DshFreeTSV_NAME_LIST()関数を使って、構造体内で使用されたメモリを解放してください。

（5）構造体

```
typedef struct{
        int         count;
        TSV_NAME    **name_list;
} TSV_NAME_LIST;
```

```
typedef struct{
        TSVID       svid;
        char        *name;
        char        *units;
} TSV_NAME;
```

２．１．８　DshPutTSV_NAME_LIST () – SV 変数名の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTSV_NAME_LIST (
        TSV_NAME_LIST *list,        // TSV_NAME_LIST 変数名情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        TVID          vid,          // 追加する変数の ID
        char          *name,        // SV の名前
        char          *units        // 単位
);
```

[VB.Net]

```
Function DshPutTSV_NAME_LIST (
        ByRef  list As TSV_NAME_LIST,
        vid    As  Integer,
        name   As  String,
        units  As  String) As Integer
```

[C#]

```
int DshPutTSV_NAME_LIST(
        ref TSV_NAME_LIST  list,
        uint    vid,
        string  name,
        string  units);
```

（2）引数

list

TSV_NAME_LIST 構造体のポインタです。

vid

list 内に追加する変数 ID です。

name

変数名です。

units

単位名です。

（3）戻り値

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

（4）説明

DshInitTSV_NAME_LIST()関数で初期設定された list 内に 1 個の変数データ情報を追加します。
追加する情報は引数に指定されている ID、変数名と単位情報です。

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

count 分を超える数の変数情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．１．９　DshFreeTSV_NAME_LIST() – SV 装置状態変数名リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTSV_NAME_LIST(
        TSV_NAME_LIST     *list              // ﾒﾓﾘを開放したい変数情報ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTSV_NAME_LIST (
        ByRef  list As TSV_NAME_LIST)
```

[C#]

```
void DshFreeTSV_NAME_LIST(
        ref TSV_NAME_LIST  list );
```

（2）引数

list

メモリを解放したい変数情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TSV_NAME_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TSV_NAME_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

２．１．10　DshInitTTRACE_INFO() – トレース S2F43 用情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTTRACE_INFO(
        TTRACE_INFO *info,                  // ﾄﾚｰｽ情報構造体の格納ﾎﾟｲﾝﾀ
        char    *trid,                      // ﾄﾚｰｽ ID
        char    *dsper,                     // ﾃﾞｰﾀ収集時間ﾃﾞｰﾀ（文字列）
        int     totsmp,                     // 総ｻﾝﾌﾟﾙ数
        int     repgsz,                     // ﾚﾎﾟｰﾄｸﾞﾙｰﾌﾟのｻｲｽﾞ
        int     svid_count                  // ﾄﾚｰｽ対象とする SV の数
);
```

[VB.Net]

```
Function DshInitTTRACE_INFO (
        ByRef info As TTRACE_INFO,
        trid   As String,
        dsper  As String,
        totsmp As Integer,
        repgsz As Integer,
        svid_count As Integer) As Integer
```

[C#]

```
int DshInitTTRACE_INFO(
        ref  TTRACE_INFO info,
        string  trid,
        string  dsper,
        int     totsmp,
        int     repgsz,
        int     svid_count );
```

（2）引数

info

初期設定する TTRACE_INFO トレース情報構造体のポインタです。

trid

トレース ID（文字列）です。

dsper

トレースデータ収集時間を文字列です。“hhmmsscc”（8 文字固定）　の文字列で表現します。

hh:時間，mm:分，ss:秒，cc:1/100 秒

totsmp

合計サンプル数を指定します。

repgsz

レポートグループのサイズです。（装置が 1 個の S6F1 で送信するサンプル数）

svid_count

トレース対象の SV の数です。（TTRACE_INFO 内のリストに設定する SV 数）

（3）戻り値

戻り値	意味
0	正常に設定できた。
(-1)	引数に指定された値が正しくなかった。

（4）説明

本関数は、info で指定された TTRACE_INFO トレース情報構造体の初期設定を行います。
最初に info 内をゼロクリアします。

info のメンバー内にそれぞれの引数を設定します。

TTRACE_INFO メンバー内の以下のメンバーについては固定データを設定します。

format = ICODE_A
(ICODE_A は DSHDR2 ドライバーが定義するデータアイテムのフォーマットで ASCII です)
tot_fmt = ICODE_A, tot_asize=0
gsz_fmt = ICODE_A. gsz_asize=0

引数の値が以下のケースの場合はエラー(-1)を返却します。

trid 文字列長が=0、dsper 文字列長が 8 でない。
totsmp, repgsz または svid_count の値が=0 である。

トレースする SVID の TTRACE_INFO 内への設定は DshPutTTRACE_INFO()関数を使用します。

TSOOL_INFO info 使用が済んだら、情報内に確保して使用したメモリは DshFreeTTRACE_INFO ()関数を使って開放することがします。

（5）構造体

```
typedef struct{
        char    *name;                  // trace name
        char    *trid;                  // trace id
        int     format;                 // trace id format
        int     asize;                  // trace id array size
        int     max_asize;
        char    *dsper;                 // trace 時間周期
        int     dsper_time;             // trace 時間周期-数値
        int     totsmp;                 // total sample 数
        int     tot_fmt;
        int     tot_asize;
        int     repgsz;                 // report group size
        int     gsz_fmt;                //
        int     gsz_asize;              //
        int     svid_count;             // svid list の size
        TSVID   *svid_list;             // svid list
} TTRACE_INFO;
```

２．１．11　DshPutTTRACE_INFO() – トレース S2F23 情報構造体への SVID 設定

（1）呼出書式

[c,C++]

```
API int APIX DshPutTTRACE_INFO(
        TTRACE_INFO *info,                  // ﾄﾚｰｽ情報構造体の格納ﾎﾟｲﾝﾀ
        int      order,                     // SVID ﾘｽﾄの設定位置(0,1,..)
        TSVID    svid                       // 設定する SVID(状態変数 ID)
);
```

[VB.Net]

```
Function DshPutTTRACE_INFO (
        ByRef  info As TTRACE_INFO,
        order  As Integer,
        svid   As Integer) As Integer
```

[C#]

```
int DshPutTTRACE_INFO(
        ref  TTRACE_INFO info,
        int  order,
        uint svid );
```

（2）引数

info

初期設定する TTRACE_INFO トレース情報構造体のポインタです。

order

info 内の svid_list リスト内の設定位置を指定します。位置は先頭が 0 から始まります。

svid

設定したい装置状態変数 ID です。

（3）戻り値

戻り値	意味
0	正常に設定できた。
(-1)	設定できなかった(order の値が間違っていた)

（4）説明

本関数は、info で指定された TTRACE_INFO トレース情報構造体内のメンバー svid_list のリストの order 番目の位置に svid で指定された装置状態変数 ID を設定します。

order の値が、info 内の svid_count の値以上であった場合は、エラー(-1)を返却します。

２．１．12　DshFreeTTRACE_INFO() – トレース情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTTRACE_INFO(
        TTRACE_INFO *info               // ﾒﾓﾘを開放したい変数情報ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTTRACE_INFO (
        ByRef  info As TTRACE_INFO)
```

[C#]

```
void DshFreeTTRACE_INFO(
        ref  TTRACE_INFO  info );
```

（2）引数

info

メモリを解放したい変数情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TTRACE_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TTRACE_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．１．13 DshInitTEC_NAME_LIST () – EC 装置定数名リスト構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTV_VALUE_LIST (
        TEC_NAME_LIST *list,          // TEC_NAME_LIST 変数値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int           count           // list に含む VID(=ECID)の数
);
```

[VB.Net]

```
Sub DshInitTEC_NAME_LIST (
        ByRef list As TEC_NAME_LIST,
        count As Integer)
```

[C#]

```
void DshInitTEC_NAME_LIST(
        ref TEC_NAME_LIST  list,
        int count );
```

（2）引数

list

TEC_NAME_LIST 構造体のポインタです。

count

list に格納する変数情報の数です。(VID の数)

（3）戻り値

なし。

（4）説明

本関数は、TEC_NAME_LIST 構造体を使用するメッセージ、S1F12 メッセージを作るときに使用します。構造体内に含める情報は、変数の数と、各変数の名前と物理単位(Units)です。

list で指定された TEC_NAME_LIST 構造体内に、count 分の名前情報を保存できるように初期設定します。

名前情報を設定するためには、DshPutTEC_NAME_LIST()関数を使用します。

TEC_NAME_LIST 構造体の使用が済んだら、DshFreeTEC_NAME_LIST()関数を使って、構造体内で使用されたメモリを解放してください。

（5）構造体

```
typedef struct{
        int         count;
        TEC_NAME    **name_list;
} TEC_NAME_LIST;
```

```
typedef struct{
        TECID       svid;
        char        *name;
        char        *units;
} TEC_NAME;
```

２．１．14 DshPutTEC_NAME_LIST () – EC 装置定数名の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTEC_NAME_LIST (
        TEC_NAME_LIST *list,        // TEC_NAME_LIST 変数名情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        TVID          vid,          // 追加する変数の ID
        char          *name,        // EC の名前
        int           format,       // 値のﾌｫｰﾏｯﾄ
        void          *min,         // 最小値
        void          *max,         // 最大値
        void          *nominal,     // 初期値
        char          *units        // 単位
);
```

[VB.Net]

```
Function DshPutTEC_NAME_LIST (
        ByRef list As TEC_NAME_LIST,
        vid     As Integer,
        name    As String,
        format  As Integer,
        min     As IntPtr,
        max     As IntPtr,
        nominal As IntPtr,
        units   As String) As Integer
```

[C#]

```
int DshPutTEC_NAME_LIST(
        ref TEC_NAME_LIST list,
        uint    vid,
        string  name,
        int     format,
        IntPtr  min,
        IntPtr  max,
        IntPtr  nominal,
        string  units);
```

（2）引数

list

TEC_NAME_LIST 構造体のポインタです。

vid

list 内に追加する変数 ID です。

name

変数名です。

format

ﾃﾞｰﾀ値のﾌｫｰﾏｯﾄです。(ICODE_U4 など)

min

最小値です。

max

最大値です。

nominal

初期値（Default 値）です。

units

単位名です。

（3）戻り値

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

（4）説明

DshInitTEC_NAME_LIST()関数で初期設定された list 内に 1 個の名前情報を追加します。
追加する情報は引数に指定されている ID、変数名、値のフォーマット、最小、最大、初期値と物理単位です。

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

count 分を超える数の変数情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．１．15　DshFreeTEC_NAME_LIST() – EC 装置定数名リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTEC_NAME_LIST(
        TEC_NAME_LIST *list             // ﾒﾓﾘを開放したい変数情報ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTEC_NAME_LIST (
        ByRef list As TEC_NAME_LIST)
```

[C#]

```
void DshFreeTEC_NAME_LIST(
        ref TEC_NAME_LIST list );
```

（2）引数

list

メモリを解放したい変数情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TEC_NAME_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TEC_NAME_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

２．１．16　DshInitTLIMIT_LIST() – 変数リミット情報リスト構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTLIMIT_LIST(
        TLIMIT_LIST  *list,          // TLIMIT_LIST 変数リミット値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int          count           // list に含む VID(変数 ID)の数
);
```

[VB.Net]

```
Sub DshInitTLIMIT_LIST(
        ByRef  list As TLIMIT_LIST,
        count  As Integer)
```

[C#]

```
void DshInitTLIMIT_LIST(
        ref  TLIMIT_LIST list,
        int  count );
```

（2）引数

list

TLIMIT_LIST 構造体のポインタです。

count

list に格納する変数リミット情報の数です。(VID の数)

（3）戻り値

なし。

（4）説明

本関数は、TLIMIT_LIST 構造体を使用するメッセージ、S2F45 メッセージを作るときに使用します。構造体内に含める情報は、変数リミットの数と、各変数リミットリミット情報保存リストになります。

DshInitTLIMIT_LIST()関数は TLIMIT_LIST 構造体を初期化するための関数です。
TLIMIT_LIST 構造体には、１個以上の変数のリミット情報を保存します。

１個の変数のリミット情報の保存には TLIMIT_INFO 構造体を使用します。

変数リミット情報の設定には、DshPutTLIMIT_LIST()関数を使用します。

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

（5）構造体

```
typedef struct{
        int         vid_count;
        TLIMIT_INFO **limit_list;
} TLIMIT_LIST;
```

```
typedef struct{
        TDVID       vid;
        int         limit_count;
        TLIMIT_ID_INFO **limitid_list;
        int         format;
        int         asize;
} TLIMIT_INFO;
```

```
typedef struct{
        TLIMITID    limit_id;
        void        *upperdb;
        void        *lowerdb;
} TLIMIT_ID_INFO;
```

２．１．17　DshPutTLIMIT_LIST() – 装置変数リミット情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTLIMIT_LIST(
        TLIMIT_LIST    *list,          // TLIMIT_LIST 変数リミット値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        TLIMIT_INFO    *info           // 追加するﾘﾐｯﾄ情報　(構造体単位)
      );
```

[VB.Net]

```
Function DshPutTLIMIT_LIST(
        ByRef list  As TLIMIT_LIST,
        ByRef info  As TLIMIT_INFO} As Integer
```

[C#]

```
int DshPutTLIMIT_LIST(
        ref TLIMIT_LIST  list,
        ref TLIMIT_INFO  info);
```

（2）引数

list

TLIMIT_LIST 構造体のポインタです。

info

追加する変数ﾘﾐｯﾄ情報です。

（3）戻り値

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

（4）説明

先に説明した DshInitTLIMIT_LIST()関数で初期設定された list 内に、info 構造体に保存されている変数リミット情報を構造体単位で追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。
DshInitTLIMIT_LIST()関数で設定した count 分だけの変数リミット情報を加えることがします。

count 分を超える数の変数リミット情報を追加しようとした場合、戻り値として(-1)が返却されます。

TLIMIT_INFO への情報設定には、DshInitTLIMIT_INFO(), DshPutTLIMIT_INFO()関数を使用してください。（後述）

２．１．18　DshFreeTLIMIT_LIST() – 変数リミット情報リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTLIMIT_LIST(
        TLIMIT_LIST *list               // ﾒﾓﾘを開放したい変数リミット情報ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTLIMIT_LIST(
        ByRef list As TLIMIT_LIST)
```

[C#]

```
void DshFreeTLIMIT_LIST(
        ref TLIMIT_LIST list );
```

（2）引数

list

メモリを解放したい変数リミット情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TLIMIT_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TLIMIT_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

２．１．19 DshInitTLIMIT_INFO() – 変数リミット情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTLIMIT_INFO(
        TLIMIT_INFO  *info,         // TLIMIT_INFO 変数ﾘﾐｯﾄ値情報構造体のﾎﾟｲﾝﾀ
        TVID         vid,           // 変数 ID
        int          fmt,           // 値のﾌｫｰﾏｯﾄ
        int          asize,         // 配列ｻｲｽﾞ(=1 固定)
        int          count          // 保存ﾘﾐｯﾄ ID 数
);
```

[VB.Net]

```
Sub DshInitTLIMIT_INFO(
        ByRef info As TLIMIT_INFO,
        vid       As UInteger,
        fmt       As Integer,
        asize     As Integer,
        count As Integer)
```

[C#]

```
void DshInitTLIMIT_INFO(
        ref TLIMIT_INFO info,
        uint  vid,
        int   fmt,
        int   asize,
        int   count );
```

（2）引数

info

TLIMIT_INFO 構造体のポインタです。

vid

変数 ID です。

fmt

変数値のフォーマットです。

asize

値の配列サイズです。

count

info に格納する変数リミット ID 数です。

（3）戻り値

なし。

（4）説明

本関数は、TLIMIT_INFO 構造体を使用するメッセージ、S2F45 メッセージを作るときに使用します。構造体内に含める情報は、変数リミットの数と、各変数リミットの ID と値になります。

DshInitTLIMIT_INFO()関数は TLIMIT_INFO 構造体を初期化するための関数です。

TLIMIT_INFO に count 分の変数リミット ID 情報を格納するための初期設定処理を行います。

変数リミット ID 値の設定には、DshPutTLIMIT_INFO()関数を使用します。

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

（5）構造体

```
typedef struct{
        TDVID       vid;
        int         limit_count;
        TLIMIT_ID_INFO **limitid_list;
        int         format;
        int         asize;
} TLIMIT_INFO;
```

```
typedef struct{
        TLIMITID    limit_id;
        void        *upperdb;
        void        *lowerdb;
} TLIMIT_ID_INFO;
```

２．１．20　DshPutTLIMIT_INFO() – 装置変数リミット情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTLIMIT_INFO(
        TLIMIT_INFO   *info,          // TLIMIT_INFO 変数ﾘﾐｯﾄ値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int           limit_id,       // 追加するﾘﾐｯﾄ情報
        void          upperdb,        // UPPERDB
        void          upperdb         // LOWERDB
        );
```

[VB.Net]

```
Function DshPutTLIMIT_INFO(
        ByRef info As TLIMIT_INFO,
        limit_id   As Integer,
        upperdb    As IntPtr,
        lowerdb    As IntPtr) As Integer
```

[C#]

```
int DshPutTLIMIT_INFO(
        ref TLIMIT_INFO info,
        int     limit_id,
        IntPtr  upperdb,
        IntPtr  lowerdb );
```

（2）引数

info

TLIMIT_INFO 構造体のポインタです。

limit_id

ﾘﾐｯﾄ ID です。

upperdb

UPPERDB です。(ﾃﾞｯﾄﾞﾊﾞﾝﾄﾞの上限値、ﾌｫｰﾏｯﾄは DshInitLIMIT_INFO で指定したもの)

lowerdb

LOWERDB です。(ﾃﾞｯﾄﾞﾊﾞﾝﾄﾞの下限値、ﾌｫｰﾏｯﾄは DshInitLIMIT_INFO で指定したもの)

（3）戻り値

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

（4）説明

先に説明した DshInitTLIMIT_INFO()関数で初期設定された info 内に、リミット ID とその ID の上下限デッドバンド値を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

DshInitTLIMIT_INFO()関数で設定した count 分だけの変数リミット情報を加えることがします。

count 分を超える数の変数リミット情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．１．21　DshFreeTLIMIT_INFO() – 変数リミット情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTLIMIT_INFO(
        TLIMIT_INFO  *info              // ﾒﾓﾘを開放したい変数ﾘﾐｯﾄ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTLIMIT_INFO(
        ByRef info As TLIMIT_INFO)
```

[C#]

```
void DshFreeTLIMIT_INFO(
        ref TLIMIT_INFOinfo );
```

（2）引数

info

メモリを解放したい変数リミット情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TLIMIT_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TLIMIT_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．２　レポート、収集イベント関連関数

以下の関数があります。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTRP_LIST()	TRP_LIST 構造体の初期設定	S2F33 (REPORT)
	DshInitTRP_LINK	TRP_LINK 構造体の初期設定	
	DshPutTRP_LINK()	同　ﾘﾝｸ情報設定 (変数 ID)	
	DshFreeTRP_LIST()	同　内部使用ﾒﾓﾘの解放	
2	DshInitTCE_LIST()	TCE_LIST 構造体の初期設定	S2F35 (CE)
	DshPutTCE_LINK()	同　ﾘﾝｸ情報設定 (ﾚﾎﾟｰﾄ ID)	
	DshFreeTCE_LIST()	同　内部使用ﾒﾓﾘの解放	
3	DshInitTS6F11_CE_INFO()	TS6F11_CE_INFO 構造体の初期設定	S6F11 S6F16 S6F20
	DshPutTS6F11_CE_INFO()	同　１個のﾚﾎﾟｰﾄ情報の設定	
	DshFreeTS6F11_CE_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTS6F11_RP_INFO()	TS6F11_RP_INFO 構造体の初期設定	
	DshPutTS6F11_RP_INFO()	同 1 個の変数値を設定	
	DshFreeTS6F11_RP_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTS6F11_V_INFO()	TS6F11_V_INFO の構造体の初期設定	
	DshFreeTS6F11_V_INFO()	同　内部使用ﾒﾓﾘの解放	

２．２．１　DshInitTRP_LIST() – レポート・リンク情報リストの初期設定

（1）呼出書式

[c,C++]

```
API int  APIX DshInitTRP_LIST(
        TRP_LIST     *list,              // TRP_LIST 構造体のﾎﾟｲﾝﾀ
        int          count               // ｺﾏﾝﾄﾞｺｰﾄﾞ情報ﾘｽﾄのｻｲｽﾞ
);
```

[VB.Net]

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

[C#]

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

（2）引数

list

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

count

TRP_LIST に含まれる RPID の数です。

（3）戻り値

なし。

（4）説明

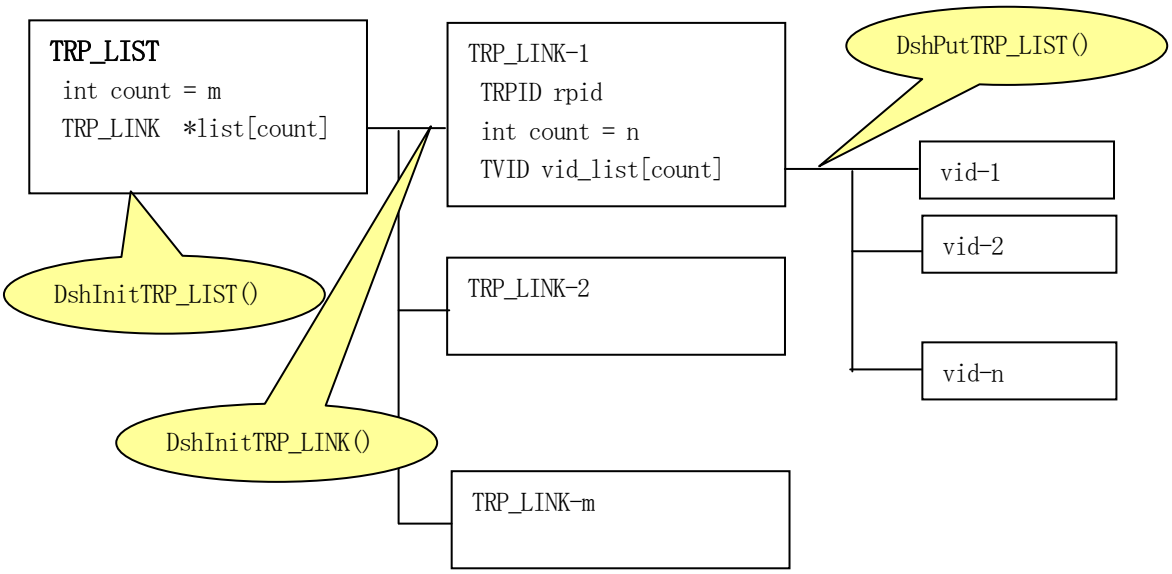
本関数は APP が S2F33 を送信する際のレポートリンク情報のためのリスト構造体を初期設定するために使用します。

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

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

DshInitTRP_LINK()

DshPutTRP_LINK()



TRP_LIST 構造体の使用後、DshFreeTRP_LIST()関数を使って構造体内部で使用したメモリを開放してください。

（5）構造体

```
typedef struct{
        int         count;
        TRP_LINK    **rp_list;
} TRP_LIST;
```

```
typedef struct{
        TRPID       rpid;
        int         count;
        TVID        *vid_list;
} TRP_LINK;
```

２．２．２　DshInitTRP_LINK() – レポート・リンク情報の初期設定

（1）呼出書式

[c,C++]

```
API int  APIX DshInitTRP_LINK(
        TRP_LINK     *list,          // TRP_LINK 構造体のﾎﾟｲﾝﾀ
        int          rp_order,       // list 内の LIST 位置 (0,1...)
        TRPID        rpid,           // TRP_LINK 内に設定する RPID
        int          v_count         // ﾚﾎﾟｰﾄﾘﾝｸ情報ﾘｽﾄのｻｲｽﾞ(ﾘﾝｸ変数の数)
);
```

[VB.Net]

```
Function DshInitTRP_LINK (
        ByRef list As dsh_info.TRP_LIST,
        rp_order As Integer,
        rpid     As Integer,
        v_count  As Integer) As Integer
```

[C#]

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

（2）引数

list

TRP_LIST 構造体のポインタ

rp_order

list 内のリスト位置を指定します。(0,1..)

rpid

設定対象のレポート ID です。

v_count

list の rp_order 番目の RPID にリンクされる変数 ID の数です。

（3）戻り値

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

（4）説明

本関数はアプリケーションが S2F33 を送信する際のレポート・リンク情報リスト構造体内の rp_order 番目のリストに位置する TRP_LINK 構造体の初期設定を行うために使用します。

rp_order 番目のリストに TRP_LINK 領域用メモリを確保し、rpid と v_count を設定します。

２．２．３　DshPutTRP_LINK() – レポート・リンク情報に変数 ID を追加する

（1）呼出書式

[c,C++]

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

[VB.Net]

```
Function DshPutTRP_LINK (
        ByRef list As dsh_info.TRP_LIST,
        rp_order  As Integer,
        vid       As Integer) As Integer
```

[C#]

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

（2）引数

list

TRP_LIST　　*list,　　// TRP_LIST 構造体のﾎﾟｲﾝﾀ

rp_order

list 内のリスト位置を指定します。(0,1..)

vid

list 内の rp_order 番目の TRP_LINK 構造体内のリストに加える変数 ID です。

（3）戻り値

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

（4）説明

本関数は APP が S2F33 を送信する際のイベント・レポート・リンク情報リスト構造体内の rp_order 番目のリストに用意された TRP_LINK 構造体内の v_list に vid を加えます。

既に、v_count 分のレポート ID が設定されていた場合には、(-1)を返却します。

２．２．４　DshFreeTRP_LIST() – レポート情報リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRP_LIST(
        TRP_LIST *list              // ﾒﾓﾘを開放したいｲﾍﾞﾝﾄﾘﾝｸﾚﾎﾟｰﾄ情報ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

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

[C#]

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

（2）引数

list

メモリを解放したいレポートリンク情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TRP_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TRP_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

２．２．５　DshInitTCE_LIST() – イベント・リンク情報リストの初期設定

（1）呼出書式

[c,C++]

```
API int  APIX DshInitTCE_LIST(
        TCE_LIST     *list,              // TCE_LIST 構造体のﾎﾟｲﾝﾀ
        int          count               // ｺﾏﾝﾄﾞｺｰﾄﾞ情報ﾘｽﾄのｻｲｽﾞ
);
```

[VB.Net]

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

[C#]

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

（2）引数

list

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

count

TCE_LIST に含まれる CEID の数です。

（3）戻り値

なし。

（4）説明

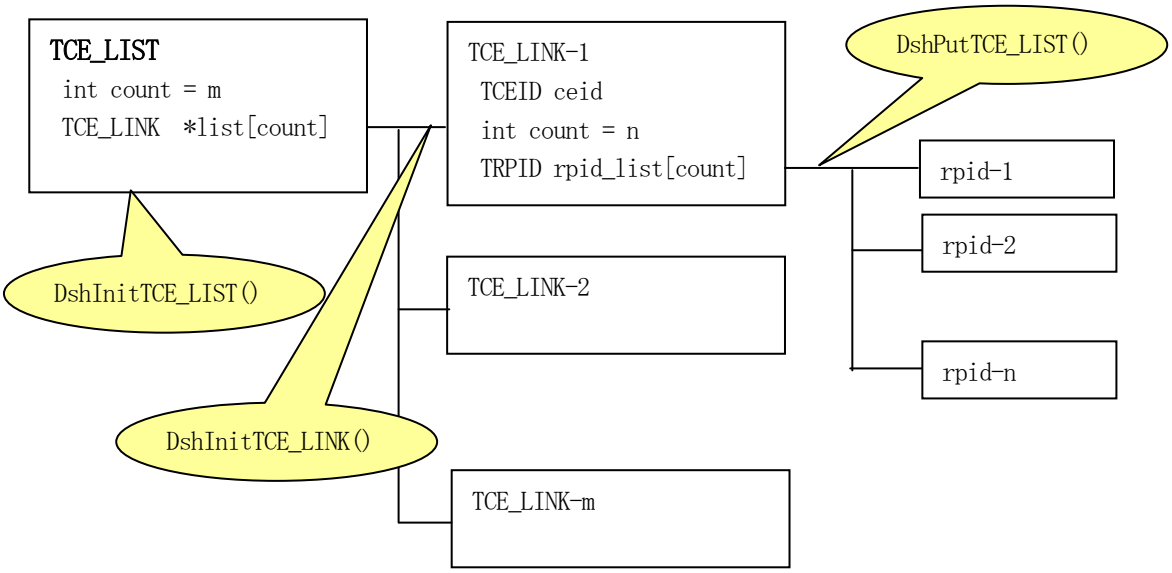
本関数は APP が S2F35 を送信する際のイベント・リンク情報のためのリスト構造体を初期設定するために使用します。

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

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

DshInitTCE_LINK()

DshPutTCE_LINK()



TCE_LIST 構造体の使用後、DshFreeTCE_LIST()関数を使って構造体内部で使用したメモリを開放してください。

（5）構造体

```
typedef struct{
        int         count;
        TCE_LINK    **ce_list;
} TCE_LIST;
```

```
typedef struct{
        TCEID       ceid;
        int         count;
        TRPID       *rpid_list;
} TCE_LINK;
```

２．２．６　DshInitTCE_LINK() – イベント・レポート・リンク情報の初期設定

（1）呼出書式

[c,C++]

```
API int  APIX DshInitTCE_LINK(
        TCE_LINK     *list,              // TCE_LINK 構造体のﾎﾟｲﾝﾀ
        int          ce_order,           // list 内の LIST 位置
        TCEID        ceid,               // TCE_LINK 内に設定する CEID
        int          rp_count            // ﾚﾎﾟｰﾄﾘﾝｸ情報ﾘｽﾄのｻｲｽﾞ (ﾚﾎﾟｰﾄ ID 数)
);
```

[VB.Net]

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

[C#]

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

（2）引数

list

TCE_LIST 構造体のポインタです。

ce_order

list 内のリスト位置を指定します。(0,1..)

ceid

設定対象のイベント ID です。

rp_count

list の ce_order 番目の CEID にリンクされる RPID の数です。
(TCE_LINK 構造体のﾒﾝﾊﾞｰ count です。)

（3）戻り値

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

（4）説明

本関数は APP が S2F35 を送信する際のイベント・レポート・リンク情報リスト構造体内の ce_order 番目のリストに位置する TCE_LINK 構造体の初期設定を行うために使用します。

ce_order 番目のリストに TCE_LINK 領域用メモリを確保し、ceid と rp_count を設定します。

２．２．７　DshPutTCE_LINK() – イベント・レポート・リンク情報にレポート ID を追加する

（1）呼出書式

[c,C++]

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

[VB.Net]

```
Function DshPutTCE_LINK (
        ByRef list As dsh_info.TCE_LIST,
        ce_order  As Integer,
        rpid      As Integer) As Integer
```

[C#]

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

（2）引数

list

TCE_LIST 構造体のポインタです。

ce_order

list 内のリスト位置を指定します。(0,1..)

rpid

list 内の ce_order 番目の TCE_LINK 構造体内のリストに加えるレポート ID です。

（3）戻り値

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

（4）説明

本関数は APP が S2F35 を送信する際のイベント・レポート・リンク情報リスト構造体内の ce_order 番目のリストに用意された TCE_LINK 構造体内の rp_list のリストに rpid を加えます。

既に、rp_count 分のレポート ID が設定されていた場合には、(-1)を返却します。

２．７．８　DshFreeTCE_LIST() – イベント・リンク・レポート情報リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTCE_LIST(
        TCE_LIST *list              // ﾒﾓﾘを開放したいｲﾍﾞﾝﾄﾘﾝｸﾚﾎﾟｰﾄ情報ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

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

[C#]

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

（2）引数

list

メモリを解放したいイベント・リンク・レポート情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TCE_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TCE_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

２．２．９　DshInitTS6F11_CE_INFO() – S6F11 イベント情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTS6F11_CE_INFO(
      TS6F11_CE_INFO  *info,        // TS6F11_CE_INFOS6F11 イベント値情報構造体のﾎﾟｲﾝﾀ
      TCEID         ceid,           // CEID
      int           rp_count        // ﾘﾝｸ・ﾚﾎﾟｰﾄ数
);
```

[VB.Net]

```
Sub DshInitTS6F11_CE_INFO(
      ByRef info As TS6F11_CE_INFO,
      ceid        As UInteger,
      rp_count    As Integer)
```

[C#]

```
void DshInitTS6F11_CE_INFO(
      ref TS6F11_CE_INFO info,
      uint  ceid,
      int   rp_count );
```

（2）引数

info

TS6F11_CE_INFO 構造体のポインタです。

ceid

CEID です。

rp_count

当該 CEID にリンクするレポート ID 数です。

（3）戻り値

なし。

（4）説明

本関数は、TS6F11_CE_INFO 構造体を初期設定するために使用します。

TS6F11_CE_INFO 内に rp_count 分の S6F11 レポート情報を格納するための処理を行います。

1 個のレポート情報（TS6F11_RP_INFO）を TS6F11_CE_INFO 内に設定には、DshPutTS6F11_CE_INFO()関数を使用します。

構造体の使用が済んだら、DshFreeTS6F11_CE_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        TCEID           ceid;
        int             rp_count;
        TS6F11_RP_INFO ** rp_list;
} TS6F11_CE_INFO;
```

```
typedef struct{
        TRPID           rpid;
        int             v_count;
        TS6F11_V_INFO   **v_list;
} TS6F11_RP_INFO;
```

```
typedef struct{
        TVID            vid;
        int             format;
        int             asize;
        void            *value;
        void            **link;  // format-L, LINK TS6F11_V_INFO を指す
} TS6F11_V_INFO;
```

２．２．10　DshPutTS6F11_CE_INFO() – S6F11 イベント情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTS6F11_CE_INFO(
      TS6F11_CE_INFO   *info,          // TS6F11_CE_INFOS6F11 ｲﾍﾞﾝﾄ値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
      TS6F11_RP_INFO   *rinfo          // TS6F11_RP_INFO ﾚﾎﾟｰﾄ情報構造体のﾎﾟｲﾝﾀｰ
      );
```

[VB.Net]

```
Function DshPutTS6F11_CE_INFO(
      ByRef info  As TS6F11_CE_INFO,
      ByRef rinfo As TS6F11_RP_INFO) As Integer
```

[C#]

```
int DshPutTS6F11_CE_INFO(
      ref TS6F11_CE_INFO  info,
      ref TS6F11_RP_INFO  rinfo);
```

（2）引数

info

TS6F11_CE_INFO 構造体のﾎﾟｲﾝﾀです。

rinfo

TS6F11_RP_INFO 構造体のﾎﾟｲﾝﾀです。

（3）戻り値

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

（4）説明

先に説明した DshInitTS6F11_CE_INFO()関数で初期設定された info 内に、1 個のレポート情報を、TS6F11_RP_INFO 構造体単位で追加します。レポート情報は、TS6F11_RP_INFO 内に保存され、その中にレポート ID にリンクされた変数（0 個以上）の値が格納されています。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。
DshInitTS6F11_CE_INFO()関数で設定した rp_count 分だけのレポート情報を加えることがします。

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

２．２．11　DshFreeTS6F11_CE_INFO() – S6F11 イベント情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTS6F11_CE_INFO(
        TS6F11_CE_INFO  *info              // ﾒﾓﾘを開放したいS6F11 ｲﾍﾞﾝﾄ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTS6F11_CE_INFO(
        ByRef info As TS6F11_CE_INFO )
```

[C#]

```
void DshFreeTS6F11_CE_INFO(
        ref TS6F11_CE_INFO info );
```

（2）引数

info

メモリを解放したいS6F11 イベント情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TS6F11_CE_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TS6F11_CE_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．２．12　DshInitTS6F11_RP_INFO() – S6F11 レポート情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTS6F11_RP_INFO(
      TS6F11_RP_INFO  *info,        // TS6F11_RP_INFO  S6F11 ﾚﾎﾟｰﾄ値情報構造体のﾎﾟｲﾝﾀ
      TRPID         rpid,           // RPID
      int           v_count         // ﾘﾝｸ・変数 ID 数
);
```

[VB.Net]

```
Sub DshInitTS6F11_RP_INFO(
      ByRef info As TS6F11_RP_INFO,
      rpid    As UInteger,
      v_count As Integer)
```

[C#]

```
void DshInitTS6F11_RP_INFO(
      ref TS6F11_RP_INFO info,
      uint  rpid,
      int   v_count );
```

（2）引数

info

TS6F11_RP_INFO 構造体のポインタです。

rpid

RPID です。

v_count

当該 RPID にリンクする変数 ID 数です。

（3）戻り値

なし。

（4）説明

本関数は、TS6F11_RP_INFO 構造体を初期設定するために使用します。

TS6F11_RP_INFO 内に v count 分の S6F11 のための変数情報を格納するための処理を行います。

1 個の変数情報（TS6F11_V_INFO）を TS6F11_RP_INFO 内に設定するために、DshPutTS6F11_RP_INFO()関数を使用します。

構造体の使用が済んだら、DshFreeTS6F11_RP_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

２．２．9-（5）を参照ください。

２．２．13　DshPutTS6F11_RP_INFO() – S6F11 レポート情報への追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTS6F11_RP_INFO(
      TS6F11_RP_INFO    *info,          // TS6F11_RP_INFOS6F11 ﾚﾎﾟｰﾄ値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
      TS6F11_V_INFO     *vinfo          // TS6F11_V_INFO ﾚﾎﾟｰﾄ情報構造体のﾎﾟｲﾝﾀｰ
      );
```

[VB.Net]

```
Function DshPutTS6F11_RP_INFO(
      ByRef info    As TS6F11_RP_INFO,
      ByRef vinfo   As TS6F11_V_INFO) As Integer
```

[C#]

```
int DshPutTS6F11_RP_INFO(
      ref TS6F11_RP_INFO   info,
      ref TS6F11_V_INFO    vinfo);
```

（2）引数

info

TS6F11_RP_INFO 構造体のﾎﾟｲﾝﾀです。

vinfo

TS6F11_V_INFO 構造体のﾎﾟｲﾝﾀです。

（3）戻り値

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

（4）説明

先に説明した DshInitTS6F11_RP_INFO()関数で初期設定された info 内に、1 個の変数情報を追加します。

vinfo 指定された TS6F11_V_INFO 構造体に保存されている変数情報を、TS6F11_RP_INFO 内の v_list に格納します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。
DshInitTS6F11_RP_INFO()関数で設定した v_count 分だけの変数情報を加えることがします。

v_count 分を超える数の変数情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．３．14　DshFreeTS6F11_RP_INFO() – S6F11 レポート情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTS6F11_RP_INFO(
        TS6F11_RP_INFO  *info              // ﾒﾓﾘを開放したいS6F11ﾚﾎﾟｰﾄ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTS6F11_RP_INFO(
        ByRef info As TS6F11_RP_INFO)
```

[C#]

```
void DshFreeTS6F11_RP_INFO(
        ref TS6F11_RP_INFO info );
```

（2）引数

info

メモリを解放したいS6F11 レポート情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TS6F11_RP_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TS6F11_RP_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．２．15 DshInitTS6F11_V_INFO() – S6F11 変数情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTS6F11_V_INFO(
        TS6F11_V_INFO  *info,          // TS6F11_V_INFO  S6F11 変数値情報構造体のﾎﾟｲﾝﾀ
        TVID           vid,            // VID
        int            format,         // 変数値のﾌｫｰﾏｯﾄ
        int            asize,          // 配列ｻｲｽﾞ
        void           *value          // 値の格納ﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshInitTS6F11_V_INFO(
        ByRef info As TS6F11_V_INFO,
        vid    As UInteger,
        format As Integer,
        asize  As Integer,
        value  As IntPtr)
```

[C#]

```
void DshInitTS6F11_V_INFO(
        ref TS6F11_V_INFO info,
        uint   vid,
        int    format,
        int    asize,
        IntPtr value );
```

（2）引数

info

TS6F11_V_INFO 構造体のポインタです。

vid

VID です。

format

変数値のフォーマットです。

asize

データの配列サイズです。

value

変数値が格納されているポインタです。

（3）戻り値

なし。

（4）説明

本関数は、TS6F11_V_INFO 構造体内に変数情報を設定するために使用します。

構造体の使用が済んだら、DshFreeTS6F11_V_INFO()関数によって内部で使用したメモリを解放してください。

２．３．16　DshFreeTS6F11_V_INFO() – S6F11 変数情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTS6F11_V_INFO(
        TS6F11_V_INFO  *info              // ﾒﾓﾘを開放したいS6F11 変数情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTS6F11_V_INFO(
        ByRef info As TS6F11_V_INFO)
```

[C#]

```
void DshFreeTS6F11_V_INFO(
        ref TS6F11_V_INFO info );
```

（2）引数

info

メモリを解放したいS6F11 変数情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TS6F11_V_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TS6F11_V_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．３　アラーム関連関数

以下の関数があります。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTAL_S5F1_INFO()	TAL_S5F1_INFO 構造体の初期設定	S5F1
	DshFreeTAL_S5F1_INFO()	同　内部使用ﾒﾓﾘの解放	
2	DshInitTAL_S5F6_LIST()	TAL_S5F6_LIST 構造体の初期設定	S5F6
	DshPutTAL_S5F6_LIST()	同 TAL_S5F1_INFO を設定	
	DshFreeTAL_S5F6_LIST()	同　内部使用ﾒﾓﾘの解放	

２．３．１　DshInitTS5F1_INFO () – アラーム情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTS5F1_INFO (
      TS5F1_INFO    *info,          // TS5F1_INFO ｱﾗｰﾑ値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
      TALID         alid.           // alarm ID
      int           alcd,           // ALCD
      char          *altx           // ALTX
);
```

[VB.Net]

```
Sub DshInitTS5F1_INFO (
      ByRef info As TS5F1_INFO,
      alid As UInteger,
      alcd As Integer,
      altx As String )
```

[C#]

```
void DshInitTS5F1_INFO(
      ref TS5F1_INFO info,
      uint   alid,
      int    alcd,
      string altx
);
```

（2）引数

info

TS5F1_INFO 構造体のポインタです。

alid

アラーム ID です。

alcd

アラームコードです。

altx

アラームテキストです。

（3）戻り値

なし。

（4）説明

本関数は、TS5F1_INFO 構造体を使用するメッセージ、S5F1 メッセージを作るときに使用します。構造体内に含める情報は、アラーム ID,　アラームコードとアラームテキストです。

DshInitTS5F1_INFO()関数は TS5F1_INFO 構造体を初期化するための関数です。

構造体の使用が済んだら、DshFreeTS5F1_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        int         on_off;
        TALID       alid;
        TALCD       alcd;
        char        *altx;
} TAL_S5F1_INFO;
```

２．３．２　DshFreeTS5F1_INFO() – アラーム情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTS5F1_INFO(
        TS5F1_INFO *info                // ﾒﾓﾘを開放したいｱﾗｰﾑ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTS5F1_INFO (
        ByRef info As TS5F1_INFO)
```

[C#]

```
void DshFreeTS5F1_INFO(
        ref TS5F1_INFO info );
```

（2）引数

info

メモリを解放したいアラーム情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TS5F1_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TS5F1_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．１．４　DshInitTAL_S5F6_LIST () – アラーム情報リスト構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTV_VALUE_LIST (
        TAL_S5F6_LIST *list,          // TAL_S5F6_LIST 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int           count           // list に含む ALID の数
);
```

[VB.Net]

```
Sub DshInitTAL_S5F6_LIST (
        ByRef list As TAL_S5F6_LIST,
        count  As Integer)
```

[C#]

```
void DshInitTAL_S5F6_LIST(
        ref TAL_S5F6_LIST list,
        int  count );
```

（2）引数

list

TAL_S5F6_LIST 構造体のポインタです。

count

list に格納するアラームの数です。(ALID の数)

（3）戻り値

なし。

（4）説明

本関数は、TAL_S5F6_LIST 構造体を使用するメッセージ、S5F6 メッセージを作るときに使用します。構造体内に含める情報は、ALID の数と、各アラームの ALCD, ALTX です。
list で指定された TAL_S5F6_LIST 構造体内に、count 分のアラーム情報を保存できるように初期設定します。

名前情報を設定するためには、DshPutTAL_S5F6_LIST()関数を使用します。

TAL_S5F6_LIST 構造体の使用が済んだら、DshFreeTAL_S5F6_LIST()関数を使って、構造体内で使用されたメモリを解放してください。

（5）構造体

```
typedef struct{
        int         count;
        TAL_S5F1_INFO    **al_list;
} TAL_S5F6_LIST;
```

```
typedef struct{
        int         on_off;
        TALID       alid;
        TALCD       alcd;
        char        *altx;
} TAL_S5F1_INFO;
```

２．１．５　DshPutTAL_S5F6_LIST () – アラーム情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTAL_S5F6_LIST (
        TAL_S5F6_LIST *list,        // TAL_S5F6_LIST ｱﾗｰﾑ名情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        TALID         alid,         // 追加する ALID
        int           alcd,         // ALCD
        char          *altx         // ALTX
);
```

[VB.Net]

```
Function DshPutTAL_S5F6_LIST (
        ByRef list As TAL_S5F6_LIST,
        alid As UInteger,
        alcd As Integer,
        ByValaltx As String) As Integer
```

[C#]

```
int DshPutTAL_S5F6_LIST(
        ref TAL_S5F6_LIST list,
        uint alid,
        int  alcd,
        string altxt);
```

（2）引数

list

TAL_S5F6_LIST 構造体のポインタです。

alid

list 内に追加するｱﾗｰﾑ ID です。

alcd

ｱﾗｰﾑｺｰﾄﾞです。

altx

ｱﾗｰﾑﾃｷｽﾄです。

（3）戻り値

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

（4）説明

DshInitTAL_S5F6_LIST()関数で初期設定された list 内に 1 個のアラーム情報を追加します。
追加する情報は引数に指定されている ID、ALCD と ALTX です。
追加によって、本関数が呼び出される順番に値が構造体内に保存されます。
DshInitTAL_S5F6_LIST()関数で設定した count 分だけのアラーム情報を加えることがします。

count 分を超える数のアラーム情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．１．６　DshFreeTAL_S5F6_LIST() – アラーム情報リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTAL_S5F6_LIST(
        TAL_S5F6_LIST *list              // ﾒﾓﾘを開放したいｱﾗｰﾑ情報ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTAL_S5F6_LIST (
        ByRef list As TAL_S5F6_LIST)
```

[C#]

```
void DshFreeTAL_S5F6_LIST(
        ref TAL_S5F6_LIST list );
```

（2）引数

list

メモリを解放したいｱﾗｰﾑ情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TAL_S5F6_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TAL_S5F6_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

２．４　PP　プロセス・プログラム関連関数

ここでは、プロセス・プログラム、書式付プロセス・プログラム関連の関数について説明します。

以下の関数があります。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTPPID_LIST()	TPPID_LIST 構造体の初期設定	S7F17
	DshPutTPPID_LIST()	同　PPID を 1 個設定	S7F19
	DshFreeTPPID_LIST()	同　内部使用ﾒﾓﾘの解放	
2	DshInitTS7F23_INFO()	TS7F23_INFO 構造体の初期設定	S7F23
	DshPutTS7F23_INFO()	同　TFPP_CCODE を 1 個設定	S7F26
	DshFreeTS7F23_INFO	同　内部使用ﾒﾓﾘの解放	
	DshInitTFPP_CCODE()	TFPP_CCODE 構造体の初期設定	
	DshPutTFPP_CCODE()	同　ﾊﾟﾗﾒｰﾀを 1 個設定	
	DshFreeTFPP_CCODE()	同　内部使用ﾒﾓﾘの解放	
3	DshInitTPP_PVS_LIST()	TPP_PVS_LIST 構造体の初期設定	S7F27
	DshPutTPP_PVS_LIST()	同　妥当性情報を設定	
	DshFreeTPP_PVS_LIST()	同　内部使用ﾒﾓﾘの解放	

S7F3, S7F5 については、それら関連の API 関数では、引数として構造体を使用していませんので、関連関数はありません。

２．４．１　DshInitTPPID_LIST() – PPID 構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPPID_LIST(
        TPPID_LIST  *list,          // TPPID_LIST 構造体のポインタ
        int         count           // PPID 数
);
```

[VB.Net]

```
Sub DshInitTPPID_LIST(
        ByRef list As TPPID_LIST,
        count As Integer)
```

[C#]

```
void DshInitTPPID_LIST(
        ref TPPID_LIST list,
        int count );
```

（2）引数

list

TPPID_LIST 構造体のポインタです。

count

保存する PPID 数です。

（3）戻り値

なし。

（4）説明

本関数は、TPPID_LIST 構造体を初期設定するために使用します。

構造体内には、count 分の PPID を保存します。

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

（5）構造体

```
typedef struct{
        int         count;
        char        **ppid_list;
} TPPID_LIST;
```

２．４．２　DshPutTPPID_LIST() – PPID の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTPPID_LIST(
        TPPID_LIST    *list,        // TPPID_LIST 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        char     *ppid              // 追加する PPID
        );
```

[VB.Net]

```
Function DshPutTPPID_LIST(
        ByRef list As TPPID_LIST,
        ppid  As string) As Integer
```

[C#]

```
int DshPutTPPID_LIST(
        ref TPPID_LIST list,
        string  ppid);
```

（2）引数

list

TPPID_LIST 構造体のﾎﾟｲﾝﾀです。

ppid

PPID（文字列）です。

（3）戻り値

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

（4）説明

先に説明した DshInitTPPID_LIST()関数で初期設定された list 内に、1 個の PPID を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．４．３　DshFreeTPPID_LIST() – PPID 構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPPID_LIST(
        TPPID_LIST  *list               // ﾒﾓﾘを開放したいPPID構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPPID_LIST(
        ByRef list As TPPID_LIST )
```

[C#]

```
void DshFreeTPPID_LIST(
        ref TPPID_LIST list );
```

（2）引数

list

メモリを解放したいPPID構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPPID_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPPID_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

２．４．４　DshInitTS7F23_INFO() – 書式付 PP 情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTS7F23_INFO(
        TS7F23_INFO  *info,          // TS7F23_INFO 構造体のﾎﾟｲﾝﾀ
        char         *ppid,          // PPID
        char         *mdln,          // MDLN
        char         *softrev,       // SOFTREV
        int          ccode_count     // ｺﾏﾝﾄﾞｺｰﾄﾞﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTS7F23_INFO(
        ByRef info As TS7F23_INFO,
        ppid      As String,
        mdln      As String,
        softrev   As String,
        ccode_count As Integer)
```

[C#]

```
void DshInitTS7F23_INFO(
        ref TS7F23_INFO info,
        string   ppid,
        string   mdln,
        string   softrev,
        int      ccode_count );
```

（2）引数

info

TS7F23_INFO 構造体のポインタです。

ppid

プロセスプログラム ID です。

mdln

MDLN です。

softrev

SOFTREV です。

ccode_count

保存するコマンドコードパラメータ数です。

（3）戻り値

なし。

（4）説明

本関数は、TS7F23_INFO 構造体を初期設定するために使用します。

構造体内には、ccode_count 分の CCODE 情報（TCCODE_INFO 構造体）を保存します。

構造体の使用が済んだら、DshFreeTS7F23_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        char        *ppid;
        char        *mdln;
        char        *softrev;
        int         ccode_count;        // # of process commands
        TFPP_CCODE  **ccode_list;
}TS7F23_INFO;
```

```
typedef struct{
        int         ccode_fmt;
        int         ccode_size;
        int         max_ccode_size;
        void        *ccode;                 // command code
        int         ppara_count;            // # of para count
        TFPP_PARA   **ppara_list;
} TFPP_CCODE;
```

```
typedef struct{
        int         ppara_fmt;
        int         ppara_size;
        int         max_ppara_size;
        void        *ppara;
} TFPP_PARA;
```

２．４．５　DshPutTS7F23_INFO() – コマンドコード情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTS7F23_INFO(
        TS7F23_INFO    *info,              // TS7F23_INFO 構造体のﾎﾟｲﾝﾀ
        TFPP_CCODE     *cinfo              // 追加するｺﾏﾝﾄﾞｺｰﾄﾞ情報構造体のﾎﾟｲﾝﾀ
        );
```

[VB.Net]

```
Function DshPutTS7F23_INFO(
        ByRef info  As TS7F23_INFO,
        ByRef cinfo As TFPP_CCODE) As Integer
```

[C#]

```
int DshPutTS7F23_INFO(
        ref TS7F23_INFO  info,
        ref TFPP_CCODE   cinfo);
```

（2）引数

info

TS7F23_INFO 構造体のポインタです。

cinfo

コマンドコード情報の構造体のポインタです。

（3）戻り値

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

（4）説明

先に説明した DshInitTS7F23_INFO()関数で初期設定された info 内に、1 個のコマンドコード情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．４．６　DshFreeTS7F23_INFO() –書式付 PPID 構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTS7F23_INFO(
        TS7F23_INFO  *info              // ﾒﾓﾘを開放したいPPID構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTS7F23_INFO(
        ByRef info As TS7F23_INFO )
```

[C#]

```
void DshFreeTS7F23_INFO(
        ref TS7F23_INFO info );
```

（2）引数

info

メモリを解放したい書式付 PPID 構造体のポインタです。

（3）戻り値

なし。

（4）説明

TS7F23_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TS7F23_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．４．７　DshInitTFPP_CCODE() – PP コマンドコード構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTFPP_CCODE(
      TFPP_CCODE  *info,          // TFPP_CCODE 構造体のﾎﾟｲﾝﾀ
      int         format,         // ccode format
      void        *ccode,         // ccode
      int         para_count      // ｺﾏﾝﾄﾞｺｰﾄﾞﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTFPP_CCODE(
      ByRef info As TFPP_CCODE,
      format  As Integer,
      ccode   As IntPtr,
      para_count As Integer)
      )
```

[C#]

```
void DshInitTFPP_CCODE(
      ref TFPP_CCODE info,
      int    format,
      IntPtr ccode,
      int    para_count );
```

（2）引数

info

TFPP_CCODE 構造体のポインタです。

format

コマンドコードのフォーマットです。

ccode

コマンドコードの格納ポインタです。

para_count

保存するコマンドコードパラメータ数です。

（3）戻り値

なし。

（4）説明

本関数は、TFPP_CCODE 構造体を初期設定するために使用します。
構造体内には、para_count 分のパラメータ情報（TCCODE_PARA 構造体）を保存します。

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

（5）構造体

２．４．7-（5）構造体を参照してください。

２．４．８　DshPutTFPP_CCODE() – コマンドコード情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTFPP_CCODE(
        TFPP_CCODE    *info,          // TFPP_CCODE 構造体のポインタ
        int           format,         // 追加するパラメータ値のフォーマット
        void          *ppara          // パラメータ値のポインタ
        );
```

[VB.Net]

```
Function DshPutTFPP_CCODE(
        ByRef info As TFPP_CCODE,
        format As Integer,
        ppara  As IntPtr) As Integer
```

[C#]

```
int DshPutTFPP_CCODE(
        ref TFPP_CCODE info,
        int    format,
        IntPtr ppara);
```

（2）引数

info

TFPP_CCODE 構造体のポインタです。

format

ppara のフォーマットです。

ppara

パラメータ値のポインタです。

（3）戻り値

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

（4）説明

先に説明した DshInitTFPP_CCODE)関数で初期設定された info 内に、1 個のパラメータを追加します
追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．４．９　DshFreeTFPP_CCODE() –PP コマンドコード構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTFPP_CCODE(
        TFPP_CCODE  *info               // ﾒﾓﾘを開放したいｺﾏﾝﾄﾞｺｰﾄﾞ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTFPP_CCODE(
        ByRef info As TFPP_CCODE )
```

[C#]

```
void DshFreeTFPP_CCODE(
        ref TFPP_CCODE info );
```

（2）引数

info

メモリを解放したいコマンドコード構造体のポインタです。

（3）戻り値

なし。

（4）説明

TFPP_CCODE 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TFPP_CCODE の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．４．10　DshInitTPP_PVS_LIST() – PP 妥当性情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPP_PVS_LIST(
      TPP_PVS_LIST  *list,          // TPP_PVS_LIST 構造体のﾎﾟｲﾝﾀ
      char          *ppid,          // PPID
      int           err_count       // ｴﾗｰ情報の数
);
```

[VB.Net]

```
Sub DshInitTPP_PVS_LIST(
      ByRef list As TPP_PVS_LIST,
      ppid       As String,
      err_count  As Integer)
      )
```

[C#]

```
void DshInitTPP_PVS_LIST(
      ref TPP_PVS_LIST list,
      string  ppid,
      int     err_count );
```

（2）引数

list

TPP_PVS_LIST 構造体のポインタです。

ppid

プロセス・プログラム ID です。

err_count

保存するエラー情報数です。

（3）戻り値

なし。

（4）説明

本関数は、TPP_PVS_LIST 構造体を初期設定するために使用します。
構造体内には、PPID と err_count 分のエラー情報を保存します。

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

（5）構造体

```
typedef struct{
        char        *ppid;
        int         err_count;
        TPP_PVS_INFO **err_list;
} TPP_PVS_LIST;
```

```
typedef struct{
        int         ackc7a;
        int         seqnum;
        char        *errw7;
} TPP_PVS_INFO;
```

２．４．11　DshPutTPP_PVS_LIST() – PVS 情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTPP_PVS_LIST(
        TPP_PVS_LIST    *list,          // TPP_PVS_LIST 構造体のﾎﾟｲﾝﾀ
        int             ackc7a          // 追加するﾊﾟﾗﾒｰﾀ値のﾌｫｰﾏｯﾄ
        int             seqnum,         // ｺﾏﾝﾄﾞ番号
        char            *errw7          // ｴﾗｰﾃｷｽﾄ
        );
```

[VB.Net]

```
Function DshPutTPP_PVS_LIST(
        ByRef list As TPP_PVS_LIST,
        ackc7a As Integer,
        seqnum As Integer,
        errw7  As String) As Integer
```

[C#]

```
int DshPutTPP_PVS_LIST(
        ref TPP_PVS_LIST list,
        int    ackc7a,
        int    seqnum,
        string errw7);
```

（2）引数

list

TPP_PVS_LIST 構造体のﾎﾟｲﾝﾀです。

ackc7a

ACK です。

seqnum

コマンド番号です。

errw7

エラーテキストです。

（3）戻り値

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

（4）説明

先に説明した DshInitTPP_PVS_LIST)関数で初期設定された list 内に、1 個のエラー情報を追加します
追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．４．12　DshFreeTPP_PVS_LIST() –PPPVS 構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPP_PVS_LIST(
        TPP_PVS_LIST  *list              // ﾒﾓﾘを開放したいPVS構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPP_PVS_LIST(
        ByRef list As TPP_PVS_LIST )
```

[C#]

```
void DshFreeTPP_PVS_LIST(
        ref TPP_PVS_LIST list );
```

（2）引数

list

メモリを解放したいPVS構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPP_PVS_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPP_PVS_LIST の内容を 全て0 で初期設定します。
list が NULL ならば、何も処理しません。

２．５　Recipe レシピ関連関数

ここでは、レシピ関連の関数について説明します。

以下の関数があります。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTRCP_ACT_INFO()	TRCP_ACT_INFO 構造体の初期設定	S15F3
	DshFreeTRCP_ACT_INFO()	同　内部使用ﾒﾓﾘの解放	
2	DshInitTRCP_RENAME_INFO()	TRCP_RENAME_INFO 構造体の初期設定	S15F5
	DshFreeTRCP_RENAME_INFO()	同　内部使用ﾒﾓﾘの解放	
3	DshInitTRCP_S15F8_INFO()	TRCP_S15F8_INFO 構造体の初期設定	S15F8
	DshPutTRCP_S15F8_INFO()	同　ｴﾗｰ情報の設定	
	DshFreeTRCP_S15F8_INFO()	同　内部使用ﾒﾓﾘの解放	
4	DshInitTRCP_S15F10_INFO()	TRCP_S15F10_INFO 構造体の初期設定	S15F10
	DshPutTRCP_S15F10_INFO()	同　ｴﾗｰ情報の設定	
	DshFreeTRCP_S15F10_INFO()	同　内部使用ﾒﾓﾘの解放	
5	DshInitTRCP_INFO()	TRCP_INFO 構造体の初期設定	S15F13
	DshPutTRCP_PARA()	同　ﾊﾟﾗﾒｰﾀ情報の設定	S16F11
	DshFreeTRCP_INFO()	同　内部使用ﾒﾓﾘの解放	S16F15
6	DshInitTRCP_ERR_INFO()	TRCP_NFO 構造体の初期設定	S15F14
	DshPutTRCP_ERR_INFO()	同　ｴﾗｰ情報の設定	
	DshFreeTRCP_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	
7	DshInitTRCP_RETRIEVE_INFO()	TRCP_RETRIEVE_INFO 構造体の初期設定	S15F17
	DshFreeTRCP_RETRIEVE_INFO()	同　内部使用ﾒﾓﾘの解放	
8	DshInitTRCP_S15F18_INFO()	TRCP_S15F18_INFO 構造体の初期設定	S15F18
	DshPutTRCP_S15F18_M_SECNM_INFO()	同　ｾｸｼｮﾝ情報を設定	
	DshPutTRCP_M_SECNM_ATTR()	同　ｾｸｼｮﾝ属性情報を設定	
	DshPutTRCP_S15F18_ERR()	同　ｴﾗｰ情報の設定	
	DshFreeTRCP_S15F18_INFO()	同　内部使用ﾒﾓﾘの解放	

２．５．１　DshInitTRCP_ACT_INFO() – レシピ・アクション構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTRCP_ACT_INFO(
        TRCP_ACT_INFO  *info,          // TRCP_ACT_INFO 構造体のﾎﾟｲﾝﾀ
        char           *rcpid,         // ﾚｼﾋﾟ ID
        int            cmd             // ﾚｼﾋﾟ・ｱｸｼｮﾝ・ｺﾏﾝﾄﾞ
);
```

[VB.Net]

```
Sub DshInitTRCP_ACT_INFO(
        ByRef info As TRCP_ACT_INFO,
        rcpid  As String,
        cmd    As Integer)
```

[C#]

```
void DshInitTRCP_ACT_INFO(
        ref TRCP_ACT_INFO info,
        string rcpid,
        int    cmd );
```

（2）引数

info

TRCP_ACT_INFO 構造体のポインタです。

rcpid

レシピ ID です。

cmd

レシピ・アクション・コマンドです。

（3）戻り値

なし。

（4）説明

本関数は、TRCP_ACT_INFO 構造体を初期設定するために使用します。
構造体内には、レシピ ID とレシピ・アクション・コマンドを保存します。

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

（5）構造体

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

２．５．２　DshFreeTRCP_ACT_INFO() －　レシピ・アクション構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_ACT_INFO(
        TRCP_ACT_INFO  *info              // ﾒﾓﾘを開放したいﾚｼﾋﾟｱｸｼｮﾝ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

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

[C#]

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

（2）引数

info

メモリを解放したいレシピ・アクション構造体のポインタです。

（3）戻り値

なし。

（4）説明

TRCP_ACT_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TRCP_ACT_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．５．３　DshInitTRCP_RENAME_INFO() – レシピ・リネーム構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTRCP_RENAME_INFO(
        TRCP_RENAME_INFO  *info,        // TRCP_RENAME_INFO 構造体のﾎﾟｲﾝﾀ
        char        *rmnsspec,          // ﾚｼﾋﾟ ID
        char        *rmnewns            // ﾚｼﾋﾟ・ｱｸｼｮﾝ・ｺﾏﾝﾄﾞ
);
```

[VB.Net]

```
Sub DshInitTRCP_RENAME_INFO(
        ByRef info As TRCP_RENAME_INFO,
        rmnsspec As String,
        rmnewns  As String)
```

[C#]

```
void DshInitTRCP_RENAME_INFO(
        ref TRCP_RENAME_INFO info,
        string  rmnsspec,
        string  rmnewns );
```

（2）引数

info

TRCP_RENAME_INFO 構造体のポインタです。

rmnsspec

現在のレシピ名です。

rmnewns

新しいレシピ名です。

（3）戻り値

なし。

（4）説明

本関数は、TRCP_RENAME_INFO 構造体を初期設定するために使用します。
構造体内には、現レシピ名（ID）と新レシピ名を保存します。

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

（5）構造体

```
typedef struct{
        char     *rmnsspec;      // rmnsspec
        char     *rmnsrmnewns;   // action command 1=create,5=delete
}TRCP_RENAME_INFO;
```

２．５．４　DshFreeTRCP_RENAME_INFO() –レシピ・リネーム構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_RENAME_INFO(
        TRCP_RENAME_INFO  *info              // ﾒﾓﾘを開放したいﾚｼﾋﾟ・ﾘﾈｰﾑ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

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

[C#]

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

（2）引数

info

メモリを解放したいレシピ・リネーム構造体のポインタです。

（3）戻り値

なし。

（4）説明

TRCP_RENAME_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TRCP_RENAME_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．５．５　DshInitTRCP_S15F8_INFO() – レシピ・スペースデータ構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTRCP_S15F8_INFO(
        TRCP_S15F8_INFO  *erinfo,       // TRCP_S15F8_INFO 構造体のﾎﾟｲﾝﾀ
        UINT             rmspace,       // ﾚｼﾋﾟ・ｽﾍﾟｰｽﾃﾞｰﾀ
        int              rmack,         // RMACK
        int              err_count      // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTRCP_S15F8_INFO(
        ByRef erinfo As TRCP_S15F8_INFO,
        rmspace   As Integer,
        rmack     As Integer,
        err_count As Integer)
```

[C#]

```
void DshInitTRCP_S15F8_INFO(
        ref TRCP_S15F8_INFO erinfo,
        uint     rmspace,
        int      rmack,
        int      err_count );
```

（2）引数

erinfo

TRCP_S15F8_INFO 構造体のポインタです。

rmspace

レシピスペースデータです。

rmack

RMACK です。

err_count

エラー情報数です。(TERR_INFO)

（3）戻り値

なし。

（4）説明

本関数は、TRCP_S15F8_INFO 構造体を初期設定するために使用します。

構造体内には、レシピ・スペースデータ、 RMACK ならびに err_count 分のエラー情報 (TERR_INFO 構造体) を保存します。

構造体の使用が済んだら、DshFreeTRCP_S15F8_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        ULONG       rmspace;
        int         rmack;              // U1
        int         err_count;
        TERR_INFO   **err_list;
} TRCP_S15F8_INFO;
```

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

２．５．６　DshPutTRCP_S15F8_INFO() – エラー情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTRCP_S15F8_INFO(
      TRCP_S15F8_INFO   *erinfo,          // TRCP_S15F8_INFO 構造体のﾎﾟｲﾝﾀ
      int     err_code,                   // ｴﾗｰｺｰﾄﾞ
      char    *err_text                   // ｴﾗｰﾃｷｽﾄ
      );
```

[VB.Net]

```
Function DshPutTRCP_S15F8_INFO(
      ByRef erinfo As TRCP_S15F8_INFO,
      err_code  As Integer,
      err_text  As String) As Integer
```

[C#]

```
int DshPutTRCP_S15F8_INFO(
      ref TRCP_S15F8_INFO erinfo,
      int     err_code,
      string err_text);
```

（2）引数

erinfo

　　TRCP_S15F8_INFO 構造体のポインタです。

err_code

　　エラーコードです。

err_text

　　エラーテキストです。

（3）戻り値

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

（4）説明

先に説明した DshInitTRCP_S15F8_INFO()関数で初期設定された erinfo 内に、1 個のエラー情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．５．７　DshFreeTRCP_S15F8_INFO() –レシピ・スペースデータ構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_S15F8_INFO(
        TRCP_S15F8_INFO  *erinfo              // ﾒﾓﾘを開放したいﾚｼﾋﾟｽﾍﾟｰｽﾃﾞｰﾀ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTRCP_S15F8_INFO(
        ByRef erinfo As TRCP_S15F8_INFO )
```

[C#]

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

（2）引数

erinfo

メモリを解放したいレシピ・スペースデータ構造体のポインタです。

（3）戻り値

なし。

（4）説明

TRCP_S15F8_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TRCP_S15F8_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

２．５．８　DshInitTRCP_S15F10_INFO() – レシピ・ステータスデータ構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTRCP_S15F10_INFO(
        TRCP_S15F10_INFO  *erinfo,      // TRCP_S15F10_INFO 構造体のﾎﾟｲﾝﾀ
        int          rcpstate,          // ﾚｼﾋﾟ・ｽﾃｰﾀｽﾃﾞｰﾀ
        char         *rcpver,           // Recipe Version
        int          rmack,             // RMACK
        int          err_count          // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTRCP_S15F10_INFO(
        ByRef erinfo As TRCP_S15F10_INFO,
        rcpstate  As Integer,
        rcpver    As String,
        rmack     As Integer,
        err_count As Integer)
```

[C#]

```
void DshInitTRCP_S15F10_INFO(
        ref TRCP_S15F10_INFO erinfo,
        int     rcpstate,
        string  rcpver,
        int     rmack,
        int err_count );
```

（2）引数

erinfo

TRCP_S15F10_INFO 構造体のポインタです。

rcpstate

レシピステータスデータです。

rcpver

レシピバージョンです。

rmack

RMACK です。

err_count

エラー情報数です。(TERR_INFO()

（3）戻り値

なし。

（4）説明

本関数は、TRCP_S15F10_INFO 構造体を初期設定するために使用します。

構造体内には、レシピ・ステータスデータ、 レシピバージョン、RMACK ならびに err_count 分のエラー情報（TERR_INFO 構造体）を保存します。

構造体の使用が済んだら、DshFreeTRCP_S15F10_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        int         rcpstat;            // status U1
        char        *rcpver;            // version A
        int         rmack;              // U1
        int         err_count;
        TERR_INFO   **err_list;
} TRCP_S15F10_INFO;
```

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

２．５．９　DshPutTRCP_S15F10_INFO() – エラー情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTRCP_S15F10_INFO(
      TRCP_S15F10_INFO   *erinfo,          // TRCP_S15F10_INFO 構造体のﾎﾟｲﾝﾀ
      int    err_code,                     // ｴﾗｰｺｰﾄﾞ
      char   *err_text                     // ｴﾗｰﾃｷｽﾄ
      );
```

[VB.Net]

```
Function DshPutTRCP_S15F10_INFO(
      ByRef erinfo As TRCP_S15F10_INFO,
      err_code As Integer,
      err_text As String) As Integer
```

[C#]

```
int DshPutTRCP_S15F10_INFO(
      ref TRCP_S15F10_INFO erinfo,
      int    err_code,
      string err_text);
```

（2）引数

erinfo

　　TRCP_S15F10_INFO 構造体のポインタです。

err_code

　　エラーコードです。

err_text

　　エラーテキストです。

（3）戻り値

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

（4）説明

先に説明した DshInitTRCP_S15F10_INFO()関数で初期設定された erinfo 内に、1 個のエラー情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．５．10　DshFreeTRCP_S15F10_INFO() –レシピ・ステータスデータ構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_S15F10_INFO(
        TRCP_S15F10_INFO  *erinfo              // ﾒﾓﾘを開放したいﾚｼﾋﾟｽﾃｰﾀｽﾃﾞｰﾀ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTRCP_S15F10_INFO(
        ByRef erinfo As TRCP_S15F10_INFO )
```

[C#]

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

（2）引数

erinfo

メモリを解放したいレシピ・ステータスデータ構造体のポインタです。

（3）戻り値

なし。

（4）説明

TRCP_S15F10_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TRCP_S15F10_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

２．５．11 DshInitTRCP_INFO() – レシピ情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTRCP_INFO(
        TRCP_INFO  *info,              // TRCP_INFO 構造体のﾎﾟｲﾝﾀ
        char        *rcpspec,          // RCPID
        char        *rcpbody,          // RCPBODY
        int         para_count         // ﾊﾟﾗﾒｰﾀの数（最大）
);
```

[VB.Net]

```
Sub DshInitTRCP_INFO(
        ByRef info As TRCP_INFO,
        rcpspec    As String,
        rcpbody    As String,
        para_count As Integer)
```

[C#]

```
void DshInitTRCP_INFO(
        ref TRCP_INFO info,
        string  rcpspec,
        string  rcpbody,
        int     para_count );
```

（2）引数

info

TRCP_INFO 構造体のポインタです。

rcpspec

レシピ ID です。

rcpbody

RCPBODY です。

para_count

保存するパラメータパラメータ数です。

（3）戻り値

なし。

（4）説明

本関数は、TRCP_INFO 構造体を初期設定するために使用します。

構造体内には、レシピ ID、PPBODY、para_count 分のパラメータ情報（TRCP_PARA 構造体）を保存します。

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

（5）構造体

```
typedef struct{
        char        *rcpspec;
        int         para_count;         // # of pparameter
        TRCP_PARA   **para_list;
        char        *rcpbody;
}TRCP_INFO;
```

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

２．５．12　DshPutTRCP_INFO() – パラメータ情報の追加

（1）呼出書式

[c,C++]

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

[VB.Net]

```
Function DshPutTRCP_INFO(
        ByRef info      As TRCP_INFO,
        ByRef rcpparnm As String,
        para_fmt    As Integer,
        para_size   As Integer,
        rcpparval   As IntPtr ) As Integer
```

[C#]

```
int DshPutTRCP_INFO(
        ref TRCP_INFO info,
        string  rcpparnm,
        int     para_fmt,
        int     para_size,
        IntPtr  rcpparval);
```

（2）引数

info

TRCP_INFO 構造体のポインタです。

rcpparnm

パラメータ名です。

para_fmt

パラメータ値のフォーマットです。

para_size

パラメータの配列サイズです。

rcpparaval

パラメータ値のポインタです。

（3）戻り値

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

（4）説明

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

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．５．13　DshFreeTRCP_INFO() –レシピ情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_INFO(
        TRCP_INFO  *info                // ﾒﾓﾘを開放したいﾚｼﾋﾟ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

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

[C#]

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

（2）引数

info

メモリを解放したいレシピ情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TRCP_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TRCP_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．５．14 DshInitTRCP_ERR_INFO() – レシピエラー情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTRCP_ERR_INFO(
        TRCP_ERR_INFO  *erinfo,         // TRCP_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        int         rmack,              // RMACK
        int         err_count           // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTRCP_ERR_INFO(
        ByRef erinfo As TRCP_ERR_INFO,
        rmack     As Integer,
        err_count As Integer)
```

[C#]

```
void DshInitTRCP_ERR_INFO(
        ref TRCP_ERR_INFO erinfo,
        int     rmack,
        int     err_count );
```

（2）引数

erinfo

TRCP_ERR_INFO 構造体のポインタです。

rmack

RMACK です。

err_count

エラー情報数です。(TERR_INFO)

（3）戻り値

なし。

（4）説明

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

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

（5）構造体

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

２．５．15 DshPutTRCP_ERR_INFO() – エラー情報の追加

（1）呼出書式

[c,C++]

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

[VB.Net]

```
Function DshPutTRCP_ERR_INFO(
        ByRef erinfo As TRCP_ERR_INFO,
        err_code As Integer,
        err_text As String) As Integer
```

[C#]

```
int DshPutTRCP_ERR_INFO(
        ref TRCP_ERR_INFO erinfo,
        int     err_code,
        string  err_text);
```

（2）引数

erinfo

TRCP_ERR_INFO 構造体のポインタです。

err_code

エラーコードです。

err_text

エラーテキストです。

（3）戻り値

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

（4）説明

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

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．５．16　DshFreeTRCP_ERR_INFO() – レシピエラー情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_ERR_INFO(
        TRCP_ERR_INFO  *erinfo              // ﾒﾓﾘを開放したいﾚｼﾋﾟｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTRCP_ERR_INFO(
        ByRef erinfo As TRCP_ERR_INFO )
```

[C#]

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

（2）引数

erinfo

メモリを解放したいレシピエラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TRCP_ERR_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TRCP_ERR_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

２．５．17 DshInitTRCP_RETRIEVE_INFO() – レシピ検索情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTRCP_RETRIEVE_INFO(
        TRCP_RETRIEVE_INFO  *info,      // TRCP_RETRIEVE_INFO 構造体のﾎﾟｲﾝﾀ
        char        *rcpspec,           // ﾚｼﾋﾟ ID
        int         rcpseccode          // ﾚｼﾋﾟｾｸｼｮﾝｺｰﾄﾞ
);
```

[VB.Net]

```
Sub DshInitTRCP_RETRIEVE_INFO(
        ByRef info As TRCP_RETRIEVE_INFO,
        rcpspec    As String,
        rcpseccode As Integer)
```

[C#]

```
void DshInitTRCP_RETRIEVE_INFO(
        ref TRCP_RETRIEVE_INFO info,
        string  rcpspec,
        int     rcpseccode );
```

（2）引数

info

TRCP_RETRIEVE_INFO 構造体のポインタです。

rcpspec

レシピ ID です。

rcpseccode

レシピセクションコードです。

（3）戻り値

なし。

（4）説明

本関数は、TRCP_RETRIEVE_INFO 構造体を初期設定するために使用します。
構造体内には、レシピ ID とレシピセクションコードを保存します。

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

（5）構造体

```
ttypedef struct{
        char        *rcpspec;           // rcpid
        int         seccode;            // sec code
 }TRCP_RETRIEVE_INFO;
```

２．５．18 DshFreeTRCP_RETRIEVE_INFO() – レシピ検索構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_RETRIEVE_INFO(
        TRCP_RETRIEVE_INFO  *info              // ﾒﾓﾘを開放したいﾚｼﾋﾟ検索情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTRCP_RETRIEVE_INFO(
        ByRef info As TRCP_RETRIEVE_INFO )
```

[C#]

```
void DshFreeTRCP_RETRIEVE_INFO(
        ref TRCP_RETRIEVE_INFO info );
```

（2）引数

info

メモリを解放したいレシピ検索情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TRCP_RETRIEVE_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TRCP_RETRIEVE_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．５．19 DshInitTRCP_S15F18_INFO() – レシピ検索データ構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTRCP_S15F18_INFO(
        TRCP_S15F18_INFO  *erinfo,      // TRCP_S15F18_INFO 構造体のﾎﾟｲﾝﾀ
        int         q_count,            // ｾｸｼｮﾝ数  q=1,2 or 3
        int         r_count,            //包括的ｾｸｼｮﾝ名数 r = 0 or 2
        char        *rcpbody,           // RCPBODY
        int         m_count,            // ｴｰｼﾞｪﾝﾄ固有ﾃﾞｰﾀｾｯﾄ数
        int          rmack,             // RMACK
        int          err_count          // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTRCP_S15F18_INFO(
        ByRef erinfo As TRCP_S15F18_INFO,
        q_count   As Integer,
        r_count   As Integer,
        rcpbody   As String,
        m_count   As Integer,
        rmack     As Integer,
        err_count As Integer)
```

[C#]

```
void DshInitTRCP_S15F18_INFO(
        ref TRCP_S15F18_INFO erinfo,
        int    q_count,
        int    r_count,
        string rcpbody.
        int    m_count,
        int    rmack,
        int    err_count );
```

（2）引数

erinfo

TRCP_S15F18_INFO 構造体のポインタです。

q_count

セクション数です。 q=1,2 or 3 の値です。(値を正しく設定してください)

r_count

包括的セクション名数です。 r=0 or 2 の値です。(値を正しく設定してください)

rcpbody

RCPBODY です。

m_count

エージェント固有のデータセット数

rmack

RMACK です。

err_count

エラー情報数です。(TERR_INFO)

（3）戻り値

なし。

（4）説明

本関数は、TRCP_S15F18_INFO 構造体を初期設定するために使用します。

構造体内には、レシピ検索データとして、包括的属性数、固有のデータセット数、RMACK ならびに err_count 分のエラー情報（TERR_INFO 構造体）を保存します。

構造体の使用が済んだら、DshFreeTRCP_S15F18_INFO()関数によって内部で使用したメモリを解放してください。

（5）構造体

```
typedef struct{
        int         q_count;                // ++ 1, 2 or 3
        int         r_count;
        TRCP_SECNM  *m_secnm;               // ++
        char        *rcpbody;
        int         m_count;
        TRCP_SECNM  **secnm_list;
        int         rmack;
        int         err_count;
        TERR_INFO   **err_list;
} TRCP_S15F18_INFO;
```

```
typedef struct{
        char        *rcpsecnm;
        int         attr_count;
        TRCP_ATTR   **attr_list;
} TRCP_SECNM;
```

```
typedef struct{
        char        *attrid;
        int         format;
        int         asize;
        void        *attrdata;
} TRCP_ATTR;
```

２．５．20　DshPutTRCP_S15F18_M_SECNM_INFO() – 包括的セクション情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTRCP_S15F18_M_SECNM_INFO(
        TRCP_S15F18_INFO   *erinfo,            // TRCP_S15F18_INFO 構造体のﾎﾟｲﾝﾀ
        char   *rcpsecname,                    // ｾｸｼｮﾝ名
        int    attrcount                       // 属性数
        );
```

[VB.Net]

```
Function DshPutTRCP_S15F18_M_SECNM_INFO(
        ByRef erinfo As TRCP_S15F18_INFO,
        rcpsecname   As String,
        attrcount    As Integer) As Integer
```

[C#]

```
int DshPutRCP_S15F18_M_SECNM_INFO(
        ref TRCP_S15F18_INFO erinfo,
        string    rcpsecname,
        int       attrcount);
```

（2）引数

erinfo

TRCP_S15F18_INFO 構造体のポインタです。

rcpsecname

セクション名です。

attrcount

属性データの個数です。

（3）戻り値

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

（4）説明

先に説明した DshInitTRCP_S15F18_INFO()関数で初期設定された erinfo 内に、1 個のエラー情報を追加します。

TRCP_S15F18_INFO 構造内の m_secm メンバーの設定に適用されます。ただし、r_count＝2 の場合のみ有効です。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．５．21　DshPutTRCP_M_SECNM_ATTR() – 包括的セクション情報の属性追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTRCP_M_SECNM_ATTR(
        TRCP_S15F18_INFO    *erinfo,            // TRCP_S15F18_INFO 構造体のﾎﾟｲﾝﾀ
        int     order,                          // m_secrm ﾒﾝﾊﾞｰの位置(0,1)
        char    *attrid,                        // 属性名
        int     fmt,                            // 属性ﾃﾞｰﾀのﾌｫｰﾏｯﾄ
        int     size,                           // 属性ﾃﾞｰﾀの配列ｻｲｽﾞ
        void    *val                            // 属性値が格納されている領域のﾎﾟｲﾝﾀ);
```

[VB.Net]

```
Function DshPutTRCP_M_SECNM_ATTR(
        ByRef erinfo As TRCP_S15F18_INFO,
        order   As Integer,
        attrid  As String,
        fmt     As Integer,
        size    As Integer,
        val     As Integer) As Integer) As Integer
```

[C#]

```
int DshPutRCP_S15F18_M_SECNM_INFO(
        ref TRCP_S15F18_INFO erinfo,
        int         order,
        string      attrid,
        int         fmt,
        int         size,
        IntPtr      val);
```

（2）引数

erinfo

TRCP_S15F18_INFO 構造体のポインタです。

order

格納位置の順位です。

attrid

セクション名です。

attrid

属性 ID です。

fmt

属性値のフォーマットです。

size

属性値の配列サイズです。

val

属性値の格納ポインタです。

（3）戻り値

戻り値	意味

0	正常に追加できた。
(-1)	属性数を超えていた。

（4）説明

先に説明したDshInitTRCP_S15F18_INFO()関数で初期設定されたerinfo内のm_secrmメンバー内に、1個の属性情報を追加します。格納位置は orderで指定されます。

DshPutTRCP_S15F18_M_SECNM_INFO()関数で指定した属性数(attrcount)分を超える数の属性情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．５．22 DshPutTRCP_S15F18_SECNM_INFO() – エージェント固有セクション情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTRCP_S15F18_SECNM_INFO(
      TRCP_S15F18_INFO   *erinfo,          // TRCP_S15F18_INFO 構造体のﾎﾟｲﾝﾀ
      char   *rcpsecname,                  // ｾｸｼｮﾝ名
      int    attrcount                     // 属性数
      );
```

[VB.Net]

```
Function DshPutTRCP_S15F18_SECNM_INFO(
      ByRef erinfo As TRCP_S15F18_INFO,
      rcpsecname  As String,
      attrcount   As Integer) As Integer
```

[C#]

```
int DshPutRCP_S15F18_M_SECNM_INFO(
      ref TRCP_S15F18_INFO erinfo,
      string    rcpsecname,
      int       attrcount);
```

（2）引数

erinfo

TRCP_S15F18_INFO 構造体のポインタです。

rcpsecname

セクション名です。

attrcount

属性データの個数です。

（3）戻り値

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

（4）説明

先に説明した DshInitTRCP_S15F18_INFO()関数で初期設定された erinfo 内に、1 個のエラー情報を追加します。

TRCP_S15F18_INFO 構造内の secnm_list メンバーの設定に適用されます。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．５．23　DshPutTRCP_SECNM_ATTR() – 固有セクション情報の属性追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTRCP_SECNM_ATTR(
        TRCP_S15F18_INFO   *erinfo,          // TRCP_S15F18_INFO 構造体のﾎﾟｲﾝﾀ
        int     order,                       // secrm_list ﾒﾝﾊﾞｰの位置
        char    *attrid,                     // 属性名
        int     fmt,                         // 属性ﾃﾞｰﾀのﾌｫｰﾏｯﾄ
        int     size,                        // 属性ﾃﾞｰﾀの配列ｻｲｽﾞ
        void    *val                         // 属性値が格納されている領域のﾎﾟｲﾝﾀ);
```

[VB.Net]

```
Function DshPutTRCP_SECNM_ATTR(
        ByRef erinfo As TRCP_S15F18_INFO,
        order   As Integer,
        attrid  As String,
        fmt     As Integer,
        size    As Integer,
        val     As Integer) As Integer) As Integer
```

[C#]

```
int DshPutRCP_S15F18_M_SECNM_INFO(
        ref TRCP_S15F18_INFO erinfo,
        int       order,
        string    attrid,
        int       fmt,
        int       size,
        IntPtr    val);
```

（2）引数

erinfo

TRCP_S15F18_INFO 構造体のポインタです。

order

格納位置の順位です。

attrid

セクション名です。

attrid

属性 ID です。

fmt

属性値のフォーマットです。

size

属性値の配列サイズです。

val

属性値の格納ポインタです。

（3）戻り値

戻り値	意味

0	正常に追加できた。
(-1)	属性数を超えていた。

（4）説明

先に説明したDshInitTRCP_S15F18_INFO()関数で初期設定されたerinfo内のsecrm_listメンバー内に、1個の属性情報を追加します。格納位置はorderで指定されます。

DshPutTRCP_S15F18_SECNM_INFO()関数で指定した属性数(attrcount)分を超える数の属性情報を追加しようとした場合、戻り値として(-1)が返却されます。

２．５．24　DshPutTRCP_S15F18_ERR() – レシピ検索エラー情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTRCP_S15F18_ERR(
      TRCP_S15F18_ERR   *erinfo,          // TRCP_S15F18_ERR 構造体のﾎﾟｲﾝﾀ
      int     err_code,                   // ｴﾗｰｺｰﾄﾞ
      char    *err_text                   // ｴﾗｰﾃｷｽﾄ
      );
```

[VB.Net]

```
Function DshPutTRCP_S15F18_ERR(
      ByRef erinfo As TRCP_S15F18_ERR,
      err_code  As Integer,
      err_text  As String) As Integer
```

[C#]

```
int DshPutTRCP_S15F18_ERR(
      ref TRCP_S15F18_ERR erinfo,
      int     err_code,
      string  err_text);
```

（2）引数

erinfo

　　TRCP_S15F18_ERR 構造体のポインタです。

err_code

　　エラーコードです。

err_text

　　エラーテキストです。

（3）戻り値

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

（4）説明

先に説明した DshInitTRCP_S15F18_ERR)関数で初期設定された erinfo 内に、1 個のエラー情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．５．25　DshFreeTRCP_S15F18_INFO() –レシピ検索データ構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTRCP_S15F18_INFO(
        TRCP_S15F18_INFO  *erinfo              // ﾒﾓﾘを開放したいﾚｼﾋﾟｽﾃｰﾀｽﾃﾞｰﾀ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTRCP_S15F18_INFO(
        ByRef erinfo As TRCP_S15F18_INFO )
```

[C#]

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

（2）引数

erinfo

メモリを解放したいレシピ検索データ構造体のポインタです。

（3）戻り値

なし。

（4）説明

TRCP_S15F18_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TRCP_S15F18_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

２．６　PRJ　プロセス・ジョブ関連関数

ここでは、プロセスジョブ関連の関数について説明します。

以下の関数があります。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTPRJ_CMD_INFO	TPRJ_CMD_INFO 構造体の初期設定	S16F5
	DshPutTPRJ_CMD_INFO	同　ｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ情報の設定	
	DshFreeTPRJ_CMD_INFO	同　内部使用ﾒﾓﾘの解放	
2	DshInitTPRJ_CMD_ERR_INFO	TPRJ_CMD_ERR_INFO 構造体の初期設定	S16F6
	DshPutTPRJ_CMD_ERR_INFO	同　ｴﾗｰ情報の設定	
	DshFreeTPRJ_CMD_ERR_INFO	同　内部使用ﾒﾓﾘの解放	
3	DshInitTPRJ_INFO()	TPRJ_INFO 構造体の初期設定	S16F11 S16F15
	DshPutPrjRcpInfo()	同　ﾌﾟﾛｾｽ・ｼﾞｮﾌﾞ情報の設定	
	DshPutPrjCarInfo()	同　ｷｬﾘｱ情報の設定	
	DshPutPrjPauseCeid()	同　PAUSE 時 CEID 設定	
	DshFreeTPRJ_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTCAR_INFO()	TCAR_INFO 構造体の初期設定	
	DshFreeTCAR_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTCAR_SLOT_INFO()	TCAR_SLOT_INFO 構造体の初期設定	
	DshPutTCAR_SLOT_INFO()	同　ｽﾛｯﾄ ID 設定	
	DshFreeTSLOT_INFO()	同　内部使用ﾒﾓﾘの解放	
4	DshInitTPRJ_LIST()	TPRJ_LIST 構造体の初期設定	S16F15
	DshPutTPRJ_LIST()	同　TPRJ_INFO を設定	
	DshFreeTPRJ_INFO()	同　内部使用ﾒﾓﾘの解放	
5	DshInitTPRJ_ERR_INFO()	TPRJ_ERR_INFO 構造体の初期設定	S16F12 S16F16 S16F18
	DshPutTPRJ_ERR_INFO()	同　ｴﾗｰ情報の設定	
	DshFreeTPRJ_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshPutTPRJ_ERR_PRJID()	同　ｴﾗｰ対象 PRJID 設定(S16F15 のみ)	

２．６．１　DshInitTPRJ_CMD_INFO() – プロセス・ジョブ・コマンド構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPRJ_CMD_INFO
        TPRJ_CMD_INFO  *info,          // TPRJ_CMD_INFO 構造体のﾎﾟｲﾝﾀ
        char           *prjobid,       // ﾌﾟﾛｾｽ・ｼﾞｮﾌﾞ ID
        char           *cmd,           // ﾌﾟﾛｾｽ・ｼﾞｮﾌﾞ・ｺﾏﾝﾄﾞ
        int             cp_count       // 付属ﾊﾟﾗﾒｰﾀ数
);
```

[VB.Net]

```
Sub DshInitTPRJ_CMD_INFO
        ByRef info As TPRJ_CMD_INFO,
        prjobid  As String,
        cmd      As String,
        cp_count  As Integer)
```

[C#]

```
void DshInitTPRJ_CMD_INFO
        ref TPRJ_CMD_INFO info,
        string  prjobid,
        string  cmd,
        int     cp_count );
```

（2）引数

info

TPRJ_CMD_INFO 構造体のポインタです。

prjobid

プロセス・ジョブ ID です。

cmd

プロセス・ジョブ・コマンドです。

cp_count

付属パラメータ数です。

（3）戻り値

なし。

（4）説明

本関数は、TPRJ_CMD_INFO 構造体を初期設定するために使用します。
構造体内には、プロセス・ジョブ ID とプロセス・ジョブ・コマンドならびにパラメータ情報を保存します。

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

（5）構造体

```
typedef struct{
        char        *prjobid;
        char        *cmd;
        int         cmd_index;
        int         cp_count;
        TCMD_PARA   **cp_list;
} TPRJ_CMD_INFO;
```

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

２．６．２　DshPutTPRJ_CMD_INFO() – パラメータ情報の追加

（1）呼出書式

[c,C++]

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

[VB.Net]

```
Function DshPutTPRJ_CMD_INFO(
        ByRef info As TPRJ_CMD_INFO,
        ByRef cpname As String,
        cpval_fmt  As Integer,
        cpval_size As Integer,
        cpval      As IntPtr ) As Integer
```

[C#]

```
int DshPutTPRJ_CMD_INFO(
        ref TPRJ_CMD_INFO info,
        string  cpname,
        int     cpval_fmt,
        int     cpval_size,
        IntPtr  cpval_);
```

（2）引数

info

TPRJ_CMD_INFO 構造体のポインタです。

cpname

パラメータ名です。

cpval_fmt

パラメータ値のフォーマットです。

cpval_size

パラメータの配列サイズです。

cpval

パラメータ値のポインタです。

（3）戻り値

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

（4）説明

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

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．６．３　DshFreeTPRJ_CMD_INFO() － プロセス・ジョブ・コマンド構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPRJ_CMD_INFO
        TPRJ_CMD_INFO  *info              // ﾒﾓﾘを開放したいﾌﾟﾛｾｽ・ｼﾞｮﾌﾞ・ｺﾏﾝﾄﾞ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPRJ_CMD_INFO
        ByRef info As TPRJ_CMD_INFO )
```

[C#]

```
void DshFreeTPRJ_CMD_INFO
        ref TPRJ_CMD_INFO info );
```

（2）引数

info

メモリを解放したいプロセス・ジョブ・コマンド構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPRJ_CMD_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPRJ_CMD_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．６．４　DshInitTPRJ_CMD_ERR_INFO() – プロセス・ジョブ・コマンドエラー情報の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPRJ_CMD_ERR_INFO(
        TPRJ_CMD_ERR_INFO  *erinfo,    // TPRJ_CMD_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        char        *prjobid,          // ﾌﾟﾛｾｽｼﾞｮﾌﾞ ID
        int         acka,              // ACKA
        int         err_count          // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTPRJ_CMD_ERR_INFO(
        ByRef erinfo As TPRJ_CMD_ERR_INFO,
        prjobid   As String,
        acka      As Integer,
        err_count As Integer)
```

[C#]

```
void DshInitTPRJ_CMD_ERR_INFO(
        ref TPRJ_CMD_ERR_INFO erinfo,
        string  prjobid,
        int     acka,
        int     err_count );
```

（2）引数

erinfo

TPRJ_CMD_ERR_INFO 構造体のポインタです。

prjobid

プロセス・ジョブ ID です。

acka

ACKA です。

err_count

エラー情報の保存数です。(TERR_INFO)

（3）戻り値

なし。

（4）説明

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

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

（5）構造体

```
typedef struct{
        int         acka;              // U1
        int         err_count;
        TERR_INFO   **err_list;
} TPRJ_CMD_ERR_INFO;
```

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

２．６．５　DshPutTPRJ_CMD_ERR_INFO() – エラー情報の追加

（1）呼出書式

[c,C++]

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

[VB.Net]

```
Function DshPutTPRJ_CMD_ERR_INFO(
        ByRef erinfo As TPRJ_CMD_ERR_INFO,
        err_code  As Integer,
        err_text  As String) As Integer
```

[C#]

```
int DshPutTPRJ_CMD_ERR_INFO(
        ref TPRJ_CMD_ERR_INFO erinfo,
        int     err_code,
        string err_text);
```

（2）引数

erinfo

TPRJ_CMD_ERR_INFO 構造体のポインタです。

err_code

エラーコードです。

err_text

エラーテキストです。

（3）戻り値

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

（4）説明

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

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．６．６　DshFreeTPRJ_CMD_ERR_INFO() – ﾌﾟﾛｾｽ・ｼﾞｮﾌﾞ・ｺﾏﾝﾄﾞｴﾗｰ情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPRJ_CMD_ERR_INFO(
        TPRJ_CMD_ERR_INFO  *erinfo              // ﾒﾓﾘを開放したいﾚｼﾋﾟｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPRJ_CMD_ERR_INFO(
        ByRef erinfo As TPRJ_CMD_ERR_INFO )
```

[C#]

```
void DshFreeTPRJ_CMD_ERR_INFO(
        ref TPRJ_CMD_ERR_INFO erinfo );
```

（2）引数

erinfo

メモリを解放したいプロセス・ジョブ・コマンドエラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPRJ_CMD_ERR_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPRJ_CMD_ERR_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

２．６．７　DshInitTPRJ_INFO() – プロセス・ジョブ情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPRJ_INFO
        TPRJ_INFO  *info,              // TPRJ_INFO 構造体のﾎﾟｲﾝﾀ
        char       *prjobid,           // ﾌﾟﾛｾｽ・ｼﾞｮﾌﾞ ID
        int        mf,                 // MF carrier / material (13/14)ｺｰﾄﾞ
        int        m_count,            // material id ｶｳﾝﾄ
        int        prrecipemethod,     // PRRECIPEMETHOD   ( 1 or 2)
        int        prprocessstart,
        int        ceid_count
);
```

[VB.Net]

```
Sub DshInitTPRJ_INFO
        ByRef info      As TPRJ_INFO,
        prjobid         As String,
        mf              As Integer,
        m_count         As Integer,
        prrecipemethod  As Integer,
        prprocessstart  As Integer,
        ceid            As Integer
        )
```

[C#]

```
void DshInitTPRJ_INFO
        ref        TPRJ_INFO info,
        string     prjobid,
        int        mf,
        int        m_count,
        int        prrecipemethod,
        int        prprocessstart,
        int        ceid_count);
```

（2）引数

info

　　TPRJ_INFO 構造体のポインタです。

prjobid

　　プロセス・ジョブ ID です。

mf

　　MF 材料ｺｰﾄﾞ　carrier / material (13/14)ｺｰﾄﾞ

m_count

　　材料数です。

prrecipemethod

　　PRRECIPEMETHOD (1 or 2)

prprocessstart

　　自動/手動

ceid_count
　　PUASE イベント数

（3）戻り値
なし。

（4）説明
本関数は、TPRJ_INFO 構造体を初期設定するために使用します。
構造体内には、プロセス・ジョブ ID その他諸情報を保存します。

（5）の構造体のメンバーの中で、MF の値によって、mf=13 の場合は、car_count, car_list そして、mf=14 のときは、mid_count, mid_list を使用します。

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

（5）構造体

```
typedef struct{
        char        *prjobid;
        int         mf;
        int         car_count;          // mf=13 のとき使用
        TCAR_INFO   **car_list;         //  〃
        int         mid_count;          // mf=14 のとき使用
        char        **mid_list;         //  〃
        int         prrecipemethod;     // fmt=51(8) U1
        TRCP_INFO   *rcp_info;
        int         prprocessstart;     // fmt 11(8) Bool 1=auto,0=man
        int         ceid_count;
        TCEID       *pause_ceid_list;
} TPRJ_INFO;
```

TRCP_INFO 構造体については、２．５．11-（5）を参照してください。

２．６．８　DshPutPrjRcpInfo() – レシピ情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutPrjRcpInfo(
        TPRJ_INFO    *info,        // TPRJ_INFO 構造体のﾎﾟｲﾝﾀ
        TRCP_INFO    *rcp_info     // ﾚｼﾋﾟ情報構造体のﾎﾟｲﾝﾀ
        );
```

[VB.Net]

```
Function DshPutPrjRcpInfo(
        ByRef info     As TPRJ_INFO,
        ByRef rcp_info As TRCP_INFO ) As Integer
```

[C#]

```
int DshPutPrjRcpInfo(
        ref TPRJ_INFO info,
        ref TRCP_INFO rcp_info);
```

（2）引数

info

TPRJ_INFO 構造体のポインタです。

rcp_info

レシピ情報構造体のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	rcp_info == が NULL であった。

（4）説明

先に説明した DshInitTPRJ_INFO()関数で初期設定された info 内のメンバー rcp_info に引数 rcp_info で指定されたレシピ情報を設定します。

２．６．９　DshPutPrjCarInfo() – キャリア情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutPrjCarInfo(
      TPRJ_INFO   *info,        // TPRJ_INFO 構造体のﾎﾟｲﾝﾀ
      TCAR_INFO   *car_info     // ｷｬﾘｱ報構造体のﾎﾟｲﾝﾀ
      );
```

[VB.Net]

```
Function DshPutPrjCarInfo(
      ByRef info     As TPRJ_INFO,
      ByRef car_info As TCAR_INFO ) As Integer
```

[C#]

```
int DshPutPrjCarInfo(
      ref TPRJ_INFO  info,
      ref TCAR_INFO  car_info);
```

（2）引数

info

TPRJ_INFO 構造体のポインタです。

car_info

キャリア情報構造体のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	car_info == が NULL であった。

（4）説明

先に説明した DshInitTPRJ_INFO()関数で初期設定された info 内のメンバー car_info に引数 car_info で指定されたキャリア情報を設定します。

２．６．10　DshPutPrjMid() – MID 情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutPrjMid(
        TPRJ_INFO   *info,        // TPRJ_INFO 構造体のﾎﾟｲﾝﾀ
        char        *mid          // MID (material ID)
        );
```

[VB.Net]

```
Function DshPutPrjMid(
        ByRef info As TPRJ_INFO,
        mid  As String ) As Integer
```

[C#]

```
int DshPutPrjMid(
        ref TPRJ_INFO info,
        string  mid);
```

（2）引数

info

TPRJ_INFO 構造体のポインタです。

mid

MID (material ID) です。

（3）戻り値

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

（4）説明

先に説明したDshInitTPRJ_INFO()関数で初期設定されたinfo内のメンバー　midに引数midで指定されたMID を設定します。

追加によって、本関数が呼び出される順番にMID が mid_list 内に保存されます。

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

２．６．11　DshPutPrjPauseCeid() – CEID 情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutPrjPauseCeid(
      TPRJ_INFO   *info,       // TPRJ_INFO 構造体のﾎﾟｲﾝﾀ
      TCEID       ceid         // CEID (Colection Event ID)
      );
```

[VB.Net]

```
Function DshPutPrjPauseCeid(
      ByRef info As TPRJ_INFO,
      ceid As UInteger ) As Integer
```

[C#]

```
int DshPutPrjPauseCeid(
      ref TPRJ_INFO info,
      uint  ceid);
```

（2）引数

info

TPRJ_INFO 構造体のポインタです。

ceid

CEID(収集イベント ID)です。

（3）戻り値

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

（4）説明

先に説明した DshInitTPRJ_INFO()関数で初期設定された info 内のメンバー pasue_ceid_list に引数 ceid で指定された CEID を設定します。

追加によって、本関数が呼び出される順番に CEID が pasue_ceid_list 内に保存されます。

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

２．６．12　DshFreeTPRJ_INFO() －　プロセス・ジョブ情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPRJ_INFO
        TPRJ_INFO  *info                // ﾒﾓﾘを開放したいﾌﾟﾛｾｽ・ｼﾞｮﾌﾞ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPRJ_INFO
        ByRef info As TPRJ_INFO )
```

[C#]

```
void DshFreeTPRJ_INFO
        ref TPRJ_INFO info );
```

（2）引数

info

メモリを解放したいプロセス・ジョブ情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPRJ_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPRJ_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．６．13　DshInitTCAR_INFO() – キャリア情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTCAR_INFO(
        TCAR_INFO  *info,           // TCAR_INFO 構造体のﾎﾟｲﾝﾀ
        char       *carid,          // ﾚｼﾋﾟ ID
        char       *usage,          // USAGE（使用しませんので、NULL にしてください。）
        int        map_status,      // =0 にしてください。
        int        id_status,       // =0 にしてください。
        int        acc_status,      // =0 にしてください。
        char       *location,       // LOCATION（使用しませんので、NULL にしてください。）
        int        slot_count       //ｽﾛｯﾄｶｳﾝﾄ（使用します。）
);
```

[VB.Net]

```
Sub DshInitTCAR_INFO(
        ByRef info  As TCAR_INFO,
        carid       As String,
        usage       As String,
        map_status  As integer,
        id_status   As integer,
        acc_status  As integer,
        location    As String,
        slot_count  As Integer)
```

[C#]

```
void DshInitTCAR_INFO(
        ref TCAR_INFO info,
        string  carid,
        string  usage,
        int     map_status,
        int     id_status,
        int     acc_status,
        string  location,
        int     slot_count );
```

（2）引数

info

TCAR_INFO 構造体のポインタです。

carid

レシピ ID です。

usage

USAGE です。(NULL にしてください。)

map_status, id_status, acc_status

キャリアの状態情報ですが、 全て=0　を設定してください。

slot_count

スロット数です。

（3）戻り値

なし。

（4）説明

本関数は、TCAR_INFO 構造体を初期設定するために使用します。

構造体内に、キャリア ID, スロット情報を保存します。

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

（5）構造体

```
typedef struct{
        int         capacity;
        char        *usage;
        char        *carid;
        int         map_status;
        int         id_status;
        int         acc_status;
        char        *location;
        int         slot_count;
        TSLOT_INFO  **slot_list;
} TCAR_INFO;
```

```
typedef struct{
        int         status;
        int         slotid;             // U1
        char        *mid;
        char        *substid;
        char        *substloc;
} TSLOT_INFO;
```

注) usage, map_status, id_status, acc_status, location は GEM-PRO では使用しません。

２．６．14　DshPutTCAR_SLOT_INFO() – キャリアスロット情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTCAR_SLOT_INFO(
        TCAR_INFO       *info,      // TCAR_INFO 構造体のﾎﾟｲﾝﾀ
        TCAR_SLOT_INFO  *sinfo      // ｷｬﾘｱｽﾛｯﾄ情報構造体のﾎﾟｲﾝﾀ
        );
```

[VB.Net]

```
Function DshPutTCAR_SLOT_INFO(
        ByRef info  As TCAR_INFO,
        ByRef sinfo As TCAR_SLOT_INFO ) As Integer
```

[C#]

```
int DshPutTCAR_SLOT_INFO(
        ref TCAR_INFO       info,
        ref TCAR_SLOT_INFO  sinfo);
```

（2）引数

info

TCAR_INFO 構造体のポインタです。

sinfo

キャリアスロット情報構造体のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	TCAR_INFO 内の slot_list 配列が満杯であった。

（4）説明

先に説明したDshInitTCAR_INFO()関数で初期設定されたinfo内のメンバー slot_list配列に、引数sinfoで指定されたキャリアスロット情報を設定します。

slot_list配列に対しては、設定順にリストに格納されます。

２．６．15　DshFreeTCAR_INFO() –キャリア構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTCAR_INFO(
        TCAR_INFO  *info              // ﾒﾓﾘを開放したいｷｬﾘｱ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTCAR_INFO(
        ByRef info As TCAR_INFO )
```

[C#]

```
void DshFreeTCAR_INFO(
        ref TCAR_INFO info );
```

（2）引数

info

メモリを解放したいキャリア情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TCAR_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TCAR_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．６．16 DshInitTCAR_SLOT_INFO() – キャリア・スロット情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTCAR_SLOT_INFO(
        TCAR_SLOT_INFO  *info,          // TCAR_SLOT_INFO 構造体のﾎﾟｲﾝﾀ
        int             slotid,         // SlotID
        char            *mid,           // MID です。
        char            *substid,       // 基板 ID です
        char            *substloc       // 基板のﾛｹｰｼｮﾝです。
);
```

[VB.Net]

```
Sub DshInitTCAR_SLOT_INFO(
        ByRef info As TCAR_SLOT_INFO,
        slotid     As Integer,
        mid        As String,
        substid    As String,
        substloc   As String)
```

[C#]

```
void DshInitTCAR_SLOT_INFO(
        ref TCAR_SLOT_INFO info,
        int             slotid,
        string          mid,
        string          substid,
        string          substloc );
```

（2）引数

info

TCAR_SLOT_INFO 構造体のポインタです。

carid

レシピ ID です。

slotid

Slot ID です。

mid

Material ID です。

substid

基板 ID です。

substloc

基板のロケーションです。

（3）戻り値

なし。

（4）説明

本関数は、キャリアの 1 個のスロットの情報を TCAR_SLOT_INFO 構造体内に設定するために使用します。
構造体内には、引数で与えられた情報を各メンバーに設定します。

２．６．17　DshFreeTCAR_SLOT_INFO() –キャリアスロット構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTCAR_SLOT_INFO(
        TCAR_SLOT_INFO  *info              // ﾒﾓﾘを開放したいｷｬﾘｱｽﾛｯﾄ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTCAR_SLOT_INFO(
        ByRef info As TCAR_SLOT_INFO )
```

[C#]

```
void DshFreeTCAR_SLOT_INFO(
        ref TCAR_SLOT_INFO info );
```

（2）引数

info

メモリを解放したいキャリアスロット情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TCAR_SLOT_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TCAR_SLOT_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．６．18　DshInitTPRJ_LIST() – PRJ リスト構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPRJ_LIST(
        TPRJ_LIST  *list,              // TPRJ_LIST 構造体のﾎﾟｲﾝﾀ
        int         prj_count          // PRJ 数
);
```

[VB.Net]

```
Sub DshInitTPRJ_LIST(
        ByRef list As TPRJ_LIST,
        prj_count As Integer)
```

[C#]

```
void DshInitTPRJ_LIST(
        ref TPRJ_LIST list,
        int prj_count );
```

（2）引数

list

TPRJ_LIST 構造体のポインタです。

prj_count

保存する最大 PRJ 数です。

（3）戻り値

なし。

（4）説明

本関数は、TPRJ_LIST 構造体を初期設定するために使用します。

構造体内には、prj_count 分の TPRJ_INFO 構造体の PRJ 情報を保存します。
（TPRJ_INFO の操作については、２．６．７以降を参照してください。

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

（5）構造体

```
typedef struct{
        int         prj_count;
        TPRJ_INFO   **prj_list;
} TPRJ_LIST;
```

２．６．19　DshPutTPRJ_LIST() – PRJ リストへの追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTPRJ_LIST(
        TPRJ_LIST   *list,          // TPRJ_LIST 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        TPRJ_INFO   *info           // 追加する TPRJ_INFO 構造体のﾎﾟｲﾝﾀ
        );
```

[VB.Net]

```
Function DshPutTPRJ_LIST(
        ByRef list As TPRJ_LIST,
        ByRef into As TPRJ_INFO) As Integer
```

[C#]

```
int DshPutTPRJ_LIST(
        ref TPRJ_LIST list,
        ref TPRJ_INFO info);
```

（2）引数

list

TPRJ_LIST 構造体のポインタです。

info

TPRJ_INFO 構造体のポインタです。（追加したい PRJ 情報が保存されている。）

（3）戻り値

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

（4）説明

先に説明した DshInitTPRJ_LIST()関数で初期設定された list 内に、1 個の PRJ 情報を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．６．20　DshFreeTPRJ_LIST() – PRJ リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPRJ_LIST(
        TPRJ_LIST  *list              // ﾒﾓﾘを開放したいPRJﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPRJ_LIST(
        ByRef list As TPRJ_LIST )
```

[C#]

```
void DshFreeTPRJ_LIST(
        ref TPRJ_LIST list );
```

（2）引数

list

メモリを解放したいPRJ構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPRJ_LIST構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPRJ_LISTの内容を 全て0 で初期設定します。
listがNULLならば、何も処理しません。

２．６．21　DshInitTPRJ_ERR_INFO() – プロセス・ジョブエラー情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPRJ_ERR_INFO(
        TPRJ_ERR_INFO  *erinfo,        // TPRJ_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        int         prj_count,         // PRJ 数
        int         acka,              // ACKA
        int         err_count          // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTPRJ_ERR_INFO(
        ByRef erinfo As TPRJ_ERR_INFO,
        prj_count As Integer,
        acka      As Integer,
        err_count As Integer)
```

[C#]

```
void DshInitTPRJ_ERR_INFO(
        ref TPRJ_ERR_INFO erinfo,
        int     prj_count,
        int     acka,
        int     err_count );
```

（2）引数

erinfo

TPRJ_ERR_INFO 構造体のポインタです。

prj_count

保存する PRJID 数です。

acka

ACKA です。

err_count

エラー情報数です。(TERR_INFO)

（3）戻り値

なし。

（4）説明

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

構造体内には、対象となる PRJ 数、ACKA と err_count の数だけのエラー情報（TPRJ_ERR_INFO 構造体）を保存します。

S16F11, S16F15 両方の応答情報に使用します。

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

（5）構造体

```
typedef struct{
        int         prj_count;
        char        **prj_list;
        int         acka;           // 0=false, 1=true
        int         err_count;
        TERR_INFO   **err_list;
} TPRJ_ERR_INFO;
```

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

２．６．22　DshPutTPRJ_ERR_INFO() – エラー情報の追加

（1）呼出書式

[c,C++]

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

[VB.Net]

```
Function DshPutTPRJ_ERR_INFO(
        ByRef erinfo As TPRJ_ERR_INFO,
        err_code As Integer,
        err_text As String) As Integer
```

[C#]

```
int DshPutTPRJ_ERR_INFO(
        ref TPRJ_ERR_INFO erinfo,
        int     err_code,
        string err_text);
```

（2）引数

erinfo

TPRJ_ERR_INFO 構造体のポインタです。

err_code

エラーコードです。

err_text

エラーテキストです。

（3）戻り値

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

（4）説明

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

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．６．23　DshFreeTPRJ_ERR_INFO() –プロセス・ジョブエラー情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPRJ_ERR_INFO(
        TPRJ_ERR_INFO  *erinfo              // ﾒﾓﾘを開放したいﾚｼﾋﾟｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPRJ_ERR_INFO(
        ByRef erinfo As TPRJ_ERR_INFO )
```

[C#]

```
void DshFreeTPRJ_ERR_INFO(
        ref TPRJ_ERR_INFO erinfo );
```

（2）引数

erinfo

メモリを解放したいプロセス・ジョブエラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPRJ_ERR_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPRJ_ERR_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

２．６．24　DshPutTPRJ_ERR_PRJID() – エラー情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTPRJ_ERR_PRJID(
        TPRJ_ERR_PRJID   *erinfo,              // TPRJ_ERR_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        char             *prjid                // ﾌﾟﾛｾｽ・ｼﾞｮﾌﾞ ID
        );
```

[VB.Net]

```
Function DshPutTPRJ_ERR_PRJID(
        ByRef  erinfo As TPRJ_ERR_INFO,
        prjid  As  String) As Integer
```

[C#]

```
int DshPutTPRJ_ERR_PRJID(
        ref TPRJ_ERR_INFO erinfo,
        string  prjid);
```

（2）引数

erinfo

TPRJ_ERR_INFO 構造体のポインタです。

prjid

プロセスジョブ ID です。

（3）戻り値

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

（4）説明

先に説明した DshInitTPRJ_ERR_INFO()関数で初期設定された erinfo 内の prj_list 配列に１個のプロセスジョブ ID を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

prj_count 分を超える数のプロセスジョブ ID を追加しようとした場合、戻り値として(-1)が返却されます。

２．６．25　DshInitTPRJ_DEQ_INFO() – プロセスジョブ DEQ 構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPRJ_DEQ_INFO(
        TPRJ_DEQ_INFO  *info,                   // TPRJ_DEQ_INFO 構造体のﾎﾟｲﾝﾀ
        int         prj_count                   // プロセスジョブ ID 数
);
```

[VB.Net]

```
Sub DshInitTPRJ_DEQ_INFO(
        ByRef info As TPRJ_DEQ_INFO,
        prj_count As Integer)
```

[C#]

```
void DshInitTPRJ_DEQ_INFO(
        ref TPRJ_DEQ_INFO info,
        int prj_count );
```

（2）引数

info

TPRJ_DEQ_INFO 構造体のポインタです。

prj_count

保存する最大プロセスジョブ ID 数です。

（3）戻り値

なし。

（4）説明

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

構造体内には、prj_count 分のプロセスジョブ ID を保存します。

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

（5）構造体

```
typedef struct{
        int         prj_count;
        char        **prj_list;
} TPRJ_DEQ_INFO;
```

２．６．26　DshPutTPRJ_DEQ_INFO() – プロセスジョブ ID の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTPRJ_DEQ_INFO(
        TPRJ_DEQ_INFO   *info,          // TPRJ_DEQ_INFO 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        char    *prjid                  // 追加するプロセスジョブ ID
        );
```

[VB.Net]

```
Function DshPutTPRJ_DEQ_INFO(
        ByRef info As TPRJ_DEQ_INFO,
        prjid As string) As Integer
```

[C#]

```
int DshPutTPRJ_DEQ_INFO(
        ref TPRJ_DEQ_INFO info,
        string prjid);
```

（2）引数

info

TPRJ_DEQ_INFO 構造体のﾎﾟｲﾝﾀです。

prjid

プロセスジョブ ID（文字列）です。

（3）戻り値

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

（4）説明

先に説明した DshInitTPRJ_DEQ_INFO()関数で初期設定された info 内に、1 個のプロセスジョブ ID を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

prj_count 分を超える数のプロセスジョブ ID を追加しようとした場合、戻り値として(-1)が返却されます。

２．６．27　DshFreeTPRJ_DEQ_INFO() – プロセスジョブ ID 構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPRJ_DEQ_INFO(
      TPRJ_DEQ_INFO  *info              // ﾒﾓﾘを開放したいプロセスジョブ ID 構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPRJ_DEQ_INFO(
        ByRef info As TPRJ_DEQ_INFO )
```

[C#]

```
void DshFreeTPRJ_DEQ_INFO(
        ref TPRJ_DEQ_INFO info );
```

（2）引数

info

メモリを解放したいプロセスジョブ ID 構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPRJ_DEQ_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPRJ_DEQ_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．６．28 DshInitTPRJ_STATE_LIST() – プロセスジョブ状態リスト構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPRJ_STATE_LIST(
      TPRJ_STATE_LIST  *list,              // TPRJ_STATE_LIST 構造体のﾎﾟｲﾝﾀ
      int         prj_count                // プロセスジョブ ID 数
);
```

[VB.Net]

```
Sub DshInitTPRJ_STATE_LIST(
      ByRef list As TPRJ_STATE_LIST,
      prj_count As Integer)
```

[C#]

```
void DshInitTPRJ_STATE_LIST(
      ref TPRJ_STATE_LIST list,
      int prj_count );
```

（2）引数

list

TPRJ_STATE_LIST 構造体のポインタです。

prj_count

保存する最大プロセスジョブ ID 数です。

（3）戻り値

なし。

（4）説明

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

構造体内には、prj_count 分のプロセスジョブ ID とその状態値を保存します。

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

（5）構造体

```
typedef struct{
        int     prj_count;
        char    **prj_list;
        int     state_list;       // U1
} TPRJ_STATE_LIST;
```

２．６．29　DshPutTPRJ_STATE_LIST() – プロセスジョブ ID、状態値の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTPRJ_STATE_LIST(
      TPRJ_STATE_LIST   *list,      // TPRJ_STATE_LIST 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
      char    *prjid                // 追加するプロセスジョブ ID
      int     state                 // 状態値
      );
```

[VB.Net]

```
Function DshPutTPRJ_STATE_LIST(
      ByRef list As TPRJ_STATE_LIST,
      prjid  As string
      state  As Integer) As Integer
```

[C#]

```
int DshPutTPRJ_STATE_LIST(
      ref TPRJ_STATE_LIST list,
      string  prjid
      int     state);
```

（2）引数

list

　　TPRJ_STATE_LIST 構造体のﾎﾟｲﾝﾀです。

prjid

　　プロセスジョブ ID（文字列）です。

state

　　プロセスジョブの状態です。

（3）戻り値

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

（4）説明

先に説明した DshInitTPRJ_STATE_LIST()関数で初期設定された list 内に、1 個のプロセスジョブ ID とその状態値を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

prj_count 分を超える数のプロセスジョブ ID を追加しようとした場合、戻り値として(-1)が返却されます。

２．６．30　DshFreeTPRJ_STATE_LIST() – プロセスジョブ ID 状態リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPRJ_STATE_LIST(
        TPRJ_STATE_LIST  *list              // ﾒﾓﾘを開放したい構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPRJ_STATE_LIST(
        ByRef list As TPRJ_STATE_LIST )
```

[C#]

```
void DshFreeTPRJ_STATE_LIST(
        ref TPRJ_STATE_LIST list );
```

（2）引数

list

メモリを解放したいプロセスジョブ ID 構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPRJ_STATE_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPRJ_STATE_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。

2．7　CJ　コントロール・ジョブ関連関数

ここでは、コントロールジョブ関連の関数について説明します。

以下の関数があります。

	関数名	機能	関連ﾒｯｾｰｼﾞ
1	DshInitTCJ_INFO()	TCJ_INFO 構造体の初期設定	S14F9
	DshPutTCJ_ATTR_INFO()	同　属性情報を設定	S14F11
	DshFreeTCJ_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTCJ_TEXT_INFO()	TCJ_TEXT_INFO 構造体の初期設定	
	DshPutTCJ_TEXT_INFO()	同　文字列を設定	
	DshFreeTCJ_TEXT_INFO()	同　内部使用ﾒﾓﾘの解放	
	DshInitTVOID_LIST()	TVOID_LIST 構造体の初期設定	
	DshPutTVOID_LIST()	同　TVOID_LIST に属性情報構造体を設定	
	DshInitTMTRL_OUT_STAT()	TMTRL_OUT_STAT 構造体の初期設定	
	DshPutTMTRL_OUT_STAT()	同　status を 1 個設定	
	DshFreeTVOID_LIST_TMTRL_OUT_STAT()	同　内部使用ﾒﾓﾘの解放	
	DshInitTMTRL_OUT_SPEC()	TMTRL_OUT_SPEC 構造体の初期設定	
	DshPutTMTRL_OUT_SPECSrc	同 source ｷｬﾘｱの slotid を設定	
	DshPutTMTRL_OUT_SPECDst	同 destination ｷｬﾘｱの slotid を設定	
	DshInitTCTRL_SPEC()	TCTRL_SPEC 構造体の初期設定	
	DshPutTCTRL_RULE()	同　ｺﾝﾄﾛｰﾙ・ﾙｰﾙ情報を設定	
	DshPutTOUT_RULE()	同　ｺﾝﾄﾛｰﾙ・ｽﾍﾟｯｸ情報を設定	
	DshFreeTVOID_LIST_TCTRL_SPEC()	同　内部使用ﾒﾓﾘの解放	
	DshInitTPAUSE_EVENT()	TPAUSE_EVENT 構造体の初期設定	
	DshPutTPAUSE_EVENT()	同　ｲﾍﾞﾝﾄ ID を設定	
	DshFreeTPAUSE_EVENT()	同　内部使用ﾒﾓﾘの解放	
	DshInitTPRJ_STATE_LIST()	TPRJ_STATE_LIST 構造体の初期設定	2.6.28
	DshPutTPRJ_STATE_LIST()	同　process job status を設定	2.6.29
	DshFreeTPRJ_STATE_LIST()	同　内部使用ﾒﾓﾘの解放	2.6.30 参照
	DshInitTOBJ_ERR_INFO()	TOBJ_NFO 構造体の初期設定	S14F10, S14F12
	DshPutTOBJ_ERR_INFO()	同　ｴﾗｰ情報の設定	
	DshFreeTOBJ_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	
2	DshInitTCJ_CMD_INFO()	TCJ_CMD_INFO 構造体の初期設定	S16F27
	DshPutTCJ_CMD_INFO()	同　ｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ情報の設定	
	DshFreeTCJ_CMD_INFO()	同　内部使用ﾒﾓﾘの解放	
3	DshInitTCJ_CMD_ERR_INFO()	TCJ_CMD_ERR_INFO 構造体の初期設定	S16F28
	DshFreeTCJ_CMD_ERR_INFO()	同　内部使用ﾒﾓﾘの解放	

２．７．１　DshInitTCJ_INFO() – コントロール・ジョブ情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTCJ_INFO
        TCJ_INFO    *info,          // TCJ_INFO 構造体のﾎﾟｲﾝﾀ
        char        *cjid,          // ｺﾝﾄﾛｰﾙ・ｼﾞｮﾌﾞ ID
        char        objspec,        // OBJSPEC
        char        objtype,        // OBJTYPE
        int         attr_count      // 属性情報数
);
```

[VB.Net]

```
Sub DshInitTCJ_INFO
        ByRef info As TCJ_INFO,
        cjid       As String,
        objspec    As String,
        objtype    As String,
        attr_count As Integer
        )
```

[C#]

```
void DshInitTCJ_INFO
        ref         TCJ_INFO info,
        string      cjid,
        string      objspec,
        string      objtype,
        int         attr_count);
```

（2）引数

info

TCJ_INFO 構造体のポインタです。

cjid

コントロール・ジョブ ID です。

objspec

OBJSPEC です。

objtype

OBJTYPE です。

attr_count

属性数

（3）戻り値

なし。

（4）説明

本関数は、TCJ_INFO 構造体を初期設定するために使用します。
構造体内には、コントロール・ジョブ ID その他属性情報などを保存します。

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

（5）構造体

```
typedef struct{
        int             objspec_flag;       //  (内部処理用)
        char            *objspec;           // = cjid です。
        int             objtype_flag;
        char            *objtype;
        char            *objid;             // = cjid です。
        int             attr_count;
        TOBJ_ATTR_INFO  **attr_list;
} TCJ_INFO;
```

```
typedef struct{
        char            *attrid;
        int             attrid_index;
        void            *attrdata;
} TOBJ_ATTR_INFO;
```

attr_data が構造体の情報であった場合、以下の構造体を使用します。

・

```
typedef struct{
        int         mtrl_status;        // U1
        char        *carid;
        int         slot_count;
        int         *slotid_list;
} TMTRL_OUT_STAT;
```

・

```
typedef struct{
        char        *src_carid;
        int         src_slot_count;
        int         *src_slotid_list;

        char        *dst_carid;
        int         dst_slot_count;
        int         *dst_slotid_list;
} TMTRL_OUT_SPEC;
```

・

```
typedef struct{
        int         count;
        void        **void_list;
} TVOID_LIST;
```

•

```
typedef struct{
        char    *name;
        int     fmt;
        int     asize;
        void    *value;
} TCTRL_RULE;
```

•

```
typedef struct{
        int     status;                 // u1
        int     fmt;
        int     asize;
        void    *value;
} TOUT_RULE;
```

•

```
typedef struct{
        char        *prjobid;
        int         ctrl_rule_count;
        TCTRL_RULE  **ctrl_rule_list;
        int         out_rule_count;
        TOUT_RULE   **out_rule_list;
} TCTRL_SPEC;
```

•

```
typedef struct{
        int     prj_count;
        char    **prj_list;
        int     *state_list;                        // U1
} TPRJ_STATE_LIST;
```

•

```
typedef struct{
        int     ce_count;
        int     *ceid_list;
} TPAUSE_EVENT;
```

•

```
typedef struct{
        int         text_count;
        char        **text_list;
} TCJ_TEXT_INFO;
```

（6）属性インデクスと属性 ID

下に示す通りです。

属性ｲﾝﾃﾞｸｽ記号	ｲﾝﾃﾞｸｽ値	属性 ID
EN_ObjID	0	"ObjID",
EN_CarrierInputSpec	1	"CarrierInputSpec",
EN_CurrentPRJob	2	"CurrentPRJob",
EN_DataCollectionPlan	3	"DataCollectionPlan",
EN_MtrlOutByStatus	4	"MtrlOutByStatus",
EN_MtrlOutSpec	5	"MtrlOutSpec",
EN_PauseEvent	6	"PauseEvent",
EN_ProcessingCtrlSpec	7	"ProcessingCtrlSpec",
EN_ProcessingOrderMgmt	8	"ProcessOrderMgmt",
EN_PRJobStatusList	9	"PRJobStatusList",
EN_StartMethod	1	"StartMethod",0
EN_State	1	"State",1

属性ｲﾝﾃﾞｸｽ記号はプログラム言語によって、次のファイルに定義されています。

c, C++ ： DshGemProInfo.h
c# ： DshGemProInfo.cs
VB.Net ： DshGemProInfo.vb

２．７．２　DshPutTCJ_ATTR_INFO() – 属性情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTCJ_ATTR_INFO(
      TCJ_INFO         *info,                  // TCJ_INFO 構造体のﾎﾟｲﾝﾀ
      TOBJ_ATTR_INFO   *attr_info              // 属性情報構造体のﾎﾟｲﾝﾀ
      );
```

[VB.Net]

```
Function DshPutTCJ_ATTR_INFO(
      ByRef info      As TCJ_INFO,
      ByRef attr_info As TOBJ_ATTR_INFO ) As Integer
```

[C#]

```
int DshPutTCJ_ATTR_INFO(
      ref TCJ_INFO        info,
      ref TOBJ_ATTR_INFO  attr_info);
```

（2）引数

info

　　TCJ_INFO 構造体のポインタです。

attr_info

　　属性情報構造体のポインタです。

（3）戻り値

戻り値	意味
0	正常に追加できた。
(-1)	attr_info == が NULL であった。

（4）説明

先に説明した DshInitTCJ_INFO()関数で初期設定された info 内のメンバー attr_list に引数 attr_info で指定された属性情報を設定します。

情報は、本関数が実行される順に、attr_list の配列位置に設定されます。

２．７．３　DshFreeTCJ_INFO() －　コントロール・ジョブ情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTCJ_INFO
        TCJ_INFO  *info              // ﾒﾓﾘを開放したいｺﾝﾄﾛｰﾙ・ｼﾞｮﾌﾞ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTCJ_INFO
        ByRef info As TCJ_INFO )
```

[C#]

```
void DshFreeTCJ_INFO
        ref TCJ_INFO info );
```

（2）引数

info

メモリを解放したいコントロール・ジョブ情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TCJ_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TCJ_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．７．４　DshInitTCJ_TEXT_INFO() －　複数テキスト属性構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTCJ_TEXT_INFO(
      TCJ_TEXT_INFO  *info,                  // TCJ_TEXT_INFO 構造体のﾎﾟｲﾝﾀ
      int            text_count              // Text 数
);
```

[VB.Net]

```
Sub DshInitTCJ_TEXT_INFO(
      ByRef info As TCJ_TEXT_INFO,
      text_count As Integer)
```

[C#]

```
void DshInitTCJ_TEXT_INFO(
      ref TCJ_TEXT_INFO info,
      int               text_count );
```

（2）引数

info

TCJ_TEXT_INFO 構造体のポインタです。

text_count

保存する最大テキスト数です。

（3）戻り値

なし。

（4）説明

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

構造体内には、text_count 分のテキスト文字列を保存します。

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

２．７．５　DshPutTCJ_TEXT_INFO() – 複数テキストリストへの追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTCJ_TEXT_INFO(
        TCJ_TEXT_INFO   *info,          // TCJ_TEXT_INFO 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        char            *info           // 追加する TPRJ_INFO 構造体のﾎﾟｲﾝﾀ
        );
```

[VB.Net]

```
Function DshPutTCJ_TEXT_INFO(
        ByRef info As TCJ_TEXT_INFO,
        ByRef text As String) As Integer
```

[C#]

```
int DshPutTCJ_TEXT_INFO(
        ref TCJ_TEXT_INFO info,
        string            text);
```

（2）引数

info

TCJ_TEXT_INFO 構造体のポインタです。

text

追加するテキスト文字列です。

（3）戻り値

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

（4）説明

先に説明した DshInitTCJ_TEXT_INFO()関数で初期設定された info 内に、text の文字列を追加します。

追加によって、本関数が呼び出される順番にテキストが構造体内に保存されます。

text_count 分を超える数の文字列を追加しようとした場合、戻り値として(-1)が返却されます。

２．７．６　DshFreeTCJ_TEXT_INFO() – 複数テキスト属性構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTCJ_TEXT_INFO(
        TCJ_TEXT_INFO  *info              // ﾒﾓﾘを開放したいﾃｷｽﾄ属性構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTCJ_TEXT_INFO(
        ByRef info As TCJ_TEXT_INFO )
```

[C#]

```
void DshFreeTCJ_TEXT_INFO(
        ref TCJ_TEXT_INFO info );
```

（2）引数

info

メモリを解放したいTCJ_TEXT_INFO構造体のポインタです。

（3）戻り値

なし。

（4）説明

TCJ_TEXT_INFO構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TCJ_TEXT_INFOの内容を 全て0 で初期設定します。
infoがNULLならば、何も処理しません。

２．７．７　DshInitTVOID_LIST() － Void構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTVOID_LIST(
      TVOID_LIST  *list,          // TVOID_LIST 構造体のﾎﾟｲﾝﾀ
      int         void_count      // 保存する構造体の数
);
```

[VB.Net]

```
Sub DshInitTVOID_LIST(
      ByRef list As TVOID_LIST,
      void_count As Integer)
```

[C#]

```
void DshInitTVOID_LIST(
      ref TVOID_LIST list,
      int  void_count );
```

（2）引数

list

　　TVOID_LIST 構造体のポインタです。

void_count

　　保存する構造体の数です。

（3）戻り値

　なし。

（4）説明

　本関数は、TVOID_LIST 構造体を初期設定するために使用します。

　構造体内には、void_count 分の他の構造体を保存します。

　TVOID_LSIT は、次の３種類の属性情報のコンテナーとして使用されます。

構造体	属性インデクス
TMTRL_OUT_STAT	EN_MtrlOutByStatus
TMTRL_OUT_SPEC	EN_MtrlOutSpec
TCTRL_SPEC	EN_ProcessingCtrlSpec

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

２．７．８　DshPutTVOID_LIST() – 複数テキストリストへの追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTVOID_LIST(
      TVOID_LIST    *list,          // TVOID_LIST 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
      void          *void_info      // 追加する構造体のﾎﾟｲﾝﾀ
      );
```

[VB.Net]

```
Function DshPutTVOID_LIST(
      ByRef list As TVOID_LIST,
      void_info  As IntPtr) As Integer
```

[C#]

```
int DshPutTVOID_LIST(
      ref TVOID_LIST list,
      IntPtr  void_info);
```

（2）引数

list

TVOID_LIST 構造体のポインタです。

void_info

追加する構造体のポインタです。

（3）戻り値

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

（4）説明

先に説明した DshInitTVOID_LIST()関数で初期設定された list 内に、他の構造体を追加します。

追加によって、本関数が呼び出される順番にリストに保存されます。

void_count 分を超える数の構造体を追加しようとした場合、戻り値として(-1)が返却されます。

２．７．９　DshInitTMTRL_OUT_STAT() － TMTRL_OUT_STAT 構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTMTRL_OUT_STAT(
      TMTRL_OUT_STAT  *info,        // TMTRL_OUT_STAT 構造体のﾎﾟｲﾝﾀ
      int         mtrl_status,      // material status
      char        *carid,           // ｷｬﾘｱ ID
      int         slot_count        // ｽﾛｯﾄ数
);
```

[VB.Net]

```
Sub DshInitTMTRL_OUT_STAT(
      ByRef info  As TMTRL_OUT_STAT,
      mtrl_status As Integer,
      carid       As String,
      slot_count  As Integer)
```

[C#]

```
void DshInitTMTRL_OUT_STAT(
      ref TMTRL_OUT_STAT info,
      int     mtrl_status,
      string carid,
      int    slot_count );
```

（2）引数

info

TMTRL_OUT_STAT 構造体のポインタです。

mtrl_status

Material Status です。

carid

キャリア ID です。

slot_count

キャリアのスロット数です。

（3）戻り値

なし。

（4）説明

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

構造体内には、材料の状態、キャリア ID,　slot_count 分のスロット ID を保存します。

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

TMTRL_OUT_STAT 構造体の情報は、CJ の属性情報として設定されますが、実際には TVOID_LIST 構造体内のリストに保存されます。

（5）構造体

```
typedef struct{
        int         mtrl_status;        // U1
        char        *carid;
        int         slot_count;
        int         *slotid_list;
} TMTRL_OUT_STAT;
```

２．７．10　DshPutTMTRL_OUT_STAT() – TMTRLOUT_STAT への SLOTID 追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTMTRL_OUT_STAT(
      TMTRL_OUT_STAT   *info,        // TMTRL_OUT_STAT 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
      int   slotid                   // ｽﾛｯﾄ ID
      );
```

[VB.Net]

```
Function DshPutTMTRL_OUT_STAT(
      ByRef info As TMTRL_OUT_STAT,
      slotid  As Integer) As Integer
```

[C#]

```
int DshPutTMTRL_OUT_STAT(
      ref  TMTRL_OUT_STAT  info,
      Int  slotid);
```

（2）引数

info

　　TMTRL_OUT_STAT 構造体のポインタです。

slotid

　　追加するスロット ID です。

（3）戻り値

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

（4）説明

先に説明した DshInitTMTRL_OUT_STAT()関数で初期設定された info 構造体内の slotid_list の配列に追加します。

追加によって、本関数が呼び出される順番にリストに保存されます。

slot_count 分を超える数の構造体を追加しようとした場合、戻り値として(-1)が返却されます。

２．７．11　DshFreeTVOID_LIST_TMTRL_OUT_STAT() – TVOID 構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTVOID_LIST_TMTRL_OUT_STAT
        TVOID_LIST  *list               // ﾒﾓﾘを開放したい TVOID_LIST 構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTVOID_LIST_TMTRL_OUT_STAT
        ByRef list As T TVOID_LIST )
```

[C#]

```
void DshFreeTVOID_LIST_TMTRL_OUT_STAT
        ref TVOID_LIST list );
```

（2）引数

list

メモリを解放したい TVOID_LIST 構造体のポインタです。

（3）戻り値

なし。

（4）説明

TVOID_LIST 構造体に保存されている TMTRL_OUT_STAT 構造体内で情報格納用に使用されているメモリを全て解放します。

開放した後、TMTRL_OUT_STAT の内容を 全て 0 で初期設定し、更に TVOID_LIST 内の内容も全て 0 で初期設定します。

list が NULL ならば、何も処理しません。

２．７．12 DshInitTMTRL_OUT_SPEC() － TMTRL_OUT_SPEC 構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTMTRL_OUT_SPEC(
        TMTRL_OUT_SPEC  *info,          // TMTRL_OUT_SPEC 構造体のﾎﾟｲﾝﾀ
        char        *src_carid,         // src ｷｬﾘｱ ID
        int          src_slot_count     // src ｽﾛｯﾄ数
        char        *dst_carid,         // dst ｷｬﾘｱ ID
        int          dst_slot_count     // dst ｽﾛｯﾄ数
);
```

[VB.Net]

```
Sub DshInitTMTRL_OUT_SPEC(
        ByRef info      As TMTRL_OUT_SPEC,
        src_carid       As String,
        src_slot_count As Integer,
        dst_carid       As String,
        dst_slot_count As Integer)
```

[C#]

```
void DshInitTMTRL_OUT_SPEC(
        ref TMTRL_OUT_SPEC info,
        int         mtrl_SPECus,
        string      src_carid,
        int         src_slot_count,
        string      dst_carid,
        int         dst_slot_count );
```

（2）引数

info

TMTRL_OUT_SPEC 構造体のポインタです。

mtrl_SPECus

Material Status です。

src_carid

Source キャリア ID です。

src_slot_count

Source キャリアのスロット数です。

dst_carid

Destination キャリア ID です。

dst_slot_count

Destination キャリアのスロット数です。

（3）戻り値

なし。

（4）説明

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

構造体内には、ソース、デスティネーションのキャリア ID,　slot_count 分のスロット ID を保存します。

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

TMTRL_OUT_SPEC 構造体の情報は、CJ の属性情報として設定されますが、実際には TVOID_LIST 構造体内のリストに保存されます。

（5）構造体

```
typedef struct{
        char        *src_carid;
        int         src_slot_count;
        int         *src_slotid_list;
        char        *dst_carid;
        int         dst_slot_count;
        int         *dst_slotid_list;
} TMTRL_OUT_SPEC;
```

２．７．13 DshPutTMTRL_OUT_SPECSrc() – TMTRL_OUT_SPEC 構造体への Src SlotID 追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTMTRL_OUT_SPECSrc (
      TMTRL_OUT_SPEC   *info,       // TMTRL_OUT_SPEC 構造体のﾎﾟｲﾝﾀ
      int              slotid       // ｽﾛｯﾄ ID
      );
```

[VB.Net]

```
Function DshPutTMTRL_OUT_SPECSrc (
      ByRef info As TMTRL_OUT_SPEC,
      slotid  As Integer) As Integer
```

[C#]

```
int DshPutTMTRL_OUT_SPECSrc (
      ref  TMTRL_OUT_SPEC  info,
      Int  slotid);
```

（2）引数

info

TMTRL_OUT_SPEC 構造体のポインタです。

slotid

追加するスロット ID です。(Source 側のキャリア)

（3）戻り値

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

（4）説明

先に説明した DshInitTMTRL_OUT_SPEC()関数で初期設定された info 構造体内の src_slotid_list の配列に追加します。

追加によって、本関数が呼び出される順番にリストに保存されます。

src_slot_count 分を超える数の構造体を追加しようとした場合、戻り値として(-1)が返却されます。

２．７．14　DshPutTMTRL_OUT_SPECDst() – TMTRL_OUT_SPEC 構造体への Dst SlotID 追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTMTRL_OUT_SPECDst (
      TMTRL_OUT_SPEC   *info,        // TMTRL_OUT_SPEC 構造体のﾎﾟｲﾝﾀ
      int   slotid                   // ｽﾛｯﾄ ID
      );
```

[VB.Net]

```
Function DshPutTMTRL_OUT_SPECDst (
      ByRef info As TMTRL_OUT_SPEC,
      slotid  As Integer) As Integer
```

[C#]

```
int DshPutTMTRL_OUT_SPECDst (
      ref  TMTRL_OUT_SPEC  info,
      Int  slotid);
```

（2）引数

info

　　TMTRL_OUT_SPEC 構造体のポインタです。

slotid

　　追加するスロット ID です。(Destination 側のキャリア)

（3）戻り値

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

（4）説明

先に説明した DshInitTMTRL_OUT_SPEC()関数で初期設定された info 構造体内の dst_slotid_list の配列に追加します。

追加によって、本関数が呼び出される順番にリストに保存されます。

dst_slot_count 分を超える数の構造体を追加しようとした場合、戻り値として(-1)が返却されます。

２．７．15　DshFreeTVOID_LIST_TMTRL_OUT_SPEC() － TMTRL_OUR_SPEC 構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTVOID_LIST_TMTRL_OUT_SPEC
        TVOID_LIST  *list               // ﾒﾓﾘを開放したい TVOID_LIST 構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTVOID_LIST_TMTRL_OUT_SPEC
        ByRef list As T TVOID_LIST )
```

[C#]

```
void DshFreeTVOID_LIST_TMTRL_OUT_SPEC
        ref TVOID_LIST list );
```

（2）引数

list

メモリを解放したい TVOID_LIST 構造体のポインタです。

（3）戻り値

なし。

（4）説明

TVOID_LIST 構造体に保存されている TMTRL_OUT_SPEC 構造体内で情報格納用に使用されているメモリを全て解放します。

開放した後、TMTRL_OUT_SPEC の内容を 全て 0 で初期設定し、更に TVOID_LIST 内の内容も全て 0 で初期設定します。

list が NULL ならば、何も処理しません。

２．７．16 DshInitTCTRL_SPEC() － TCTRL_SPEC 構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTCTRL_SPEC(
        TCTRL_SPEC  *info,                  // TCTRL_SPEC 構造体のﾎﾟｲﾝﾀ
        char        *prjobid,               // ﾌﾟﾛｾｽ・ｼﾞｮﾌﾞ ID
        int         ctrl_rulre_count,       // Control Rule count
        int         out_rule_count          // Out Rule count
);
```

[VB.Net]

```
Sub DshInitTCTRL_SPEC(
        ByRef info      As TCTRL_SPEC,
        prjobid         As String,
        ctrl_rule_count As Integer,
        out_rule_count  As Integer)
```

[C#]

```
void DshInitTCTRL_SPEC(
        ref TCTRL_SPEC info,
        string      prjobid,
        int         ctrl_rule_count,
        int         out_rule_count );
```

（2）引数

info

TCTRL_SPEC 構造体のポインタです。

prjobid

Source キャリア ID です。

ctrl_rule_count

Source キャリアのスロット数です。

out_rule_count

Destination キャリアのスロット数です。

（3）戻り値

なし。

（4）説明

本関数は、TCTRL_SPEC 構造体を初期設定するために使用します。

構造体内には、プロセス・ジョブ ID と保存するコントロール・ルール、アウト・ルール情報の数を設定します。

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

TCTRL_SPEC 構造体の情報は、CJ の属性情報として設定されますが、実際には TVOID_LIST 構造体内のリス

トに保存されます。

（5）構造体

```
typedef struct{
        char        *prjobid;
        int         ctrl_rule_count;
        TCTRL_RULE  **ctrl_rule_list;
        int         out_rule_count;
        TOUT_RULE   **out_rule_list;
} TCTRL_SPEC;
```

```
typedef struct{
        char    *name;
        int     fmt;
        int     asize;
        void    *value;
} TCTRL_RULE;
```

```
typedef struct{
        int     status;                 // u1
        int     fmt;
        int     asize;
        void    *value;
} TOUT_RULE;
```

２．７．17　DshPutTCTRL_RULE() – TCTRL_SPEC 構造体への CTRL_RULE 追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTCTRL_RULE (
        TCTRL_SPEC    *info,          // TCTRL_SPEC 構造体のﾎﾟｲﾝﾀ
        char          *name,          // ﾙｰﾙ名
        int           fmt,            // ﾙｰﾙ値のﾌｫｰﾏｯﾄ
        int           asize,          // ﾙｰﾙ値の配列ｻｲｽﾞ
        void          *value          // ﾙｰﾙ値
        );
```

[VB.Net]

```
Function DshPutTCTRL_RULE (
        ByRef info As TCTRL_SPEC,
        name     As String,
        fmt      As Integer,
        asize    As Integer,
        value    As IntPtr) As Integer
```

[C#]

```
int DshPutTCTRL_RULE (
        ref  TCTRL_SPEC  info,
        string       name,
        int          fmt,
        int          asize,
        IntPtr       *value
```

（2）引数

info

TCTRL_SPEC 構造体のポインタです。

name

ルール名です。

fmt

ルール値のフォーマットです。

asize

ルール値の配列サイズです。

value

ルール値です。

（3）戻り値

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

（4）説明

先に説明した DshInitTCTRL_SPEC()関数で初期設定された info 構造体内の ctrl_rule_list の配列に追加します。

追加によって、本関数が呼び出される順番にリストに保存されます。

ctrl_rule_count 分を超える数の構造体を追加しようとした場合、戻り値として(-1)が返却されます。

２．７．18 DshPutTOUT_RULE() – TCTRL_SPEC 構造体への OUT_RULE 追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTOUT_RULE (
        TCTRL_SPEC *info,           // TCTRL_SPEC 構造体のﾎﾟｲﾝﾀ
        int        status,          // status
        int        fmt,             // ﾙｰﾙ値のﾌｫｰﾏｯﾄ
        int        asize,           // ﾙｰﾙ値の配列ｻｲｽﾞ
        void       *value           // ﾙｰﾙ値
        );
```

[VB.Net]

```
Function DshPutTOUT_RULE (
        ByRef info As TCTRL_SPEC,
        status    As Integer,
        fmt       As Integer,
        asize     As Integer,
        value     As IntPtr) As Integer
```

[C#]

```
int DshPutTOUT_RULE (
        ref  TCTRL_SPEC  info,
        int        status,
        int        fmt,
        int        asize,
        IntPtr     *value
```

（2）引数

info

TCTRL_SPEC 構造体のポインタです。

status

状態語です。

fmt

ルール値のフォーマットです。

asize

ルール値の配列サイズです。

value

ルール値です。

（3）戻り値

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

（4）説明

先に説明したDshInitTCTRL_SPEC()関数で初期設定された info 構造体内の out_rule_list の配列に追加します。

追加によって、本関数が呼び出される順番にリストに保存されます。

out_rule_count 分を超える数の構造体を追加しようとした場合、戻り値として(-1)が返却されます。

２．７．19　DshFreeTVOID_LIST_TCTRL_SPEC() – TVOID_LIST (TCTRL_SPEC)構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTVOID_LIST_TCTRL_SPEC
        TVOID_LIST  *list               // ﾒﾓﾘを開放したいTVOID_LIST構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTVOID_LIST_TCTRL_SPEC
        ByRef list As T TVOID_LIST )
```

[C#]

```
void DshFreeTVOID_LIST_TCTRL_SPEC
        ref TVOID_LIST list );
```

（2）引数

list

メモリを解放したいTVOID_LIST構造体のポインタです。

（3）戻り値

なし。

（4）説明

TVOID_LIST構造体に保存されているTCTRL_SPEC構造体内で情報格納用に使用されているメモリを全て解放します。

開放した後、TCTRL_SPECの内容を 全て0 で初期設定し、更にTVOID_LIST内の内容も全て0 で初期設定します。

listがNULLならば、何も処理しません。

２．７．20　DshInitTPAUSE_EVENT() – PAUSE EVENT 構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTPAUSE_EVENT(
        TPAUSE_EVENT  *info,                    // TPAUSE_EVENT 構造体のﾎﾟｲﾝﾀ
        int           ce_count                  // Collection Event 数
);
```

[VB.Net]

```
Sub DshInitTPAUSE_EVENT(
        ByRef info As TPAUSE_EVENT,
        ce_count As Integer)
```

[C#]

```
void DshInitTPAUSE_EVENT(
        ref TPAUSE_EVENT info,
        int ce_count );
```

（2）引数

info

TPAUSE_EVENT 構造体のポインタです。

ce_count

保存する CEID 数です。

（3）戻り値

なし。

（4）説明

本関数は、TPAUSE_EVENT 構造体を初期設定するために使用します。

構造体内には、ce_count 分の CEID を保存します。

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

（5）構造体

```
typedef struct{
        int     ce_count;
        int     *ceid_list;
} TPAUSE_EVENT;
```

２．７．21　DshPutTPAUSE_EVENT() – CEID の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutTPAUSE_EVENT(
      TPAUSE_EVENT   *info,          // TPAUSE_EVENT 構造体ﾘｽﾄのﾎﾟｲﾝﾀ
      int     ceid                   // CEID
      );
```

[VB.Net]

```
Function DshPutTPAUSE_EVENT(
      ByRef info As TPAUSE_EVENT,
      ceid As Integer) As Integer
```

[C#]

```
int DshPutTPAUSE_EVENT(
      ref TPAUSE_EVENT info,
      int  ceid);
```

（2）引数

info

TPAUSE_EVENT 構造体のﾎﾟｲﾝﾀです。

ceid

CEID です。

（3）戻り値

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

（4）説明

先に説明した DshInitTPAUSE_EVENT()関数で初期設定された info 内に、1 個の CEID を追加します。

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．７．22　DshFreeTPAUSE_EVENT() – PAUSE EVENT 構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTPAUSE_EVENT(
        TPAUSE_EVENT  *info              // ﾒﾓﾘを開放したい構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTPAUSE_EVENT(
        ByRef info As TPAUSE_EVENT )
```

[C#]

```
void DshFreeTPAUSE_EVENT(
        ref TPAUSE_EVENT info );
```

（2）引数

info

メモリを解放したい TPAUSE_EVENT 構造体のポインタです。

（3）戻り値

なし。

（4）説明

TPAUSE_EVENT 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPAUSE_EVENT の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．７．23　DshInitTOBJ_ERR_INFO() – オブジェクトエラー情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTOBJ_ERR_INFO(
        TOBJ_ERR_INFO  *erinfo,        // TOBJ_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        int         objack,            // OBJACK
        int         err_count          // ｴﾗｰﾊﾟﾗﾒｰﾀの数
);
```

[VB.Net]

```
Sub DshInitTOBJ_ERR_INFO(
        ByRef erinfo As TOBJ_ERR_INFO,
        objack    As Integer,
        err_count As Integer)
```

[C#]

```
void DshInitTOBJ_ERR_INFO(
        ref TOBJ_ERR_INFO erinfo,
        int      objack,
        int      err_count );
```

（2）引数

erinfo

TOBJ_ERR_INFO 構造体のポインタです。

objack

OBJACK です。

err_count

エラー情報数です。(TERR_INFO)

（3）戻り値

なし。

（4）説明

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

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

（5）構造体

```
typedef struct{
        int         objack;             // U1
        int         err_count;
        TERR_INFO   **err_list;
} TOBJ_ERR_INFO;
```

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

２．７．24　DshPutTOBJ_ERR_INFO() – エラー情報の追加

（1）呼出書式

[c,C++]

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

[VB.Net]

```
Function DshPutTOBJ_ERR_INFO(
        ByRef erinfo As TOBJ_ERR_INFO,
        err_code As Integer,
        err_text As String) As Integer
```

[C#]

```
int DshPutTOBJ_ERR_INFO(
        ref TOBJ_ERR_INFO erinfo,
        int     err_code,
        string err_text);
```

（2）引数

erinfo

TOBJ_ERR_INFO 構造体のポインタです。

err_code

エラーコードです。

err_text

エラーテキストです。

（3）戻り値

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

（4）説明

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

追加によって、本関数が呼び出される順番に値が構造体内に保存されます。

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

２．７．25 DshFreeTOBJ_ERR_INFO() – オブジェクトエラー情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTOBJ_ERR_INFO(
      TOBJ_ERR_INFO  *erinfo              // ﾒﾓﾘを開放したいｵﾌﾞｼﾞｪｸﾄｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTOBJ_ERR_INFO(
        ByRef erinfo As TOBJ_ERR_INFO )
```

[C#]

```
void DshFreeTOBJ_ERR_INFO(
        ref TOBJ_ERR_INFO erinfo );
```

（2）引数

erinfo

メモリを解放したいオブジェクトエラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TOBJ_ERR_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TOBJ_ERR_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

２．７．26 DshInitTCJ_CMD_INFO() – コントロール・ジョブ・コマンド構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTCJ_CMD_INFO
        TCJ_CMD_INFO  *info,          // TCJ_CMD_INFO 構造体のﾎﾟｲﾝﾀ
        char          *cjid,          // ｺﾝﾄﾛｰﾙ・ｼﾞｮﾌﾞ ID
        BYTE          cmd,            // ｺﾝﾄﾛｰﾙ・ｼﾞｮﾌﾞ・ｺﾏﾝﾄﾞ
);
```

[VB.Net]

```
Sub DshInitTCJ_CMD_INFO
        ByRef info As TCJ_CMD_INFO,
        cjid  As String,
        cmd   As Byte )
```

[C#]

```
void DshInitTCJ_CMD_INFO
        ref TCJ_CMD_INFO info,
        string  cjid,
        byte    cmd );
```

（2）引数

info

TCJ_CMD_INFO 構造体のポインタです。

cjid

コントロール・ジョブ ID です。

cmd

コントロール・ジョブ・コマンドです。

（3）戻り値

なし。

（4）説明

本関数は、TCJ_CMD_INFO 構造体を初期設定するために使用します。
構造体内には、コントロール・ジョブ ID とコントロール・ジョブ・コマンドならびにパラメータ情報を保存します。

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

（5）構造体

```
typedef struct{
        char        *ctljobid;
        int         cmd;                // 2008.6.10 A->U1 に訂正
        TCMD_PARA   *cp_info;
} TCJ_CMD_INFO;
```

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

２．７．27　DshPutTCJ_CMD_INFO() – パラメータ情報の追加

（1）呼出書式

[c,C++]

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

[VB.Net]

```
Function DshPutTCJ_CMD_INFO(
        ByRef info    As TCJ_CMD_INFO,
        ByRef cpname As String,
        cpval_fmt     As Integer,
        cpval_size    As Integer,
        cpval_        As IntPtr ) As Integer
```

[C#]

```
int DshPutTCJ_CMD_INFO(
        ref TCJ_CMD_INFO info,
        string  cpname,
        int     cpval_fmt,
        int     cpval_size,
        IntPtr  cpval_);
```

（2）引数

info

TCJ_CMD_INFO 構造体のポインタです。

cpname

パラメータ名です。

cpval_fmt

パラメータ値のフォーマットです。

cpval_size

パラメータの配列サイズです。

cpval

パラメータ値のポインタです。

（3）戻り値

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

（4）説明

先に説明した DshInitTCJ_CMD_INFO()関数で初期設定された info 内に、パラメータ情報を設定します。

２．７．28 DshFreeTCJ_CMD_INFO() － コントロール・ジョブ・コマンド構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTCJ_CMD_INFO
      TCJ_CMD_INFO  *info              // ﾒﾓﾘを開放したいｺﾝﾄﾛｰﾙ・ｼﾞｮﾌﾞ・ｺﾏﾝﾄﾞ構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTCJ_CMD_INFO
        ByRef info As TCJ_CMD_INFO )
```

[C#]

```
void DshFreeTCJ_CMD_INFO
        ref TCJ_CMD_INFO info );
```

（2）引数

info

メモリを解放したいコントロール・ジョブ・コマンド構造体のポインタです。

（3）戻り値

なし。

（4）説明

TCJ_CMD_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TCJ_CMD_INFO の内容を 全て 0 で初期設定します。
info が NULL ならば、何も処理しません。

２．７．29 DshInitTCJ_CMD_ERR_INFO() – CJ コマンドエラー情報構造体の初期設定

（1）呼出書式

[c,C++]

```
API int APIX DshInitTCJ_CMD_ERR_INFO(
        TCJ_CMD_ERR_INFO  *erinfo,      // TCJ_CMD_ERR_INFO 構造体のﾎﾟｲﾝﾀ
        int         acka,               // ACKA
        int         err_flag,           // ｴﾗｰ有無
        int         err_code,           // ｴﾗｰｺｰﾄﾞ
        char        *err_text           // ｴﾗｰﾃｷｽﾄ
);
```

[VB.Net]

```
Sub DshInitTCJ_CMD_ERR_INFO(
        ByRef erinfo As TCJ_CMD_ERR_INFO,
        acka     As Integer,
        err_flag As Integer,
        err_code As Integer,
        err_text As String )
```

[C#]

```
void DshInitTCJ_CMD_ERR_INFO(
        ref TCJ_CMD_ERR_INFO erinfo,
        int      acka,
        int      err_flag,
        int      err_code,
        string   err_text );
```

（2）引数

erinfo

TCJ_CMD_ERR_INFO 構造体のポインタです。

acka

ACKA です。

err_flag

エラー有無フラグです。(0=なし、1=あり、 1 の場合、err_code, err_text がｾｯﾄされます。)

err_code

エラーコードです。

err_text

エラーテキストです。

（3）戻り値

なし。

（4）説明

本関数は、TCJ_CMD_ERR_INFO 構造体を初期設定するために使用します。
構造体内には、ACKA とエラー情報（TERR_INFO 構造体）を保存します。

err_flag = 0 の場合、エラーがなかったということになります。

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

（5）構造体

```
typedef struct{
        int         acka;
        TERR_INFO   *err_info;
} TCJ_CMD_ERR_INFO;
```

２．７．30　DshFreeTCJ_CMD_ERR_INFO() – CJ コマンドエラー情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTCJ_CMD_ERR_INFO(
        TCJ_CMD_ERR_INFO  *erinfo              // ﾒﾓﾘを開放したいCJ ｺﾏﾝﾄﾞｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[VB.Net]

```
Sub DshFreeTCJ_CMD_ERR_INFO(
        ByRef erinfo As TCJ_CMD_ERR_INFO )
```

[C#]

```
void DshFreeTCJ_CMD_ERR_INFO(
        ref TCJ_CMD_ERR_INFO erinfo );
```

（2）引数

erinfo

メモリを解放したいCJ コマンドエラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

TCJ_CMD_ERR_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TCJ_CMD_ERR_INFO の内容を 全て 0 で初期設定します。
erinfo が NULL ならば、何も処理しません。

（ ２．８ 以降は Vol-2 に続く ）