

DSHEng4 装置通信エンジン (GEM+GEM300)
ソフトウェア・パッケージ

APP インタフェース
ライブラリ関数説明書
(C, C++, .Net-Vb,C#)

VOL- 1 3 / 1 5

3 . 2 1 CJ コントロールジョブ情報アクセスサービス関数

2 0 0 9 年 7 月

株式会社データマップ

[取り扱い注意]

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

【改訂履歴】

番号	改訂日付	項目	概略
1.	2009.7	初版	

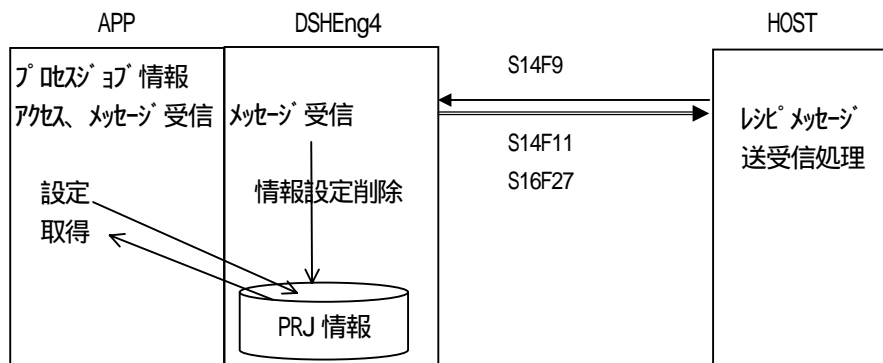
3.21 CJ コントロールジョブ情報アクセスサービスとメッセージ送信関数	1
3.21.1 使用する情報格納構造体.....	4
3.21.2 CJ コントロールジョブ情報アクセス、送信関数.....	8
3.21.2.1 EngAllocCjInfo() - コントロールジョブの登録.....	8
3.21.2.2 EngSetCjInfo() - コントロールジョブ情報の設定.....	9
EngSetCjInfoX() - インデクスでのコントロールジョブ情報の設定.....	9
3.21.2.3 EngGetCjInfo() - コントロールジョブ情報の取得.....	11
EngGetCjInfoX() - インデクス指定でのコントロールジョブ情報の取得.....	11
3.21.2.4 EngDelCjInfo() - コントロールジョブの削除.....	13
EngDelCjInfoX() - インデクスでのコントロールジョブの削除.....	13
3.21.2.5 EngSetCjState() - コントロールジョブ状態の設定.....	14
EngSetCjStateX() - インデクスでのコントロールジョブ状態の設定.....	14
3.21.2.6 EngGetCjState() - コントロールジョブ状態の取得.....	16
EngGetCjStateX() - インデクスでのコントロールジョブ状態の取得.....	16
3.21.2.7 EngGetCjList() - 全登録コントロールジョブID取得関数.....	18
3.21.2.8 EngGetCjId() - インデクスからCJIDの取得.....	19
3.21.2.9 EngGetCjIdIndex() - CJIDからインデクスの取得.....	20
3.21.3 CJ コントロールジョブ関連ライブラリ関数.....	21
3.21.3.1 DshDecodeS14F9() - S14F9 をコントロールジョブ情報にデコード.....	21
3.21.3.2 DshFreeTOBJ_INFO() - コントロールジョブ情報構造体メモリの開放.....	22
3.21.3.3 DshCopyTOBJ_INFO() - コントロールジョブ情報構造体メモリのコピー.....	23
3.21.3.4 DshFreeTOBJ_S14_ERR_INFO() - コントロールジョブ応答情報メモリの開放.....	24
3.21.3.5 DshInitCjInfo - コントロールジョブTOBJ_INFOの初期設定.....	25
3.21.3.6 DshPutCjAttrInfo() - CJ属性情報の追加.....	27
3.21.3.7 DshInitCjTextInfo() - TextInfo情報の追加(EN_CarrierInputSpec, EN_CurrentPRJob)	29
3.21.3.8 DshPutCjTextInfo() - テキスト情報を追加.....	30
3.21.3.9 DshInitCjVoidList() - VoidList情報の追加(EN_CarrierInputSpec, EN_CurrentPRJob)	31
3.21.3.10 DshPutCjVoidList() - VOID情報を追加.....	32
3.21.3.11 DshInitCjMtrlOutByStatus() - MtrlOutByStatus情報の追加.....	33
3.21.3.12 DshPutCjMtrlOutByStatus() - MtrlOutByStatus情報を追加.....	34
3.21.3.13 DshInitCjMtrlOutSpec() - MtrlOutSpec情報の追加.....	35
3.21.3.14 DshPutCjMtrlOutSpec() - MTRLOutSpec情報を追加.....	37
3.21.3.15 DshInitCjProcessCtrlSpec() - ProcessCtrlSpec情報の追加.....	38
3.21.3.16 DshPutCjCtrlRuleInfo() - ControlRule情報を追加.....	39
3.21.3.17 DshPutCjOutRuleInfo() - OutRule情報を追加.....	41
3.21.3.18 DshInitCjPRJobStatusList() - PRJobStatusList情報の追加.....	43
3.21.3.19 DshPutCjPRJobStatus() - PRJob状態情報を追加.....	44
3.21.3.20 DshInitCjPauseEvent() - Pause Event情報の追加.....	45
3.21.3.21 DshPutCjPauseEvent() - Pause Event追加.....	46
3.21.3.22 DshMakeS14F9Response() - S14F9 の応答メッセージの生成.....	47
3.21.3.23 DshDecodeS14F11() - S14F11 をCJ削除情報にデコード.....	49
3.21.3.24 DshMakeS14F11Response() - S14F11 の応答メッセージの生成.....	50
3.21.3.25 DshDecodeS16F27 - S16F27 をCJコマンド情報にデコード.....	52

3 . 21 . 3 . 26	DshFreeTCJ_CMD_INFO() - CJコマンド情報構造体メモリの開放	53
3 . 21 . 3 . 27	DshCopyTCJ_CMD_INFO() - CJコマンド情報構造体メモリのコピー	54
3 . 21 . 3 . 28	DshInitTCJ_CMD_INFO() - CJコマンドTCJ_CMD_INFOの初期設定	55
3 . 21 . 3 . 29	DshAddTCJ_CMD_INFO() - コントロールジョブコマンドパラメータの追加	57
3 . 21 . 3 . 30	DshInitTCJ_CMD_ERR_INFO () - CJコマンド応答情報の初期化	59
3 . 21 . 3 . 31	DshFreeTCJ_CMD_ERR_INFO() - コントロールジョブコマンド応答情報メモリの開放	61
3 . 21 . 3 . 32	DshMakeS16F27Response() - S16F27 の応答メッセージの生成	62
3 . 21 . 4	ユーザ作成ライブラリ関数	64
3 . 21 . 4 . 1	DshResponseS14F10() - S14F10 コントロールジョブ生成要求応答メッセージ	64
3 . 21 . 4 . 2	DshResponseS14F12() - S14F12 コントロールジョブ削除要求応答メッセージ	66
3 . 21 . 4 . 4	DshResponseS16F28() - S16F28 コントロールジョブコマンド応答メッセージ	68

(VOL - 1 4に続く)

3.21 CJ コントロールジョブ情報アクセスサービスとメッセージ送信関数

ここで述べるコントロールジョブ情報は、DSHEng4 が管理します。従って、APP はこれらの情報をアクセスと関連メッセージを送信するために以下の DSHEng4 API 関数を使用します。



(1) 情報アクセスと送信 API 関数

コントロールジョブ情報のアクセスとホストへのメッセージ送信に関連するサービスのための API 関数名は一覧表のとおりです。

	API 関数名	機能
1	EngAllocCjInfo()	コントロールジョブ領域を割当て登録します。
2	EngSetCjInfo()	コントロールジョブ情報を設定・変更します。
3	EngGetCjInfo()	CJID 指定でコントロールジョブ情報を取得します。
4	EngGetCjInfoX()	コントロールジョブインデックス指定でコントロールジョブ情報を取得します。
5	EngDelCjInfo()	CJID 指定でコントロールジョブ情報を削除します。
6	EngDelCjInfoX()	コントロールジョブインデックス指定でコントロールジョブ情報を削除します。
7	EngGetCjId()	指定したコントロールジョブインデックスの CJID を取得します。
8	EngGetCjIdIndex()	指定した CJID の情報インデックスを取得します。
9	EngSetCjIdStatus()	CJID 指定で CJ の状態を設定します。
10	EngSetCjIdStatusX()	コントロールジョブインデックス指定で CJ の状態を設定します。
11	EngGetCjIdStatus()	CJID 指定で CJ の状態を取得します。
12	EngGetCjIdStatusX()	コントロールジョブインデックス指定で CJ の状態を取得します。
13	EngGetCjList()	コントロールジョブ ID の一覧リストを取得します。

CJID インデックスは、DSHEng4 が管理する各 CJID 領域の番号です。このインデックスの値は、EngAllocCjInfo() 関数実行時に DSHEng4 によって割当てられ、APP に渡されます。また、コントロールジョブの取得時に、情報格納構造体のメンバー、index に設定されます。

(2) ライブラリ関数

他に APP が使用できるコントロールジョブ情報処理用 API 関数として、以下の関数があります。

	API 関数名	機能
1	DshDecodeS14F9()	S14F9 のメッセージ内コントロールジョブ情報を TOBJ_INFO 構造体にデコード・収納するための関数です。
2	DshFreeTOBJ_INFO()	コントロールジョブ情報が格納されている TOBJ_INFO 構造体内部で使用されているメモリを開放するための関数です。
3	DshCopyTOBJ_INFO()	TOBJ_INFO のコントロールジョブ情報を別の構造体にコピーします。
4	DshFreeTOBJ_S14_ERR_INFO()	S14F10、S14F12 応答情報内のメモリを開放します。
5	DshInitCjInfo()	TOBJ_INFO 構造体を初期設定します。 (CJ 情報設定、S14F9、S14F11 用)
6	DshPutCjAttrInfo()	CJ 情報の属性情報を TOBJ_INFO 内に 1 個追加設定します。
7	DshInitCjTextInfo()	CJ 属性情報、EN_CarrierInputSpec、EN_CurrentPRJob 用テキスト情報構造体を初期設定します。TCJ_TEXT_INFO 構造体を使用します。
8	DshCjPutTextInfo()	テキスト情報を TCJ_TEXT_INFO 内に 1 個追加設定します。
9	DshInitCjVoidList()	CJ 属性情報、EN_MtrlOutByStatus、EN_MtrlOutSpec、EN_ProcessingCtrlSpec 用 VOID 情報構造体を初期設定します。TVOID_LIST 構造体を使用します。
10	DshCjPutVoidList()	テキスト情報を TTEXT_INFO 内に 1 個追加設定します。
11	DshInitCjMtrlOutByStatus()	MtrlOutByStatus 情報構造体 TMTRL_OUT_STAT を初期設定します。
12	DshPutCjMtrlOutByStatus()	MtrlOutByStatus 情報を TMTRL_OUT_STAT 構造体に 1 個追加設定します。
13	DshInitCjMtrlOutBySpec()	MtrlOutSpec 情報構造体 TMTRL_OUT_SPEC を初期設定します。
14	DshPutCjMtrlOutBySpec()	MtrlOutSpec 情報を TMTRL_OUT_SPEC 構造体に 1 個追加設定します。
15	DshInitCjProcessCtrlSpec()	ProcessingCtrlSpec 情報構造体 TCTRL_SPEC を初期設定します。
16	DshPutCjCtrlRuleInfo()	CtrlRule 情報構造体 TCTRL_RULE 構造体を初期設定します。 (TCTRL_SPEC 構造体内に設定する情報です。)
17	DshPutCjOutRuleInfo()	OutRule 情報構造体 TOUT_RULE 構造体を初期設定します。 (TCTRL_SPEC 構造体内に設定する情報です。)
18	DshInitCjPRJobStatusList()	PRJID 状態情報構造体 TPRJ_STAT を初期設定します。
19	DshPutCjPRJobStatus()	PRJID 状態情報を TPRJ_STAT 構造体に 1 個追加設定します。
20	DshInitCjPauseEvent()	Pause Event 情報構造体 TPAUSE_EVENT を初期設定します。
21	DshPutCjPauseEvent()	Pause Event の CEID を TPAUSE_EVENT 構造体に 1 個追加設定します。

22	DshMakeS14F9Response()	S14F9 メッセージに対する応答メッセージ S14F10 を TOBJ_INFO, TOBJ_ERR_INFO 構造体に設定された情報に基づいて生成します。
23	DshDecodeS14F11()	CJ 削除メッセージ S14F11 を CJ 削除情報構造体にデコードします。
24	DshMakeS14F11Response()	S14F11 メッセージに対する応答メッセージ S14F12 を TOBJ_INFO, TOBJ_ERR_INFO 構造体に設定された情報に基づいて生成します。
25	DshDecodeS16F27()	CJ コマンドメッセージ S16F27 メッセージを TCJ_CMD_INFO 構造体にデコードします。
26	DshFreeTCJ_CMD_INFO()	TCJ_CMD_INFO 構造体内のメモリを開放します。
27	DshCopyTCJ_CMD_INFO()	TCJ_CMD_INFO 構造体をコピーします。
28	DshInitTCJ_CMD_INFO()	S16F27 CJ コマンドメッセージのための TCJ_CMD_INFO 構造体の初期設定をします。
29	DshAddTCJ_CMD_INFO()	TCJ_CMD_INFO 構造体にコマンドパラメータを 1 個追加設定します。
30	DshInitTCJ_CMD_ERR_INFO()	S16F27 に対する S16F28 生成のために必要なコントロールジョブコマンド 応答情報を TCJ_CMD_ERR_INFO 構造体内に生成します。
31	DshFreeTCJ_CMD_ERR_INFO()	TCJ_CMD_ERR_INFO 構造体内で使用されているメモリを開放します。
32	DshMakeS16F27Response()	S16F27 メッセージに対する応答メッセージ S16F28 を TCJ_CMD_INFO, TCJ_CMD_ERR_INFO 構造体に設定された情報に基づいて生成します。

(3) ユーザ作成ライブラリ関数

	ライブラリ関数名	機能
1	DshResponseS14F10()	S14F9 コントロールジョブ生成応答
2	DshResponseS14F12()	S14F3 コントロールジョブ削除表示応答
3	DshResponseS16F27()	S16F27 コントロールジョブコマンド応答

3.21.1 使用する情報格納構造体

コントロールジョブ情報を操作する関数は、共通の情報格納のための TOBJ_INFO 構造体を使用します。コントロールジョブに関連する構造体は下記のとおりです。

(1) TOBJ_INFO Control Job Information

```
typedef struct{
    int          index;          // 管理領域の index
    int          state;         // CJ 状態(ユーザ 定義)
    int          objspec_flag;  // DSEng4 内部処理用フラグ
    char         *objspec;      // オブジェクト ID(=CJID)
    int          objtype_flag;  // オブジェクトタイプコード
    char         *objtype;      // オブジェクトタイプ名
    char         *objid;        // control job id
    int          attr_count;    // attrid の数
    TOBJ_ATTR_INFO **attr_list; // attrid 情報のリスト (S14F9 に出てくる順番)
} TOBJ_INFO;
```

objtype_flag のコードは次の通り・

```
#define EN_ControlJob      0
#define EN_Substrate      1
#define EN_xxx2           2
```

(2) TCJ_ATTR_INFO - Control Job Attribute Information

```
typedef struct{
    char         *attrid;       // attrid
    int          attrid_index;  // attrid の識別 index(下に示す)
    void         *attrdata;     // attrdata 情報領域のポインタ
} TOBJ_ATTR_INFO;
```

attrid_index の値は次のとおり。EN_ の後が、attrid を示しています。

```
#define EN_ObjID          0
#define EN_CarrierInputSpec 1
#define EN_CurrentPRJob  2
#define EN_DataCollectionPlan 3
#define EN_MtrIOutByStatus 4
#define EN_MtrIOutSpec   5
#define EN_PauseEvent    6
#define EN_ProcessingCtrlSpec 7
#define EN_ProcessingOrderMgmt 8
#define EN_PRJobStatusList 9
#define EN_StartMethod   10
#define EN_State         11
```


(3) TOBJ_ERR_INFO - S14F10 Response Information

```
typedef struct{
    int      objack;
    int      err_count;
    TERR_INFO **err_list;
} TOBJ_ERR_INFO;
```

(4) TMTRL_OUT_STAT - "MtrlOutByStatus" の Attribute Information

```
typedef struct{
    int      mtrl_status;      // U1
    char     *carid;
    int      slot_count;
    int      *slotid_list;
} TMTRL_OUT_STAT;
```

(4) TMTRL_OUT_SPEC - "MtrlOutSpec" の Attribute Information

```
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;
```

(5) TVOID_LIST - attrid で attribute の中で単純なデータ配列 Information 格納用

```
typedef struct{
    int      count;
    void     **void_list;
} TVOID_LIST;
```

(6) TCTRL_SPEC - "ProcessingCtrlSpec"

```
typedef struct{
    char     *prjobid;
    int      ctrl_rule_count;
    TCTRL_RULE **ctrl_rule_list;
    int      out_rule_count;
    TOUT_RULE **out_rule_list;
} TCTRL_SPEC;
```

(7) TCTRL_RULE - TCTRL_SPEC のメンバー - ctrl_rule_list

```
typedef struct{
    char    *name;
    int     fmt;
    int     asize;
    void    *value;
} TCTRL_RULE;
```

(8) TOUT_RULE - TCTRL_SPEC のメンバー out_rule_list

```
typedef struct{
    int     status;           // u1
    int     fmt;
    int     asize;
    void    *value;
} TOUT_RULE;
```

(9) TPAUSE_EVENT - "PauseEvent" のパラメータ CEID イベントリスト

```
typedef struct{
    int     ce_count;
    int     *ceid_list;
} TPAUSE_EVENT;
```

(10) TCJ_TEXT_INFO - "CarrierInputSpec" のキャリア ID リスト、 "CurrentPrJob" のプロセッシング ID リスト用

```
typedef struct{
    int     text_count;
    char    **text_list;
} TCJ_TEXT_INFO;
```

(11) TCJ_CMD_INFO - Control Job Command Information - S16F5

```
typedef struct{
    int     prj_count;
    char    **prj_list;           // prjid list
    int     *state_list;        // state list (U1)
} TPRJ_STATE_LIST;
```

(12) TCJ_CMD_INFO - S16F27 Control Job Command Information

```
typedef struct{
    char      *ctljobid;      // コントロールジョブ ID
    char      *cmd;          // コマンド
    int       cmd_index;
    TCMD_PARA *cp_info;      // コマンドパラメータリスト
} TCJ_CMD_INFO;
```

cmd_index の値は以下のとおり。CM_ 以降の文字列がコマンド名に対応しています。

```
#define CM_Start      0
#define CM_Pause     1
#define CM_Resume    2
#define CM_Cancel    3
#define CM_Deselect  4
#define CM_Stop      5
#define CM_Abort     6
#define CM_CJHOQ     7
```

(13) TCMD_PARA - TCJ_CMD_INFO のメンバー - cp_info

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

(14) S16F28 応答情報格納用

```
typedef struct{
    int       objack;
    TERR_INFO *err_info;
} TCJ_CMD_ERR_INFO;
```

(15) 装置外関連通信メッセージの応答情報用

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

3.21.2 CJ コントロールジョブ情報アクセス、送信関数

3.21.2.1 EngAllocCjInfo() - コントロールジョブの登録

(1) 呼出書式

[C, C++]

```
API int APIX EngAllocCjInfo(
    char *cjid,           // コントロールジョブ ID 格納領域のポインタ
    int *index           // 得られた情報領域インデックス格納用ポインタ
);
```

[.NET VB]

```
Function EngAllocCjInfo (
    ByVal cjid As String,
    ByRef index As Int32) As Int32
```

[.NET C#]

```
int EngAllocCjInfo(
    byte[] cjid,
    ref int index );
```

(2) 引数

cjid

登録したいコントロールジョブ ID が格納されているポインタです。

index

登録された PRJ 情報領域のインデックス値が格納される領域のポインタです。

(3) 戻り値

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

(4) 説明

コントロールジョブを新規にシステムに登録するための関数です。

登録は、引数 cjid で与えられるコントロールジョブ ID をシステムに登録します。

正常に登録できた場合は、index で指定される領域に登録された情報領域のインデックスが設定返却されます。もし、cjid に指定されたコントロールジョブが既に登録済みであった場合には関数の戻り値 =1 を返却します。index には既に登録されている情報領域のインデックスが設定されます。

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

3.21.2.2 EngSetCjInfo() - コントロールジョブ情報の設定 EngSetCjInfoX() インデクスでのコントロールジョブ情報の設定

(1) 呼出書式

[C, C++]

```
API int APIX EngSetCjInfo(
    TOBJ_INFO *pinfo // コントロールジョブ情報格納構造体のポインタ
);

API int APIX EngSetCjInfoX(
    int index, // EngAllocCjInfo()で得られたコントロールジョブ情報のインデクス
    TOBJ_INFO *pinfo // コントロールジョブ情報格納構造体のポインタ
);
```

[.NET VB]

```
Function EngSetCjInfo (
    ByRef pinfo As dsh_info.TOBJ_INFO) As Int32

Function EngSetCjInfoX (
    ByVal index As Int32,
    ByRef pinfo As dsh_info.TOBJ_INFO) As Int32
```

[.NET C#]

```
int EngSetCjInfo(
    ref TOBJ_INFO pinfo );

int EngSetCjInfoX(
    int index,
    ref TOBJ_INFO pinfo );
```

(2) 引数

pinfo

設定したいコントロールジョブ情報が格納されている格納構造体領域のポインタです。

index

コントロールジョブ ID 情報のインデクスです。登録時に EngAllocCjInfo()関数によって与えられます。

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

(3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	設定できなかった。

(4) 説明

本関数は、pinfo に格納されているコントロールジョブ情報の設定・変更に使用します。

引数 pinfo 内の cjid メンバーに指定されるコントロールジョブ ID の情報として設定されます。

pinfo 内にはCJID の他、キャリア、レシピ情報などの情報が含まれます。
指定した CJID が既に登録済みであった場合にはその情報は pinfo 内の情報にすべて書き換えられます。

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

3.21.2.3 EngGetCjInfo() - コントロールジョブ情報の取得 EngGetCjInfoX() インデクス指定でのコントロールジョブ情報の取得

(1) 呼出書式

[C, C++]

```
API int APIX EngGetCjInfo(
    char *cjid, // CJID が格納されている領域のポインタ
    TOBJ_INFO *pinfo // コントロールジョブ情報を格納するための構造体ポインタ
);
```

```
API int APIX EngGetCjInfoX(
    int index, // コントロールジョブ情報のインデクス
    TOBJ_INFO *pinfo // コントロールジョブ情報を格納するための構造体ポインタ
);
```

[.NET VB]

```
Function EngGetCjInfo (
    ByVal cjid As String,
    ByRef pinfo As dsh_info.TOBJ_INFO) As Int32
```

```
Function EngGetCjInfoX (
    ByVal index As Int32,
    ByRef pinfo As dsh_info.TOBJ_INFO) As Int32
```

[.NET C#]

```
int EngGetCjInfo(
    byte[] cjid,
    ref TOBJ_INFO pinfo );
```

```
int EngGetCjInfoX(
    int index,
    ref TOBJ_INFO pinfo );
```

(2) 引数

cjid

コントロールジョブ ID が格納されている領域のポインタです。

pinfo

取得したコントロールジョブ情報を格納するための構造体領域のポインタです。

index

コントロールジョブ ID 情報のインデクスです。登録時に EngAllocCjInfo()関数によって与えられます。

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

(3) 戻り値

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

(4) 説明

cjid または index に指定されているコントロールジョブの情報を pinfo 構造体取得格納します。

TOBJ_INFO 構造体の中に情報を格納するために必要なメモリは、DSHEng4 が準備確保します。即ち、構造体のメンバーの中でポインタになっている情報の実体即ち、コントロールジョブ、キャリア情報などのためのメモリは DSHEng4 が準備します。

これらのメモリは、使用后、ユーザが DSHEng4 の API 関数を使って次のように開放してください。

```
TOBJ_INFO *pinfo;

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


3.21.2.4 EngDelCjInfo() - コントロールジョブの削除 EngDelCjInfoX() インデクスでのコントロールジョブの削除

(1) 呼出書式

[C, C++]

```
API int APIX EngDelCjInfo(
    char *cjid // CJID が格納されている領域のポインタ
);
```

```
API int APIX EngDelCjInfo(
    int index // インデクス(0,1,2,...)
);
```

[.NET VB]

```
Function EngDelCjInfo (
    ByVal cjid As String) As Int32
```

```
Function EngDelCjInfoX (
    ByVal index As Int32) As Int32
```

[.NET C#]

```
int EngDelCjInfo(
    byte[] cjid );
```

```
int EngDelCjInfoX(
    int index );
```

(2) 引数

cjid

コントロールジョブ ID が格納されている領域のポインタです。

index

コントロールジョブ ID 情報のインデクスです。登録時に EngAllocCjInfo()関数によって与えられます。

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

(3) 戻り値

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

(4) 説明

cjid または index に指定されているコントロールジョブ ID の情報をシステムの登録から削除します。

3.21.2.5 EngSetCjState() コントロールジョブ状態の設定 EngSetCjStateX() インデクスでのコントロールジョブ状態の設定

(1) 呼出書式

[C, C++]

```
API int APIX EngSetCjState(
    char    *cjid,           // CJID が格納されている領域のポインタ
    int     state           // 設定したい状態値
);
```

```
API int APIX EngSetCjStateX(
    int     index,          // コントロールジョブ情報のインデクス
    int     state           // 設定したい状態値
);
```

[.NET VB]

```
Function EngSetCjState (
    ByVal cjid As String,
    ByVal state As Int32) As Int32
```

```
Function EngSetCjStateX (
    ByVal index As Int32,
    ByVal state As Int32) As Int32
```

[.NET C#]

```
int EngSetCjState(
    byte[] cjid,
    int state );
```

```
int EngSetCjStateX(
    int index,
    int state );
```

(2) 引数

cjid

コントロールジョブ ID が格納されている領域のポインタです。

state

設定したいCJの状態値です。

index

コントロールジョブ ID 情報のインデクスです。登録時に EngAllocCjInfo()関数によって与えられます。

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

(3) 戻り値

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

(4) 説明

コントロールジョブ情報の状態値を設定します。
デフォルトの状態値は下表のとおりです。

コントロールジョブ状態のデフォルト値

状態値記号	値
ST_CJ_Queued	0
ST_CJ_Selected	1
ST_CJ_WaitingForHost	2
ST_CJ_Executing	3
ST_CJ_Paused	4
ST_CJ_Completed	5

3.21.2.6 EngGetCjState() コントロールジョブ状態の取得 EngGetCjStateX() インデクスでのコントロールジョブ状態の取得

(1) 呼出書式

[C, C++]

```
API int APIX EngGetCjState(
    char    *cjid,           // CJID が格納されている領域のポインタ
    int     *state          // 取得した状態値格納用領域ポインタ
);
```

```
API int APIX EngGetCjStateX(
    int     index,          // コントロールジョブ のインデクス
    int     *state          // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function EngGetCjState (
    ByVal cjid As String,
    ByRef state As Int32) As Int32
```

```
Function EngGetCjStateX (
    ByVal index As Int32,
    ByRef state As Int32) As Int32
```

[.NET C#]

```
int EngGetCjState(
    byte[] cjid,
    ref int state );
```

```
int EngGetCjStateX(
    int index,
    ref int state );
```

(2) 引数

cjid

コントロールジョブ ID が格納されている領域のポインタです。

state

取得したコントロールジョブの状態値を格納する領域のポインタです。

index

コントロールジョブ ID 情報のインデクスです。登録時に EngAllocCjInfo()関数によって与えられます。

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

(3) 戻り値

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

(4) 説明

コントロールジョブ情報の状態値を取得します。
デフォルトの状態値は下表のとおりです。

コントロールジョブ状態のデフォルト値

状態値記号	値
ST_CJ_Queued	0
ST_CJ_Selected	1
ST_CJ_WaitingForHost	2
ST_CJ_Executing	3
ST_CJ_Paused	4
ST_CJ_Completed	5

3.21.2.7 EngGetCjList() 全登録コントロールジョブ ID 取得関数

(1) 呼出書式

[C, C++]

```
API int APIX EngGetCjList(
    TTEXT_DLIST **list           // 取得リスト格納ポインタの格納ポインタ
);
```

[.NET VB]

```
Function EngGetCjList (
    ByRef list As IntPtr) As Int32
```

[.NET C#]

```
int EngGetCjList(
    IntPtr list );
```

(2) 引数

list

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

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

(4) 説明

システムに登録されている全 CJID(コントロールジョブ ID)を TTEXT_DLIST 構造体に取り出すための関数です。info->name_list には NULL が設定されます。(定義名がありません。)

取得した情報の処理が終了した後、DshFreeTText_DLIST()関数で list 内部の情報格納用に使用されているメモリを開放してください。

TTEXT_DLIST 構造体は次のとおりです。

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

3.21.2.8 EngGetCjId() インデクスから CJID の取得

(1) 呼出書式

[C, C++]

```
API int APIX EgnGetCjId(
    int index,           // コントロールジョブ情報のインデクス
    char *cjid           // 取得した CJID を格納する領域のポインタ
);
```

[.NET VB]

```
Function EngGetCjId (
    ByVal index As Int32,
    ByVal cjid As String) As Int32
```

[.NET C#]

```
int EngGetCjId(
    int index,
    byte[] cjid );
```

(2) 引数

index

コントロールジョブ情報のインデクスです。登録時に EngAllocJInfo() 関数によって与えられます。インデクスは、CJID から EngGetCjIdIndex() 関数で取得することができます。

cjid

CJID を格納するための領域のポインタです。

(3) 戻り値

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

(4) 説明

コントロールジョブ情報のインデクスから CJID を取得し、cjid に格納します。正常に取得できた場合は関数戻り値として 0 が返却されます。

3.21.2.9 EngGetCjldIndex() CJID からインデクスの取得

(1) 呼出書式

[C, C++]

```
API int APIX EgnGetCjldIndex(
    char *cjid,           // CJID が格納されている領域のポインタ
    int *index           // 取得したインデクスを格納するための領域のポインタ
);
```

[.NET VB]

```
Function EngGetCjldIndex (
    ByVal cjid As String,
    ByRef index As Int32) As Int32
```

[.NET C#]

```
int EngGetCjldIndex(
    byte[] cjid,
    ref int index );
```

(2) 引数

cjid

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

index

取得したインデクスの値を格納するための領域のポインタです。

(3) 戻り値

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

(4) 説明

cjid に指定される CJID からコントロールジョブ情報インデクスを取得するための関数です。
 取得されたインデクスは index で指定された領域に格納されます。
 正常に取得できた場合は関数戻り値として 0 が返却されます。

3.21.3 CJ コントロールジョブ関連ライブラリ関数

3.21.3.1 DshDecodeS14F9() - S14F9 をコントロールジョブ情報にデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS14F9(
    DSHMSG *smsg, // SECS メッセージ 情報構造体のポインタ
    TOBJ_INFO *pinfo // デコードした CJ 情報を格納する構造体のポインタ
);
```

[.NET VB]

```
Function DshDecodeS14F9 (
    ByRef smsg As dshdr2.DSHMSG,
    ByRef info As dsh_info.TOBJ_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS14F9(
    ref DSHMSG smsg,
    ref TOBJ_INFO info );
```

(2) 引数

smsg

S14F9 の SECS メッセージ 情報が格納されている構造体のポインタです。

pinfo

デコードしたコントロールジョブ情報を格納するための構造体ポインタです。

(3) 戻り値

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

(4) 説明

S14F9 メッセージに含まれるコントロールジョブ情報を、ユーザプログラムが処理しやすい TOBJ_INFO 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTOBJ_INFO() 関数を使って開放してください。

smsg S14F9

L,3

objspec

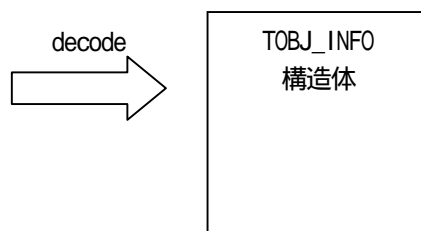
objtype

L,a

L,2

attrid1

.



3.21.3.2 DshFreeTOBJ_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 DshFreeTOBJ_INFO(
    TOBJ_INFO *pinfo           // メリを開放したい情報が格納されている構造体のポインタ
);
```

[.NET VB]

```
Sub DshFreeTOBJ_INFO (
    ByRef info As dsh_info.TOBJ_INFO)
```

[.NET C#]

```
void DshFreeTOBJ_INFO(
    ref TOBJ_INFO info );
```

(2) 引数

pinfo

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

TOBJ_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_INFO の内容を全て 0 で初期設定します。

pinfo が NULL ならば、何も処理しません。

3.21.3.3 DshCopyTOBJ_INFO() - コントロールジョブ情報構造体メモリのコピー

(1) 呼出書式

[C, C++]

```
API int APIX DshCopyTOBJ_INFO(
    TOBJ_INFO *dinfo,           // 北°-先のポインタ
    TOBJ_INFO *sinfo           // 北°-元のポインタ
);
```

[.NET VB]

```
Function DshCopyTOBJ_INFO (
    ByRef info As dsh_info.TOBJ_INFO,
    ByRef sinfo As dsh_info.TOBJ_INFO) As Int32
```

[.NET C#]

```
int DshCopyTOBJ_INFO(
    ref TOBJ_INFO info,
    ref TOBJ_INFO sinfo );
```

(2) 引数

dinfo

コントロールジョブ情報のコピー先ポインタです。

sinfo

コピー元のコントロールジョブ情報が格納されている構造体メモリのポインタです。

(3) 戻り値

戻り値	意味
0	正常に北°-できた。
(-1)	sinfo または dinfo の値が NULL だったので北°-できなかった。

(4) 説明

sinfo が指す TOBJ_INFO 構造体内に格納されているコントロールジョブ情報を dinfo が指定する TOBJ_INFO 構造体にコピーします。

dinfo 内のメンバーで新しいメモリが必要なものは本関数が取得します。

dinfo 内メンバーで確保されたメモリは、使用后、DshFreeTOBJ_INFO()関数を使って開放してください。

3.21.3.4 DshFreeTOBJ_S14_ERR_INFO() - コントロールジョブ応答情報メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTOBJ_S14_ERR_INFO(
    TOBJ_S14_ERR_INFO *info // メモリを開放したい応答情報が格納されている構造体のポインタ
);
```

[.NET VB]

```
Sub DshFreeTOBJ_S14_ERR_INFO (
    ByRef info As dsh_info.TOBJ_S14_ERR_INFO)
```

[.NET C#]

```
void DshFreeTOBJ_S14_ERR_INFO(
    ref TOBJ_S14_ERR_INFO info );
```

(2) 引数

info

メモリを解放したいコントロールジョブ応答情報構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TOBJ_S14_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。

TOBJ_S14_ERR_INFO 構造体は S14F10, S14F12 応答メッセージのデコード結果を格納するために使用されま
す。

3.21.3.5 DshInitCjInfo コントロールジョブ TOBJ_INFO の初期設定

(1) 呼出書式

[C, C++]

```
API void APIX DshInitCjInfo(
    TOBJ_INFO *info,           // TOBJ_INFO 構造体のポインタ
    char *objid,              // コントロールジョブ ID
    char *objspec,           // object spec
    char *objtype,           // object type
    int attr_count           // 属性数
);
```

[.NET VB]

```
Sub DshInitCjInfo (
    ByRef info As dsh_info.TOBJ_INFO,
    ByVal objid As String,
    ByVal objspec As String,
    ByVal objtype As String,
    ByVal attr_count As Int32)
```

[.NET C#]

```
void DshInitCjInfo(
    ref TOBJ_INFO info,
    byte[] objid,
    byte[] objspec,
    byte[] objtype,
    int attr_count );
```

(2) 引数

info

コントロールジョブ TOBJ_INFO 構造体のポインタです。このメンバーを初期設定します。

objid

設定するコントロールジョブ ID です。

objspec

Object Spec 文字列です。NULL でも構いません。

objtype

Object Type 文字列です。NULL の場合は "ControlJob" を設定します。

attr_count

TOBJ_INFO 構造体に含まれる属性情報数です。

本関数が自動的に設定する "Obj ID" 属性を含めた属性の数です。1 以上の値を設定してください。

(3) 戻り値

なし。

(4) 説明

本関数は APP が S14F9 を送信する際に、そして、装置側では、OFFLINE でコントロールジョブ情報を生成する際に使用することができます。

最初に info 内をクリアします。そして、引数で指定された情報を info 内に設定します。
メモリが必要なメンバーについてはメモリを確保し情報をコピーします。

属性情報のうち、“Obj ID”属性情報については本関数が自動的に先頭に設定します。

属性情報の追加設定には DshPutCjAttrInfo()関数を使用してください。

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

3.21.3.6 DshPutCjAttrInfo() CJ 属性情報の追加

(1) 呼出書式

[c,C++]

```
API int APIX DshPutCjAttrInfo(
    TOBJ_INFO *info,           // CJ 情報構造体のポインタ
    int attr_index,          // 属性 ID の index 値
    void *attr_info          // 属性情報格納領域のポインタ
);
```

[.NET VB]

```
Function DshPutCjAttrInfo (
    ByRef info As dsh_info.TOBJ_INFO,
    ByVal attr_index As Int32,
    ByVal attr_info As Int32) As Int32
```

[.NET C#]

```
int DshPutCjAttrInfo(
    ref TOBJ_INFO info,
    int attr_index,
    byte[] attr_info );
```

(2) 引数

info

コントロールジョブ情報構造体のポインタです。

attr_index

加えたい属性 ID のインデクス値です。

オブジェクト属性インデクス値は以下のとおりです。

対応する属性 ID はマクロの EN_XXXX 文字列の EN_ 部分を取り除いた文字列になります。

```
#define EN_ObjID           0
#define EN_CarrierInputSpec 1
#define EN_CurrentPRJob   2
#define EN_DataCollectionPlan 3
#define EN_MtrIOutByStatus 4
#define EN_MtrIOutSpec    5
#define EN_PauseEvent     6
#define EN_ProcessingCtrlSpec 7
#define EN_ProcessingOrderMgmt 8
#define EN_PRJobStatusList 9
#define EN_StartMethod    10
#define EN_State          11
```

(3) 戻り値

戻り値	意味
0	正常に追加できた。
(-1)	属性情報の数が指定数を超過している。

(4) 説明

先に DshInitCjInfo() で初期設定された info 構造体の中の attr_list に属性情報 TOBJ_ATTR_INFO を 1 個加えます。

本関数が実行される順に属性情報リスト attr_list 上に、情報を追加していきます。

設定後 0 を返却します。

もし、info 内の attr_count で指定された分の情報が既に設定済みであった場合は、(-1)を返却します。

属性 EN_DataCollectionPlan, EN_StartMethod と EN_State については引数 attr_info に設定したい数値を指定して実行してください。(例 DshPutCjAttrInfo(info, EN_StartMethod, (void*)1);)

3.21.3.7 DshInitCjTextInfo() TextInfo 情報の追加(EN_CarrierInputSpec, EN_CurrentPRJob)

(1) 呼出書式

[C, C++]

```
API int APIX DshInitCjTextInfo(
    TCJ_TEXT_INFO *info,           // Text 情報構造体のポインタ
    int text_count                 // Text 情報内 text_list の設定数
);
```

[.NET VB]

```
Sub DshInitCjTextInfo (
    ByRef info As IntPtr,
    ByVal text_count As Int32)
```

[.NET C#]

```
void DshInitCjTextInfo(
    ref TCJ_TEXT_INFO info,
    int text_count );
```

(2) 引数

info

新規に取得した TCJ_TEXT_INFO 情報構造体メモリのポインタを格納するポインタです。

text_count

TCJ_TEXT_INFO 構造体内に設定したいテキスト情報の数です。

(3) 戻り値

なし。

(4) 説明

属性インデクス EN_CarrierInputSpec, EN_CurrentPRJob に適用されます。

TCJ_TEXT_INFO 構造体のためのメモリは本関数が取得します。そして、与えられた引数を設定します。また text_count 分のテキスト情報設定のためのリスト text_list を準備します。

text_list へのテキスト情報の設定は DshPutCjTextInfo() 関数を使って行います。

3.21.3.8 DshPutCjTextInfo () テキスト情報を追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutCjTextInfo(
    TCJ_TEXT_INFO *info,          // CJテキスト情報構造体のポインタ
    char          *text          // テキスト(文字列)のポインタ
);
```

[.NET VB]

```
Function DshPutCjTextInfo (
    ByRef info As dsh_info.TCJ_TEXT_INFO,
    ByVal text As String) As Int32
```

[.NET C#]

```
int DshPutCjTextInfo(
    ref TCJ_TEXT_INFO info,
    byte[] text );
```

(2) 引数

info

CJテキスト情報構造体のポインタです。

text

設定したいテキスト(文字列)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)	TEXT情報の数が指定数を超えている。

(4) 説明

先に DshInitCjTextInfo() で初期設定された info 構造体の中の text_list に text の情報を 1 個加えます。本関数が実行される順に情報リスト text_list 上に、テキスト情報を追加していきます。

設定後 0 を返却します。

もし、info 内の text_count で指定された分の情報が既に設定済みであった場合は、(-1) を返却します。

3.21.3.9 DshInitCjVoidList() VoidList 情報の追加(EN_CarrierInputSpec, EN_CurrentPRJob)

(1) 呼出書式

[C, C++]

```
API int APIX DshInitCjVoidList(
    TVOID_LIST *info,           // Void 情報構造体のポインタ
    int void_count             // Void 情報内 void_list の設定数
);
```

[.NET VB]

```
Sub DshInitCjVoidList (
    ByVal info As IntPtr,
    ByVal long_count As Int32)
```

[.NET C#]

```
void DshInitCjVoidList(
    ref TVOID_LIST info,
    int void_count );
```

(2) 引数

info

新規に取得した TVOID_LIST 情報構造体メモリのポインタです。

void_count

TVOID_LIST 構造体内に設定したい VOID 情報の数です。

(3) 戻り値

なし。

(4) 説明

属性インデクス EN_MtrIOutByStatus, EN_MtrIOutSpec, EN_ProcessingCtrlSpec に適用されます。

TVOID_LIST 構造体のためのメモリは本関数が取得します。そして、与えられた引数を設定します。また void_count 分の VOID 情報設定のためのリスト void_list を準備します。

void_list への VOID 情報の設定は DshPutCjVoidList() 関数を使って行います。

3.21.3.10 DshPutCjVoidList () VOID 情報を追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutCjVoidList(
    TVOID_LIST *info,           // CJVOID 情報構造体のポインタ
    void *void_info           // VOID 情報のポインタ
);
```

[.NET VB]

```
Function DshPutCjVoidList (
    ByRef info As dsh_info.TVOID_LIST,
    ByVal long_info As Int32) As Int32
```

[.NET C#]

```
int DshPutCjVoidList(
    ref TVOID_LIST info,
    byte[] void_info );
```

(2) 引数

info

CJVOID 情報構造体のポインタです。

void_info

設定したい VOID 情報のポインタです。

(3) 戻り値

戻り値	意味
0	正常に追加できた。
(-1)	VOID 情報の数が指定数を超えている。

(4) 説明

先に DshInitCjVoidList() で初期設定された info 構造体の中の void_list に VOID 情報を 1 個加えます。本関数が実行される順に情報リスト void_list 上に、VOID 情報を追加していきます。

設定後 0 を返却します。

もし、info 内の void_count で指定された分の情報が既に設定済みであった場合は、(-1) を返却します。

3.21.3.11 DshInitCjMtrlOutByStatus() MtrlOutByStatus 情報の追加

(1) 呼出書式

[C, C++]

```
API int APIX DshInitCjMtrlOutByStatus(
    TMTRL_OUT_STAT *info,           // MtrlOut 情報構造体のポインタ
    int mtrl_status,               // Mtrl(材料)状態
    char *carid,                   // キャリア ID
    int slot_count                  // キャリア内スロット数
);
```

[.NET VB]

```
Sub DshInitCjMtrlOutByStatus (
    ByVal info As IntPtr,
    ByVal mtrl_status As Int32,
    ByVal carid As String,
    ByVal slot_count As Int32)
```

[.NET C#]

```
void DshInitCjMtrlOutByStatus(
    ref TMTRL_OUT_STAT info,
    int mtrl_status,
    byte[] carid,
    int slot_count );
```

(2) 引数

info
新規に取得した TMTRL_OUT_STAT 情報構造体メモリのポインタポインタです。

mtrl_status
Mtrl の状態です。

carid
キャリア ID です。

slot_count
キャリアに含むスロット数です。

(3) 戻り値

なし。

(4) 説明

属性インデクス EN_MtrlOutByStatus に適用されます。

TMTRL_OUT_STAT 構造体のためのメモリは本関数が取得します。そして、与えられた引数を設定します。また、slot_count 分のスロット ID 設定のためのリスト slot_list を準備します。

slot_list へのスロット ID の設定は DshPutCjMtrlStatusSlotId () 関数を使って行います。

(注) EN_MtrlOutByStatus 情報は TVOID_LIST 構造体内に 1 個以上の TMTRL_OUT_STAT 情報が保存されます。

3.21.3.12 DshPutCjMtrIOutByStatus () MtrIOutByStatus 情報を追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutCjMtrIOutByStatus(
    TMTRL_OUT_STAT *info,           // MtrIOutByStatus 情報構造体のポインタ
    int slotid                       // 設定したいスロット ID
);
```

[.NET VB]

```
Function DshPutCjMtrIOutByStatus (
    ByRef info As dsh_info.TMTRL_OUT_STAT,
    ByVal slotid As Int32) As Int32
```

[.NET C#]

```
int DshPutCjMtrIOutByStatus(
    ref TMTRL_OUT_STAT info,
    int slotid );
```

(2) 引数

info

TMTRL_OUT_STAT 構造体のポインタです。

slot

設定したいスロット ID です。

(3) 戻り値

戻り値	意味
0	正常に追加できた。
(-1)	スロット ID の数が指定数を超えている。

(4) 説明

先に DshInitCjMtrIOutByStatus() で初期設定された info 構造体の中の slot_list に slot の情報を 1 個加えます。

本関数が実行される順に情報リスト slot_list 上に、スロット ID を追加していきます。

設定後 0 を返却します。

もし、info 内の slot_count で指定された分の情報が既に設定済みであった場合は、(-1) を返却します。

3.21.3.13 DshInitCjMtrIOutSpec() MtrIOutSpec 情報の追加

(1) 呼出書式

[C, C++]

```
API int APIX DshInitCjMtrIOutSpec(
    TMTL_OUT_SPEC *info,           // MtrIOut 情報構造体のポインタ
    char *src_carid,              // 元のキャリア ID
    int src_slot_count,           // 元のキャリアの-slot数
    char *dst_carid,              // 先のキャリア ID
    int dst_slot_count,           // 先のキャリアの-slot数
);
```

[.NET VB]

```
Sub DshInitCjMtrIOutSpec (
    ByVal info As IntPtr,
    ByVal src_carid As String,
    ByVal src_slot_count As Int32,
    ByVal dst_carid As String,
    ByVal dst_slot_count As Int32)
```

[.NET C#]

```
void DshInitCjMtrIOutSpec(
    ref TMTL_OUT_SPEC info,
    byte[] src_carid,
    int src_slot_count,
    byte[] dst_carid,
    int dst_slot_count );
```

(2) 引数

info

新規に取得した TMTL_OUT_SPEC 情報構造体メモリのポインタです。

src_carid

元のキャリア ID です。

src_slot_count

元のキャリアに含むスロット数です。

dst_carid

先のキャリア ID です。

dst_slot_count

先のキャリアに含むスロット数です。

(3) 戻り値

なし。

(4) 説明

属性インデクス EN_MtrIOutSpec に適用されます。

TMTL_OUT_SPEC 構造体のためのメモリは本関数が取得します。そして、与えられた引数を設定します。また src_slot_count、dst_slot_count 分のスロット ID 設定のためのリストをそれぞれ src_slot_list

dst_slot_list を準備します。

スロット ID の設定は DshPutCjMtrlSpecSlotId () 関数を使って行います。

(注)

EN_MtrlOutSpec 情報は TVOID_LIST 構造体内に 1 個以上の TMTRL_OUT_SPEC 情報が保存されます。

3.21.3.14 DshPutCjMtrIOutSpec() MTRLOutSpec 情報を追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutCjMtrIOutSpec(
    TMTRL_OUT_SPEC *info,           // MtrIOutSpec 情報構造体のポインタ
    int src_slotid,                 // 設定したい元のスロット ID
    int dst_slotid                  // 設定したい先のスロット ID
);
```

[.NET VB]

```
Function DshPutCjMtrIOutSpec (
    ByRef info As dsh_info.TMTRL_OUT_SPEC,
    ByVal src_slotid As Int32,
    ByVal dst_slotid As Int32) As Int32
```

[.NET C#]

```
int DshPutCjMtrIOutSpec(
    ref TMTRL_OUT_SPEC info,
    int src_slotid,
    int dst_slotid );
```

(2) 引数

info

TMTRL_OUT_SPEC 構造体のポインタです。

src_slot

設定したい元のスロット ID です。(値が -1 ならば設定しません。)

dst_slot

設定したい先のスロット ID です。(値が -1 ならば設定しません。)

(3) 戻り値

戻り値	意味
0	正常に追加できた。
(-1)	スロット ID の数が指定数を超えている。

(4) 説明

先に DshInitCjMtrIOutSpec() で初期設定された info 構造体の中の src_slot_list と dst_slot_list にそれぞれ slot ID を 1 個加えます。

本関数が実行される順に情報リスト src_slot_list と dst_slot_list 上に、スロット ID を追加していきます。ただし、スロット ID の値が (-1) の場合はそのスロット ID は設定しません。

設定後 0 を返却します。

もし、info 内の src_slot_count または dst_slot_count で指定された分の情報が既に設定済みであった場合は、(-1) を返却します。

3.21.3.15 DshInitCjProcessCtrlSpec() ProcessCtrlSpec 情報の追加

(1) 呼出書式

[C, C++]

```
API int APIX DshInitCjProcessCtrlSpec(
    TCTRL_SPEC *info,           // ProcessCtrlSpec 情報構造体のポインタ
    char *prjobid,             // プロセスジョブ ID
    int ctrl_rule_count,       // Control ルール情報の数
    int out_rule_count,        // Out ルール情報の数
);
```

[.NET VB]

```
Sub DshInitCjProcessCtrlSpec (
    ByVal info As IntPtr,
    ByVal prjobid As String,
    ByVal ctrl_rule_count As Int32,
    ByVal out_rule_count As Int32)
```

[.NET C#]

```
void DshInitCjProcessCtrlSpec(
    ref TCTRL_SPEC info,
    byte[] prjobid,
    int ctrl_rule_count,
    int out_rule_count );
```

(2) 引数

info
新規に取得した TCTRL_SPEC 情報構造体メモリのポインタです。

prjobid
プロセスジョブ ID 文字列のポインタです。

ctrl_rule_count
Control ルール情報の数です。

out_rule_count
Out ルール情報の数です。

(3) 戻り値

なし。

(4) 説明

属性インデクス EN_ProcessCtrlSpec に適用されます。

TCTRL_SPEC 構造体のためのメモリは本関数が取得します。そして、与えられた引数を設定します。

また ctrl_rule_count、out_rule_count 分のルール情報設定のためのリストをそれぞれ ctrl_rule_list、out_rule_list に準備します。

制御ルール情報の追加設定は DshPutCjCtrlRuleInfo()、搬出ルール情報の追加は DshPutCjOutRuleInfo() 関数を使って設定することができます。

(注) EN_ProcessingCtrlSpec 情報は TV0ID_LIST 構造体内に 1 個以上の TCTRL_SPEC 情報が保存されます。

3.21.3.16 DshPutCjCtrlRuleInfo() ControlRule 情報を追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutCjCtrlRuleInfo(
    TCTRL_SPEC *info,           // ProcessCtrlSpec 情報構造体のポインタ
    char *name,                 // ルール名格納ポインタ
    int fmt,                    // ルール値のフォーマット
    int asize,                  // ルール値の配列サイズ
    void *value                 // ルール値の格納ポインタ
);
```

[.NET VB]

```
Function DshPutCjCtrlRuleInfo (
    ByRef info As dsh_info.TCTRL_SPEC,
    ByVal name As String,
    ByVal fmt As Int32,
    ByVal asize As Int32,
    ByVal value As Int32) As Int32
```

[.NET C#]

```
int DshPutCjCtrlRuleInfo(
    ref TCTRL_SPEC info,
    byte[] name,
    int fmt,
    int asize,
    byte[] value );
```

(2) 引数

info
TCTRL_SPEC 構造体のポインタです。

name
ルール名が格納されている領域のポインタです。

fmt
ルール値 value のフォーマットです。

asize
ルール値 value の配列サイズです。

value
ルール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)	Control Rule 情報の数が指定数を超えている。

(4) 説明

先に DshInitCjProcessCtrlSpec() で初期設定された info 構造体の中の ctrl_rule_list に Control Rule 情報を 1 個加えます。

本関数が実行される順に情報リスト ctrl_rule_list 上に、Control Rule 情報を追加していきます。設定後 0 を返却します。

もし、info 内の ctrl_rule_count で指定された分の情報が既に設定済みであった場合は、(-1)を返却します。

3.21.3.17 DshPutCjOutRuleInfo() OutRule 情報を追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutCjOutRuleInfo(
    TCTRL_SPEC *info,           // ProcessCtrlSpec 情報構造体のポインタ
    int status,                 // Material 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ズ
    void *value                 // ルール値の格納ポインタ
);
```

[.NET VB]

```
Function DshPutCjOutRuleInfo (
    ByRef info As dsh_info.TCTRL_SPEC,
    ByVal status As Int32,
    ByVal fmt As Int32,
    ByVal asize As Int32,
    ByVal value As Int32) As Int32
```

[.NET C#]

```
int DshPutCjOutRuleInfo(
    ref TCTRL_SPEC info,
    int status,
    int fmt,
    int asize,
    byte[] value );
```

(2) 引数

info
TCTRL_SPEC 構造体のポインタです。

status
Material Status です。

fmt
ルール値 value のフォーマットです。

asize
ルール値 value の配列サイズです。

value
ルール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)	Out Rule 情報が指定数を超えている。

(4) 説明

先に DshInitCjProcessCtrlSpec () で初期設定された info 構造体の中の out_rule_list に Out Rule 情報を 1 個加えます。

本関数が実行される順に情報リスト out_rule_list 上に、Out Rule 情報を追加していきます。設定後 0 を返却します。

もし、info内のout_rule_count で指定された分の情報が既に設定済みであった場合は、(-1)を返却します。

3.21.3.18 DshInitCjPRJobStatusList() PRJobStatusList 情報の追加

(1) 呼出書式

[C, C++]

```
API int APIX DshInitCjPRJobStatusList(
    TPRJ_STATE_LIST *info,           // プロセスジョブ 状態情報構造体のポインタ
    int prj_count                    // プロセスジョブ 状態リストのサイズ
);
```

[.NET VB]

```
Sub DshInitCjPRJobStatusList (
    ByVal info As IntPtr,
    ByVal prj_count As Int32)
```

[.NET C#]

```
void DshInitCjPRJobStatusList(
    ref TPRJ_STATE_LIST info,
    int prj_count );
```

(2) 引数

info

新規に取得した TPRJ_STATE_LIST 情報構造体メモリのポインタです。

prj_count

プロセスジョブ状態リストのサイズです。

(3) 戻り値

なし。

(4) 説明

属性インデクス EN_PRJobStatusList に適用されます。

TPRJ_STATE_LIST 構造体のためのメモリは本関数が取得します。そして、与えられた引数を設定します。また prj_count 分のプロセスジョブ状態情報設定のためのリストを prj_list と state_list を準備します。

プロセスジョブ ID と状態情報の追加設定は DshPutCjPRJobStatus() を使って設定することができます。

3.21.3.19 DshPutCjPRJobStatus() PRJob 状態情報を追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutCjPRJobStatus(
    TPRJ_STATE_LIST *info,           // プロセスジョブ状態情報構造体のポインタ
    char *prjid,                     // プロセスジョブ ID 格納ポインタ
    int state                          // 状態値
);
```

[.NET VB]

```
Function DshPutCjPRJobStatus (
    ByRef info As dsh_info.TPRJ_STATE_LIST,
    ByVal prjid As String,
    ByVal state As Int32) As Int32
```

[.NET C#]

```
int DshPutCjPRJobStatus(
    ref TPRJ_STATE_LIST info,
    byte[] prjid,
    int state );
```

(2) 引数

info

プロセスジョブ状態リスト TPRJ_STATE_LIST 構造体のポインタです。

prjid

プロセスジョブ ID です。

state

指定された prjid の状態値です。

(3) 戻り値

戻り値	意味
0	正常に追加できた。
(-1)	プロセスジョブ状態情報の数が指定数を超えている。

(4) 説明

先に DshInitCjPRJobStatusList() で初期設定された info 構造体の中の prj_list にプロセスジョブ ID を state_list に state 情報を 1 個加えます。

本関数が実行される順に prjid と state を prj_list と、state_list 上に追加していきます。設定後 0 を返却します。

もし、info 内の prj_count で指定された分の情報が既に設定済みであった場合は、(-1) を返却します。

3.21.3.20 DshInitCjPauseEvent() Pause Event 情報の追加

(1) 呼出書式

[C, C++]

```
API int APIX DshInitCjPauseEvent(
    TPAUSE_EVENT *info,           // Pause Event リスト構造体のポインタ
    int ce_count                 // Pause Event リストのサイズ
);
```

[.NET VB]

```
Sub DshInitCjPauseEvent (
    ByVal info As IntPtr,
    ByVal ce_count As Int32)
```

[.NET C#]

```
void DshInitCjPauseEvent(
    ref TPAUSE_EVENT info,
    int ce_count );
```

(2) 引数

info

新規に取得した TPAUSE_EVENT 情報構造体メモリのポインタです。

ce_count

PAUSE EVENT リストのサイズです。

(3) 戻り値

なし。

(4) 説明

属性インデクス EN_PauseEvent に適用されます。

TPAUSE_EVENT 構造体のためのメモリは本関数が取得します。そして、与えられた引数を設定します。また ce_count 分の CEID(収集イベント)設定のためのリスト ceid_list を準備します。

CEID の追加設定は DshPutCjPauseEvent() を使って設定することができます。

3.21.3.21 DshPutCjPauseEvent () Pause Event 追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutCjPauseEvent(
    TPAUSE_EVENT *info,          // Pause Event 情報用構造体のポインタ
    int          ceid           // 設定する CEID
);
```

[.NET VB]

```
Function DshPutCjPauseEvent (
    ByRef info As dsh_info.TPAUSE_EVENT,
    ByVal ceid As Int32) As Int32
```

[.NET C#]

```
int DshPutCjPauseEvent(
    ref TPAUSE_EVENT info,
    int ceid );
```

(2) 引数

info

CEID(収集イベント)を設定するための TPAUSE_EVENT 構造体のポインタです。

ceid

Pause Event として設定する CEID(収集イベント)です。

(3) 戻り値

戻り値	意味
0	正常に追加できた。
(-1)	CEID の数が指定数を超えている。

(4) 説明

先に DshInitCjPauseEvent() で初期設定された info 構造体の中に CEID を 1 個加えます。

本関数が実行される順に ceid を ceid_list 上に追加していきます。設定後 0 を返却します。

もし、info 内の ce_count で指定された分の CEID が既に設定済みであった場合は、(-1) を返却します。

3 . 21 . 3 . 22 DshMakeS14F9Response() - S14F9 の応答メッセージの生成

(1) 呼出書式

[c,C++]

```
API int APIX DshMakeS14F9Response(
    TOBJ_INFO *info,           // コントロールジョブ 情報格納領域のポインタ
    TOBJ_ERR_INFO *erinfo,    // S14F10 に設定する応答情報格納領域のポインタ
    DSHMSG *smsg,            // S14F10 メッセージ を格納するメッセージ 構造体のポインタ
    BYTE *buff,              // S14F10 のテキスト格納バッファポインタ
    int buff_size            // buff のバイトサイズ
);
```

[.NET VB]

```
Function DshMakeS14F9Response (
    ByRef pinfo As dsh_info.TOBJ_INFO,
    ByRef erinfo As dsh_info.TOBJ_ERR_INFO,
    ByRef smsg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS14F9Response(
    ref TOBJ_INFO pinfo,
    ref TOBJ_ERR_INFO erinfo,
    ref DSHMSG smsg,
    byte[] buff,
    int buff_size );
```

(2) 引数

info

コントロールジョブ情報が格納されている領域のポインタです。

erinfo

S14F10 メッセージに設定する応答情報が格納されている領域のポインタです。

msg

S14F10 応答メッセージ情報を格納するためのメッセージ構造体のポインタです。

buff

S14F10 応答メッセージのテキストを格納するためのバッファポインタです。

buff_size

buff のバイトサイズです。

(3) 戻り値

戻り値	意味
0	正常に生成できた。
(-1)	生成できなかった。(buff 領域不足)

(4) 説明

S14F9 に対する S14F10 応答メッセージを info に含まれるコントロールジョブ生成情報と応答情報に従って作成します。

応答情報内の、objack を S14F10 の OBJACK として設定します。
OBJACK はユーザが S14F9 コントロール生成メッセージを評価した結果です。

erinfo 応答情報の初期設定と情報設定にはそれぞれ DshInitTOBJ_ERR_INFO()、DshPutTOBJ_ERR_INFO() 関数を使用することができます。これらの関数については、次の説明書を参照してください。

“VOL- 1 5 / 1 5 3 . 24 オブジェクト関連メッセージ応答情報とエラー情報設定ライブラリ関数”

3.21.3.23 DshDecodeS14F11() - S14F11 を CJ 削除情報にデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS14F11(
    DSHMSG *smsg, // SECS メッセージ 情報構造体のポインタ
    TOBJ_INFO *info // デコードした情報を格納する構造体のポインタ
);
```

[.NET VB]

```
Function DshDecodeS14F11 (
    ByRef smsg As dshdr2.DSHMSG,
    ByRef info As dsh_info.TOBJ_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS14F11(
    ref DSHMSG smsg,
    ref TOBJ_INFO info );
```

(2) 引数

smsg

S14F11 の SECS メッセージ 情報が格納されている構造体のポインタです。

info

デコードした CJ 削除情報を格納するための構造体ポインタです。

(3) 戻り値

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

(4) 説明

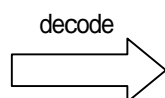
S14F11 メッセージに含まれる CJ 削除情報を、ユーザプログラムが処理しやすい TOBJ_INFO 構造体の中にデコードします。

削除コントロールジョブ情報を格納する構造体は S14F9 で使用したものと同じです。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTOBJ_INFO()関数を使って開放してください。

smsg S14F11

L,2
objspec
L,a
.
.



3 . 21 . 3 . 24 DshMakeS14F11Response() - S14F11 の応答メッセージの生成

(1) 呼出書式

[c,C++]

```
API int APIX DshMakeS14F11Response(
    TOBJ_INFO *info,           // コントロールジョブ 情報格納領域のポインタ
    TOBJ_ERR_INFO *erinfo,    // S14F12 に設定する応答情報格納領域のポインタ
    DSHMSG *smsg,            // S14F12 メッセージ を格納するメッセージ 構造体のポインタ
    BYTE *buff,              // S14F12 のテキスト格納バッファポインタ
    int buff_size            // buff のバイトサイズ
);
```

[.NET VB]

```
Function DshMakeS14F11Response (
    ByRef pinfo As dsh_info.TOBJ_INFO,
    ByRef erinfo As dsh_info.TOBJ_ERR_INFO,
    ByRef smsg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS14F11Response(
    ref TOBJ_INFO pinfo,
    ref TOBJ_ERR_INFO erinfo,
    ref DSHMSG smsg,
    byte[] buff,
    int buff_size );
```

(2) 引数

info

コントロールジョブ情報が格納されている領域のポインタです。

erinfo

S14F12 メッセージに設定する応答情報が格納されている領域のポインタです。

msg

S14F12 応答メッセージ情報を格納するためのメッセージ構造体のポインタです。

buff

S14F12 応答メッセージのテキストを格納するためのバッファポインタです。

buff_size

buff のバイトサイズです。

(3) 戻り値

戻り値	意味
0	正常に生成できた。
(-1)	生成できなかった。(buff 領域不足)

(4) 説明

S14F11 に対する S14F12 応答メッセージを info に含まれるコントロールジョブ削除情報と応答情報に従って作成します。

応答情報内の、objack を S14F12 の OBJACK として設定します。
OBJACK はユーザが S14F11 コントロール削除メッセージを評価した結果です。

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

3.21.3.25 DshDecodeS16F27 - S16F27 をCJ コマンド情報にデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS16F27(
    DSHMSG *msg, // SECS メッセージ情報構造体のポインタ
    TCJ_CMD_INFO *pinfo // デコードした情報を格納する構造体のポインタ
);
```

[.NET VB]

```
Function DshDecodeS16F27 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef info As dsh_info.TCJ_CMD_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS16F27(
    ref DSHMSG msg,
    ref TCJ_CMD_INFO info );
```

(2) 引数

msg

S16F27 の SECS メッセージ情報が格納されている構造体のポインタです。

pinfo

デコードした CJ コマンド情報を格納する構造体のポインタです。

(3) 戻り値

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

(4) 説明

S16F27 メッセージに含まれる CJ コマンド情報を、ユーザプログラムが処理しやすい TCJ_CMD_INFO 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTCJ_CMD_INFO() 関数を使って開放してください。

msg S16F27

L,3

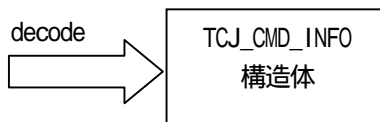
ctljobid

ctljobcmd

L,2

cpname

cpval



3.21.3.26 DshFreeTCJ_CMD_INFO() CJ コマンド情報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 *pinfo // メモリを開放したい情報が格納されている構造体のポインタ
);
```

[.NET VB]

```
Sub DshFreeTCJ_CMD_INFO (
    ByRef info As dsh_info.TCJ_CMD_INFO)
```

[.NET C#]

```
void DshFreeTCJ_CMD_INFO(
    ref TCJ_CMD_INFO info );
```

(2) 引数

pinfo

メモリを解放したいCJ コマンド情報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。

3.21.3.27 DshCopyTCJ_CMD_INFO() CJ コマンド情報構造体メモリのコピー

(1) 呼出書式

[C, C++]

```
API int APIX DshCopyTCJ_CMD_INFO(
    TCJ_CMD_INFO *dinfo,           // 北°-先のポインタ
    TCJ_CMD_INFO *sinfo           // 北°-元のポインタ
);
```

[.NET VB]

```
Function DshCopyTCJ_CMD_INFO (
    ByRef dinfo As dsh_info.TCJ_CMD_INFO,
    ByRef sinfo As dsh_info.TCJ_CMD_INFO) As Int32
```

[.NET C#]

```
int DshCopyTCJ_CMD_INFO(
    ref TCJ_CMD_INFO dinfo,
    ref TCJ_CMD_INFO sinfo );
```

(2) 引数

dinfo

CJ コマンド情報のコピー先構造体メモリのポインタです。

sinfo

コピー元の CJ コマンド情報が格納されている構造体メモリのポインタです。

(3) 戻り値

戻り値	意味
0	正常に北°-できた。
(-1)	sinfo または dinfo の値が NULL だったので北°-できなかった。

(4) 説明

sinfo が指す TCJ_CMD_INFO 構造体内に格納されている CJ コマンド情報を dinfo が指定する TCJ_CMD_INFO 構造体にコピーします。

dinfo 内のメンバーで新しいメモリが必要なものは本関数が取得します。

dinfo 内メンバーで確保されたメモリは、使用后、DshFreeTCJ_CMD_INFO()関数を使って開放してください。

3.21.3.28 DshInitTCJ_CMD_INFO CJ コマンド TCJ_CMD_INFO の初期設定

(1) 呼出書式

[C, C++]

```
API void APIX DshInitTCJ_CMD_INFO(
    TCJ_CMD_INFO *info,           // コントロールジョブコマンド情報 TCJ_CMD_INFO 構造体のポインタ
    char          *ctljobid,      // コントロールジョブ ID
    int           cj_cmd,         // コントロールジョブコマンド
    int           cp_count       // 設定できるコマンドパラメータ数
);
```

[.NET VB]

```
Sub DshInitTCJ_CMD_INFO (
    ByRef info As dsh_info.TCJ_CMD_INFO,
    ByVal ctljobid As String,
    ByVal cmd As Byte)
```

[.NET C#]

```
void DshInitTCJ_CMD_INFO(
    ref TCJ_CMD_INFO info,
    byte[] ctljobid,
    byte cmd );
```

(2) 引数

info

コントロールジョブコマンド TCJ_CMD_INFO 構造体のポインタです。このメンバーを初期設定します。

ctljobid

コントロールジョブ ID (文字列) のポインタです。

cj_cmd

コントロールジョブコマンドです。

```
#define CJ_Start          1
#define CJ_Pause         2
#define CJ_Resume        3
#define CJ_Cancel        4
#define CJ_Deselect      5
#define CJ_Stop          6
#define CJ_Abort         7
#define CJ_CJHQ          8
```

cp_count

コントロールジョブコマンド情報の中に設定できるコントロールジョブコマンドパラメータの数です。

(3) 戻り値

なし。

(4) 説明

info で指定された TCJ_CMD_INFO 構造体内に ct_ljobid で指定されたコントロールジョブ ID と cmd で指定されたコマンドを設定します。また cp_count 分のコマンドパラメータを格納できるリストを生成します。

info にコントロールジョブ ID を加えるためには DshAddTCJ_CMD_INFO() 関数を使用してください。

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

3.21.3.29 DshAddTCJ_CMD_INFO() コントロールジョブコマンドパラメータの追加

(1) 呼出書式

[C, C++]

```
API int APIX DshAddTRPJ_LIST(
    TCJ_CMD_INFO *info,           // コントロールジョブコマンド情報構造体のポインタ
    char *cpname,                // コマンドパラメータ名
    int cpval_fmt,               // コマンドパラメータのフォーマット( ICODE_A, ICODE_U1 etc )
    int cpval_size,              // パラメータデータの配列サイズ
    void *cpval                  // パラメータデータ格納ポインタ
);
```

[.NET VB]

```
Function DshAddTCJ_CMD_INFO (
    ByRef info As dsh_info.TCJ_CMD_INFO,
    ByVal cpname As String,
    ByVal cpval_fmt As Int32,
    ByVal cpval_size As Int32,
    ByVal cpval As IntPtr) As Int32
```

[.NET C#]

```
int DshAddTCJ_CMD_INFO(
    ref TCJ_CMD_INFO info,
    byte[] cpname,
    int cpval_fmt,
    int cpval_size,
    byte[] cpval );
```

(2) 引数

info

コントロールジョブコマンド情報構造体のポインタです。

cpname

加えたいコマンドパラメータ名が格納されているポインタです。

cpval_fmt

コマンドパラメータデータのフォーマットです。(ICODE_A, ICODE_U1 など)

cpval_size

コマンドパラメータデータの配列サイズです。

cpval

コマンドパラメータデータが格納されているポインタです。

(3) 戻り値

戻り値	意味
0	正常に追加できた。
(-1)	コマンドパラメータの数が既に指定数に達している。

(4) 説明

先に DshInitTCJ_CMD_INFO() で初期設定されたコントロールジョブコマンド構造体 info にコントロールジョブコマンドパラメータを 1 個追加します。追加はリストの空き位置に行います。

設定は、新たに TCMD_PARA パラメータ構造体のメモリを確保し、その中に cpname と cpval を設定し、cp_list にパラメータ構造体のポインタを設定します。

3.21.3.30 DshInitTCJ_CMD_ERR_INFO () CJ コマンド応答情報の初期化

(1) 呼出書式

[C, C++]

```
API void APIX DshInitTCJ_CMD_ERR_INFO(
    TCJ_CMD_INFO *info,           // S16F27 で得られた CJ コマンド 情報構造体のポインタ
    TCJ_CMD_ERR_INFO *erinfo,    // エラ-情報格納構造体リストのポインタ
    int acka,                    // ack データ
    int err_flag,               // エラ-有無フラグ (0/1)
    int err_code,              // エラ-コードです。
    char *err_text             // エラ-テキストです。
);
```

[.NET VB]

```
Sub DshInitTCJ_CMD_ERR_INFO (
    ByRef errinfo As dsh_info.TCJ_CMD_ERR_INFO,
    ByVal acka As Int32,
    ByVal err_flag As Int32,
    ByVal errcode As Int32,
    ByVal errtext As String)
```

[.NET C#]

```
void DshInitTCJ_CMD_ERR_INFO(
    ref TCJ_CMD_ERR_INFO errinfo,
    int acka,
    int err_flag,
    int errcode,
    byte[] errtext );
```

(2) 引数

info

S16F27 で得られた TCJ_CMD_INFO 情報構造体のポインタです。

erinfo

TCJ_CMD_ERR_INFO 応答情報構造体のポインタです。

acka

acka - ACK の値です。

err_flag

エラー情報(err_code, err_text)の有無を指定します。

= 0 はエラー情報なし、 != 0 はエラー情報があることを意味します。

err_code

err_flag != 0 のとき、S16F28 に err_code のエラーコードを設定します。

err_flag = 0 の場合は、無視されます。

err_text

err_flag != 0 のとき、S16F28 に err_text のエラーテキストを設定します。

err_flag = 0 の場合は、無視されます。

(3) 戻り値

なし。

(4) 説明

本関数は、CJ コマンド関連応答メッセージ TCJ_CMD_ERR_INFO 構造体に初期設定を行います。
erinfo で指定された構造体の acka メンバーに引数 acka の値を設定し、err_flag != 0 の場合、err_code
と err_text を設定します。
erinfo への情報設定用関数は本関数だけです。

ここで設定された erinfo は、DshMakeS16F27Response()関数の引数として使用します。

3.21.3.31 DshFreeTCJ_CMD_ERR_INFO() - コントロールジョブコマンド応答情報メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTCJ_CMD_ERR_INFO(
    TCJ_CMD_ERR_INFO *erinfo // メリを開放したい応答情報が格納されている構造体のポインタ
);
```

[.NET VB]

```
Sub DshFreeTCJ_CMD_ERR_INFO (
    ByRef info As dsh_info.TCJ_CMD_ERR_INFO)
```

[.NET C#]

```
void DshFreeTCJ_CMD_ERR_INFO(
    ref TCJ_CMD_ERR_INFO info );
```

(2) 引数

erinfo

メモリを解放したいコントロールジョブコマンド応答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。

3.21.3.32 DshMakeS16F27Response() - S16F27 の応答メッセージの生成

(1) 呼出書式

[C, C++]

```
API int APIX DshMakeS16F27Response(
    TCJ_CMD_INFO *info,           // CJコマンド 情報格納領域のポインタ
    TCJ_CMD_ERR_INFO *erinfo,    // S16F28 に設定する応答情報格納領域のポインタ
    DSHMSG *msg,                 // S16F28 メッセージ を格納するメッセージ 構造体のポインタ
    BYTE *buff,                  // S16F28 のテキスト格納バッファポインタ
    int buff_size                 // buff のバイトサイズ
);
```

[.NET VB]

```
Function DshMakeS16F27Response (
    ByRef info As dsh_info.TCJ_CMD_INFO,
    ByRef erinfo As dsh_info.TCJ_CMD_ERR_INFO,
    ByRef msg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS16F27Response(
    ref TCJ_CMD_INFO info,
    ref TCJ_CMD_ERR_INFO erinfo,
    ref DSHMSG msg,
    byte[] buff,
    int buff_size );
```

(2) 引数

info

CJ コマンド情報が格納されている領域のポインタです。

erinfo

S16F28 メッセージに設定する応答情報が格納されている領域のポインタです。

msg

S16F28 応答メッセージ情報を格納するためのメッセージ構造体のポインタです。

buff

S16F28 応答メッセージのテキストを格納するためのバッファポインタです。

buff_size

buff のバイトサイズです。

(3) 戻り値

戻り値	意味
0	正常に生成できた。
(-1)	生成できなかった。(buff 領域不足)

(4) 説明

S16F27 に対する S16F28 応答メッセージを info に含まれる CJ コマンド情報と応答情報に従って作成します。応答情報内の、acka を S16F28 の ACKA として設定します。

ACKA はユーザが S16F27CJ コマンドメッセージを評価した結果です。

erinfo の情報生成には、DshInitTCJ_CMD_ERR_INFO()関数を使用することができます。

3.21.4 ユーザ作成ライブラリ関数

3.21.4.1 DshResponseS14F10() S14F10 コントロールジョブ生成要求応答メッセージ

(1) 呼出書式

[C, C++]

```
API int APIX DshResponseS14F10(
    ID_TR trid, // DSHDR2 のトランザクション ID
    TOBJ_INFO *info, // コントロールジョブ生成要求メッセージ情報格納領域のポインタ
    TOBJ_ERR_INFO *erinfo // S14F10 応答情報格納用構造体のポインタ
);
```

[.NET VB]

```
Function DshResponseS14F10 (
    ByVal trid As Int32,
    ByRef info As dsh_info.TOBJ_INFO,
    ByRef erinfo As dsh_info.TOBJ_ERR_INFO) As Int32
```

[.NET C#]

```
int DshResponseS14F10(
    uint trid,
    ref TOBJ_INFO info,
    ref TOBJ_ERR_INFO erinfo );
```

(2) 引数

trid

S14F9 受信時に DSHEng4 から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

コントロールジョブ生成要求情報が格納されている構造体のポインタです。

erinfo

送信する応答メッセージ S14F10 に含まれる情報を格納するための構造体領域のポインタを指定します。

(3) 戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

(4) 説明

コントロールジョブ生成要求メッセージ S14F9 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL(dsh_ulib.dll)に含まれる関数ですが、ここでは DSHEng4 パッケージに標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている TOBJ_ERR_INFO 構造体に含まれている情報から S14F10 メッセージを組み立て、その後、S14F10 メッセージを送信します。

送信が完了したら、TOBJ_ERR_INFO の構造体で使用されたメモリを DshFreeTOBJ_ERR_INFO ()関数を使って開放します。

なお、S14F10 メッセージの組み立てに、DshMakeS14F10Response()関数を使用できます。

3.21.4.2 DshResponseS14F12() S14F12 コントロールジョブ削除要求応答メッセージ

(1) 呼出書式

[C, C++]

```
API int APIX DshResponseS14F12(
    ID_TR trid,           // DSHDR2 のトランザクション ID
    TOBJ_INFO *info,     // コントロールジョブ削除要求メッセージ情報格納領域のポインタ
    TOBJ_ERR_INFO *erinfo // S14F12 応答情報格納用構造体のポインタ
);
```

[.NET VB]

```
Function DshResponseS14F12 (
    ByVal trid As Int32,
    ByRef info As dsh_info.TOBJ_INFO,
    ByRef erinfo As dsh_info.TOBJ_ERR_INFO) As Int32
```

[.NET C#]

```
int DshResponseS14F12(
    uint trid,
    ref TOBJ_INFO info,
    ref TOBJ_ERR_INFO erinfo );
```

(2) 引数

trid

S14F11 受信時に DSHEng4 から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

コントロールジョブ削除要求情報が格納されている構造体のポインタです。

erinfo

送信する応答メッセージ S14F12 に含まれる情報を格納するための構造体領域のポインタを指定します。

(3) 戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

(4) 説明

コントロールジョブ削除要求メッセージ S14F11 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL(dsh_ulib.dll)に含まれる関数ですが、ここでは DSHEng4 パッケージに標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている TOBJ_ERR_INFO 構造体に含まれている情報から S14F12 メッセージを組み立て、その後、S14F12 メッセージを送信します。

送信が終わったら、TOBJ_ERR_INFO の構造体で使用されたメモリを DshFreeTOBJ_ERR_INFO ()関数を使って開

放します。

なお、S14F12 メッセージの組み立てに、DshMakeS14F12Response()関数を使用できます。

3.21.4.4 DshResponseS16F28() S16F28 コントロールジョブコマンド応答メッセージ

(1) 呼出書式

[C, C++]

```
API int APIX DshResponseS16F28(
    ID_TR trid,           // DSHDR2 のトランザクション ID
    TCJ_CMD_INFO *info,  // コントロールジョブコマンドメッセージ情報格納領域のポインタ
    TCJ_CMD_ERR_INFO *erinfo // S16F28 応答情報格納用構造体のポインタ
);
```

[.NET VB]

```
Function DshResponseS16F28 (
    ByVal trid As Int32,
    ByRef info As dsh_info.TCJ_CMD_INFO,
    ByRef erinfo As dsh_info.TCJ_CMD_ERR_INFO) As Int32
```

[.NET C#]

```
int DshResponseS16F28(
    uint trid,
    ref TCJ_CMD_INFO info,
    ref TCJ_CMD_ERR_INFO erinfo );
```

(2) 引数

trid

S16F27 受信時に DSHEng4 から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

コントロールジョブコマンド情報が格納されている構造体のポインタです。

erinfo

送信する応答メッセージ S16F28 に含まれる情報を格納するための構造体領域のポインタを指定します。

(3) 戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

(4) 説明

コントロールジョブコマンドメッセージ S16F27 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL(dsh_ulib.dll)に含まれる関数ですが、ここでは DSHEng4 パッケージに標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている TCJ_CMD_ERR_INFO 構造体に含まれている情報から S16F28 メッセージを組み立て、その後、S16F28 メッセージを送信します。

送信が終わったら、TCJ_CMD_ERR_INFO の構造体で使用されたメモリを DshFreeTCJ_CMD_ERR_INFO ()関数を使

って開放します。

なお、S16F28 メッセージの組み立てに、DshMakeS16F28Response()関数を使用できます。