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

ソフトウェア・パッケージ

DSHEng5 GEM 通信エンジン・クラス説明書

Vol－3

プロセス情報関連クラス

(PP, FPP, RECIPE, PRJ, CJ, CARRIER, SUBSTRATE)

２０１９年6月

株式会社データマップ

文書番号　DSHENG5-１9-30323-00

［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.	2019-06-28	初版	
2.			
3.			
4.			

目　次

１．はじめに

DSHEng5 GEM 通信エンジン説明書は、Vol-1 から６までの６つの Volume の分けられています。
本説明書の Vol 番号は３です。

本説明書では、プロセス関連情報のクラスの機能、コンストラクタ、プロパティ、メソッドなどについて説明します。

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

２．プロセス・プログラム情報クラス

プロセス・プログラム関連情報として以下の３種類のものがあります。

	記号	情報名と内容	使用するメッセージ
１．	PP	Process Program プロセスプログラム ID, PPBODY	S7F3, S7F5
2	FPP	Formatted Process rogram 書式付きプロセスプログラム ID,プロセスコマンドとパラメータ	S7F23, S7F25
3	RECIPE	Recipe - レシピ ID, 属性 ID と属性データ	S15F13

システムで実際に使用される情報は、上記３つの中の１つです。

ここでは、PP 情報に使用されるクラスについて説明します。

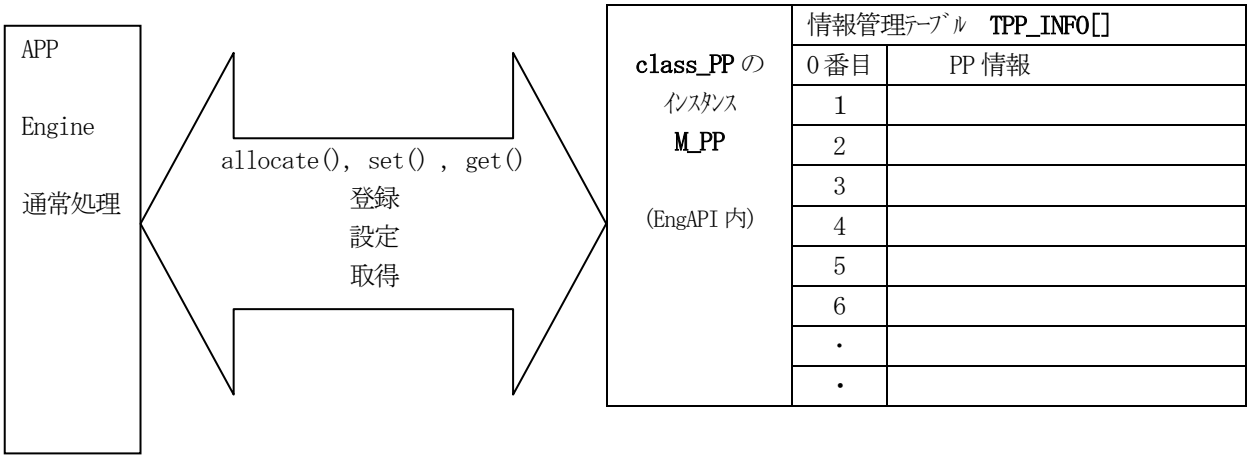
２．１　class_PP - プロセスプログラム

class_PP クラスは全ての**プロセスプログラム**の登録、参照、管理サービスを行うためのクラスです。

class_PP のインスタンス M_PP は、EngAPI クラスの中に生成されます。

各 PP 情報の保存には TPP_INFO クラスを使用します。（２．２で説明します）

PP 情報の構成と参照については概略以下の通りです。



s

２．１．１　コンストラクタ

	名前	説明
1.	public class_PP(int max_id)	ｲﾝｽﾀﾝｽを生成します。 引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_PP クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)

引数 max_id は管理する ID の最大数を指定します。プロパティ pp_info_tab[] の配列サイズになります。
(EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_PP を生成します。)

２．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1.	TPP_INFO[] pp_info_tab	PP 情報の登録テーブル	

TPP_INFO クラスについては、２．２　TPP_INFO クラス　の説明を参照ください。

２．１．３　メソッド

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

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int allocate_id()	PPID を登録します。
4	public int set()	指定 ID の PP 情報を設定します。
5	public int get()	指定 ID の PP 情報を取得します。
6	public int set_ppbody()	指定 ID の ppbody を設定します。
7	public string get_ppbody()	指定 ID の ppbody を取得します。
8	public void delete_all_id()	全登録 ID を削除します。
9	public void delete()	指定 ID の情報を削除します。

例えば、PP_123 の情報を取得したい場合、次のようなコーディングになります。

```
TPP_INFO info = new TPP_INFO();
int result = EngAPI.M_PP.get (PP_123, ref info);
```

２．１．３．１　get_id_count() - 登録されている ID 数の取得

DSHEng5 内に登録されている PPID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されている PPID 数を取得します。

２．１．３．２　get_id_list() - 登録されている ID リストの取得

DSHEng5 内に登録されている全 PPID をリストに取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大サイズ

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list リスト内に取得します。

戻り値は、取得した ID 数です。

２．１．３．３　allocate() - PPID の登録

指定された PPID を登録します。

【構文】

public int allocate_id(string id)

【引数】

id

登録したい PPID

【戻り値】

返却値	意　味
0	登録できた。(新規登録)
1	ID が登録済であった。
(-1)	登録できなかった。

【説明】

指定された PPID を管理下に登録します。
もし、既に登録されていた場合、PPID 以外の PP 情報をクリアします。戻り値=1 を返却します。
登録できるスペースが無かった場合は、(-1)を返却します。

２．１．３．４　set()　- PP 情報の設定

指定された ID の PP 情報を設定します。
未登録であれば、allocate します。そして設定します。

【構文】

public int set(string id, TPP_INFO src_info)

【引数】

id

設定したい PPID

src_info

設定したい PP 情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	src_info が null であった。

【説明】

指定された ID の PP 情報内に、src_info で指定されたインスタンスの情報を設定します。

もし、ID が未登録の場合は、最初に allocate()メソッドを使ってエンジン内部で登録します。
既に登録済の場合は、元の情報をクリアした後に設定します。

２．１．３．５　get()－PP 情報の取得

指定された ID の PP 情報を取得します。

【構文】

public int get(string id, ref TPP_INFO dst_info)

【引数】

id

　　取得したい PPID

dst_info

　　取得した PP 情報を保存するインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定された PPID が登録されていなかった。

【説明】

指定された ID の PP 情報を dst_info のインスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

２．１．３．６　set_ppbody()－PP 情報に ppbody を設定

指定された ID の PP 情報の中に PPBODY を設定します。

【構文】

public int set_ppbody(string id, string ppbody)

【引数】

id

　　設定したい PPID

ppbody

　　設定したい PPBODY

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定 ID が無かった。

【説明】

指定 ID インスタンスの ppbody プロパティに、引数の ppbody を設定します。

２．１．３．７　get_ppbody()－PP 情報の取得

指定された ID の PP 情報に設定されている PPBODY を取得します。

【構文】

public string get_ppbody(string id)

【引数】

id

取得したい PPID

【戻り値】

返却値	意　味
PPBODY	取得できた。
“”	指定された PP が登録されていなかった。

【説明】

指定された ID の PPBODY を取得します。

もし、ID が未登録の場合は、”” を返却します。

２．１．３．８　delete_all_id()－全 PP 情報の消去

登録されているすべての PP 情報を削除します。

【構文】

public void delete_all_id()

【引数】

なし。

【戻り値】

なし。

【説明】

登録されているすべての PP 情報を削除します。

２．１．３．９　delete()－PP 情報の削除

指定された ID の PP 情報を削除し、登録から外します。

【構文】

public void delete(string id)

【引数】

id

削除したい PPID

【戻り値】

なし。

【説明】

指定された ID の PP 情報を削除します。そして登録から外します。

２．２　TPP_INFO - プロセス・プログラム情報保存クラス

TPP_INFO クラスは、１個の PP 情報を保存するために使用します。

２．２．１　コンストラクタ

TPP_INFO クラスのインスタンスを生成します。

インスタンス pp_info を生成する例です。

（1）空のインスタンスを生成

```
TPP_INFO pp_info = new TPP_INFO();
```

（2）PP_123 の ID のインスタンスを生成します。

```
string PP_123 = “PP100”;
TPP_INFO pp_info = new TPP_INFO(PP_123);
```

（注）PP_123 の PP が登録されていない場合は空のインスタンスを生成します。

２．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string ppid	Process Program の ID
2	public string ppbody	Process Program の Body

２．２．３　メソッド

PP 情報クラス TPP_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int set()	PP 情報を設定します。
4	public int get()	PP 情報を取得します。
5	public void set_ppbody()	PPBODY を設定します。
6	public void set_ppbody()	PPBODY を取得します。
7	public static int copy()	インスタンスを別のインスタンスにコピーします。

２．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

２．２．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

２．２．３．３　set()　- PP 情報の設定

当該インスタンスに PPID と PPBODY を設定します。

【構文】

public int set(string id, string body)

【引数】

id

　　設定したい PPID

ppbody

　　設定したい PPBODY

【戻り値】

返却値	意　味
0	設定できた。
(-1)	id が null であった。

【説明】

当該インスタンスに、PPID と PPBODY を設定します。

２．２．３．４　get() - PP 情報の取得

当該インスタンスから PPID と PPBODY を取得します。

【構文】

public int get(ref string id, ref string ppbody)

【引数】

id

　　取得した PPID の保存用

body

　　取得した PPBODY の保存用

【戻り値】

返却値	意　味
0	取得できた。
(-1)	プロパティ ppid が null であった。

【説明】

当該インスタンスから PPID と PPBODY を取得します。

もし、PPID が null の場合は、(-1)を返却します。

２．２．３．５　set_ppbody()-PPBODY の設定

当該インスタンスに PPBODY を設定します。

【構文】

public void set(string body)

【引数】

ppbody

設定したい PPBODY

【戻り値】

なし。

【説明】

当該インスタンスに、PPBODY を設定します。

２．２．３．６　get_ppbodyt()-PPBODY の取得

当該インスタンスから PPBODY を取得します。

【構文】

public void get(ref string body)

【引数】

body

取得した PPBODY の保存用 string

【戻り値】

なし。

【説明】

当該インスタンスから PPID と PPBODY を取得します。

２．２．３．７　copy() - インスタンスのコピー

TPP_INFO クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TPP_INFO dst, TPP_INFO src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

TPP_INFO インスタンス src を dst にコピーします。

２．３　TPPINQ_INFO - プロセス・プログラムロード問合わせ情報クラス

TPPINQ_INFO クラスは、S7F1 メッセージの PP ロード問合せ情報を保存します。

２．３．１　コンストラクタ

TPPINQ_INFO クラスのインスタンスを生成します。

２．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string ppid	Process Program の ID
2	public uint length	ロードしたいプロセスプログラムのサイズ

２．３．３　メソッド

PP ロード問合せ情報クラス TPPINQ_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int set()	PP ロード問合せ情報を設定します。

２．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

２．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

２．３．３．３　set()　-PP ロード問合せ情報の設定

当該インスタンスに PPID と PP 情報のサイズを設定します。

【構文】

public void set(string ppid, uint len)

【引数】

id

問合せしたい PPID

len

PP 情報のサイズ

【戻り値】

なし。

【説明】

当該インスタンスに、PPID と PP 情報のサイズを設定します。

２．４　TPPID_LIST – プロセス・プログラム配列リストクラス

TPPID_LIST クラスは、S7F17, S7F20 メッセージの内容を保存します。

２．４．１　コンストラクタ

TPPID_LIST クラスのインスタンスを生成します。

２．４．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	ppid_list[]に設定されている PPID 数
2	public string[] ppid_list	PPID を保存する配列リスト

２．４．３　メソッド

PP ロード問合せ情報クラス TPPID_LIST のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int add()	ppid_list に ID を 1 個追加します。
4	public static int copy()	TPPID_LIST クラスのインスタンスをコピーします。

２．４．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。

Dispose()の後、このインスタンスを使用することはできません。

２．４．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

２．４．３．３　add() - 配列リストへPPIDを追加

ppid_list配列リストにPPIDを追加します。

【構文】

public void add(string ppid)

【引数】

id

追加したいPPID

【戻り値】

なし。

【説明】

当該インスタンスのppid_listリストにppidを追加します。
追加した後、count +1 します。

２．４．３．４　copy() - インスタンスのコピー

TPP_INFOクラスのインスタンスをコピーします。

【構文】

public static int copy(ref TPPID_LIST dst, TPPID_LIST src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	srcがnull であった。

【説明】

TPPID_LISTインスタンス srcをdstにコピーします。

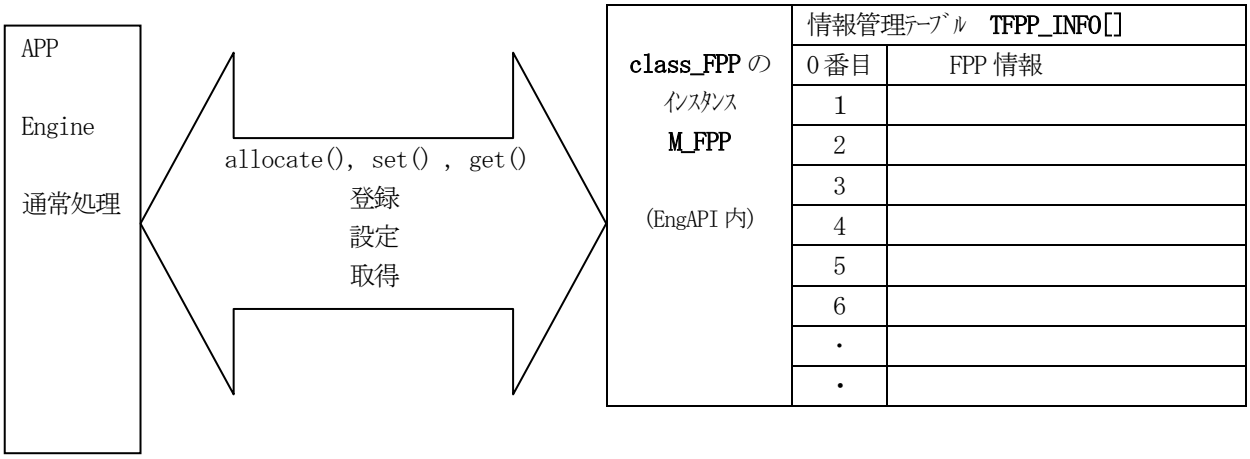
３．FPP － 書式付きプロセスプログラム情報クラス

FPP(Formatted Process Program)クラスについて説明します。

３．１　class_FPP － 書式付きプロセスプログラム

class_FPP クラスは全ての**プロセスプログラム**の登録、参照、管理サービスを行うためのクラスです。
各 FPP 情報の保存には TFPP_INFO クラスを使用します。（**３．２**で説明します）

FPP 情報の構成と参照については概略以下の通りです。



３．１．１　コンストラクタ

	名前	説明
1.	public class_FPP(int max_id)	ｲﾝｽﾀﾝｽを生成します。 引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_FPP クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)
引数 max_id は管理する ID の最大数を指定します。プロパティ fpp_info_tab[] の配列サイズになります。

(本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_FPP を生成します。)

３．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1.	TFPP_INFO[] fpp_info_tab	FPP 情報の登録テーブル	

TFPP_INFO クラスについては、３．２　TFPP_INFO クラス　の説明を参照ください。

３．１．３　メソッド

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

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int allocate_id()	FPPID を登録します。
4	public int set()	指定 ID の FPP 情報を設定します。
5	public int get()	指定 ID の FPP 情報を取得します。
6	public int set_mdln()	指定 ID のモデル名を設定します。
7	public string get_mdln()	指定 ID のモデル名を取得します。
8	public int set_softrev()	指定 ID のソフトウェア版番号を設定します。
9	public int get_softrev()	指定 ID のソフトウェア版番号を取得します。
10	public void delete_all_id()	全登録 ID を削除します。
11	public void delete()	指定 ID の情報を削除します。

例えば、FPP_123 の情報を取得したい場合、次のようなコーディングになります。

```
TFPP_INFO info = new TFPP_INFO();
int result = EngAPI.M_FPP.get (FPP_123, ref info);
```

３．３．３．１　get_id_count() - 登録されている ID 数の取得

DSHEng5 内に登録されている FPPID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されている FPPID 数を取得します。

３．３．３．２　get_id_list() - 登録されている ID リストの取得

DSHEng5 内に登録されている全 FPPID をリストに取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大容量

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list リスト内に取得します。

戻り値は、取得した ID 数です。

３．３．３．３　allocate() - FPP ID の登録

指定された FPPID を登録します。

【構文】

public int allocate_id(string id)

【引数】

id

予約したい FPPID

【戻り値】

返却値	意　味
0	登録できた。(新規登録)
1	ID が登録済であった。
(-1)	登録できなかった。

【説明】

指定された FPPID を管理下に登録します。
もし、既に登録されていた場合、FPPID 以外の FPP 情報をクリアします。戻り値=1 を返却します。
登録できるスペースが無かった場合は、(-1)を返却します。

３．３．３．４　set()　- FPP 情報の設定

指定された ID の FPP 情報を設定します。
未登録であれば、allocate します。そして設定します。

【構文】

public int set(string id, TFPP_INFO src_info)

【引数】

id

設定したい FPPID

src_info

設定したい FPP 情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	src_info が null であった。

【説明】

指定された ID に、src_info で指定されたインスタンスの情報を設定します。

もし、ID が未登録の場合は、最初に allocate()メソッドを使ってエンジン内部で登録します。
既に登録済の場合は、元の情報をクリアした後に設定します。

３．３．３．５　get() - FPP 情報の取得

指定された ID の FPP 情報を取得します。

【構文】

```
public int get(string id, ref TFPP_INFO dst_info)
```

【引数】

id

　　取得したい FPPID

dst_info

　　取得した FPP 情報を保存するインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定された FPPID が登録されていなかった。

【説明】

指定された ID の FPP 情報を dst_info のインスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

３．３．３．６　set_mdln() –model 名の設定

指定された ID の FPP 情報のモデル名を設定します。

【構文】

```
public int set_mdln(string id, string mdln)
```

【引数】

id

　　設定したい FPPID

mdln

　　設定したいモデル名

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定された FPPID が登録されていなかった。

【説明】

指定された ID の FPP 情報のモデル名をプロパティ mdln に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

３．３．３．７　get_mdln()–model 名の取得

指定された ID の FPP 情報の中のモデル名を取得します。

【構文】

public int get_mdln(string id, ref string mdln))

【引数】

id

　　取得したい FPPID

mdln

　　取得したモデル名の保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定された FPPID が登録されていなかった。

【説明】

指定された ID の FPP 情報の中のモデル名をプロパティ mdln から取得します。
もし、ID が未登録の場合は、(-1)を返却します。

３．３．３．８　set_softrev()- ソフトウェア版番号の設定

指定された ID の FPP 情報のソフトウェア版番号を設定します。

【構文】

public int set_softrev(string id, string softrev)

【引数】

id

　　設定したい FPPID

softrev

　　設定したいソフトウェア版番号

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定された FPPID が登録されていなかった。

【説明】

指定された ID の FPP 情報のソフトウェア版番号をプロパティ softrev に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

３．３．３．９　get_softrev()－ソフトウェア版番号の取得

指定された ID の FPP 情報のソフトウェア版番号を取得します。

【構文】

public int get_softrev(string id, ref string softrev))

【引数】

id

　　取得したい FPPID

softrev

　　取得したソフトウェア版番号の保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定された FPPID が登録されていなかった。

【説明】

指定された ID の FPP 情報のソフトウェア版番号をプロパティ softrev から取得します。
もし、ID が未登録の場合は、(-1)を返却します。

３．３．３．10　delete_all_id() 全 FPP 情報の消去

登録されているすべての FPP 情報を削除します。

【構文】

public void delete_all_id()

【引数】

なし。

【戻り値】

なし。

【説明】

登録されているすべての FPP 情報を削除します。

３．３．３．11　delete() - FPP 情報の削除

指定された ID の FPP 情報を削除し、登録から外します。

【構文】

public void delete(string id)

【引数】

id

削除したい FPPID

【戻り値】

なし。

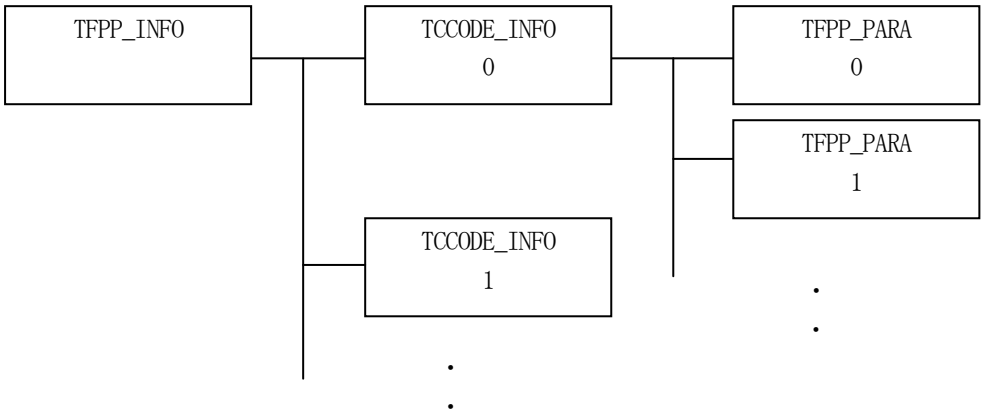
【説明】

指定された ID の FPP 情報を削除します。そして登録から外します。

３．２　TFPP_INFO - 書式付きプロセス・プログラム情報保存クラス

TFPP_INFO クラスは、１個の FPP 情報を保存するために使用します。

TFPP_INFO のクラス構成は以下のようになります。



３．２．１　コンストラクタ

TFPP_INFO クラスのインスタンスを生成します。

インスタンス fpp_info を生成する例です。

（1）空のインスタンスを生成

```
TFPP_INFO fpp_info = new TFPP_INFO();
```

（2）FPP_123 の ID のインスタンスを生成します。

```
string FPP_123 = “FPP100”;
TFPP_INFO pp_info = new TFPP_INFO(FPP_123);
```

（注）FPP_123 の FPP が登録されていない場合は空のインスタンスを生成します。

３．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string fppid	Formatted Process Program の ID
2	public string mdln	装置モデル名
3	public string softrev	ソフトウェア版番号
4	public int ccode_count	コマンド情報の数
5	public TFPP_CCODE[] ccode_list	コマンド情報配列リスト

３．２．３　メソッド

FPP 情報クラス TFPP_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_fppid_info()	FPPID、モデル名、ソフトウェア版版号を設定します。
4	public int add_ccode()	コマンド情報を１個 ccode_list[]に追加します。
5	public int set_ccode_list()	コマンド情報リストを設定します。
6	public int copy() public static int copy()	インスタンスを別のインスタンスにコピーします。

３．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．２．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．２．３．３　set_fppid_info()- FPP 情報の設定

当該インスタンスに FPPID、モデル名、ソフトウェア版番号を設定します。

【構文】

public void set_fppid_info(string id, string mdln, string softrev)

【引数】

id

　　設定したい FPPID

mdln

　　モデル番号

softrev

　　ソフトウェア版番号

【戻り値】

なし。

【説明】

当該インスタンスに、コマンド情報以外の情報を設定します。

３．２．３．４　add_ccode()- コマンド情報の追加

コマンド情報を ccode_list 配列リストに追加します。

【構文】

public int add_ccode(TFPP_CCODE info)

【引数】

info

　　追加したい TFPP_CCODE クラスのインスタンスです。

【戻り値】

返却値	意　味
ccode_count の値	ccode_list リストに設定したコマンド情報の数です。

【説明】

ccode_list コマンド情報リストに TFPP_CCODE クラスのインスタンスを追加します。
追加した後、ccode_list リストに設定できた全コマンド情報の数を返却します。

３．２．３．５　set_ccode_list() - コマンドリスト情報の設定

コマンドリスト情報を ccode_list 配列リストに設定します。

【構文】

public int set_ccode_list(TFPP_CCODE[] info, int count)

【引数】

list

設定したい TFPP_CCODE クラスの配列リストです。

【戻り値】

返却値	意　味
0	設定できた。

【説明】

最初に、プロパテイ ccode_list の内容をクリアします。
そして、list の内容を ccode_list コマンド情報リストにコピーします。

３．２．３．６　copy() - インスタンスのコピー

TFPP_INFO クラスのインスタンスをコピーします。

【構文】

public int copy(TFPP_INFO src)
public static int copy(ref TFPP_INFO dst, TFPP_INFO src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

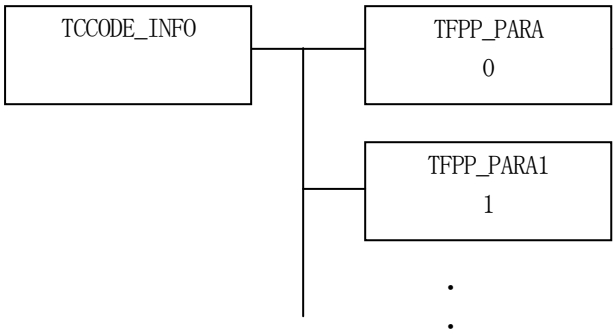
引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。本メソッドは static メソッドです。

３．３　TFPP_CCODE - FPP コマンド情報保存クラス

TFPP_CCODE クラスは、１個の FPP のコマンド情報を保存するために使用します。

TFPP_CCODE のクラス構成は以下のようになります。



３．３．１　コンストラクタ

TFPP_CCODE クラスのインスタンスを生成します。

３．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int ccode_fmt	コマンドのフォーマット
2	public int ccode_size	コマンド値のサイズ
3	public IntPtr ccode	コマンドコードの保存メモリ
4	public int para_count	パラメータ情報の数
5	public TFPP_PARA[] para_list	コマンド情報配列リスト

３．３．３　メソッド

FPP 情報クラス TFPP_CCODE のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_ccode()	コマンドコードを設定します。 format, size, c_code
4	public int add_ccode()	コマンド情報を１個 ccode_list[]に追加します。
5	public int set_ccode_list()	コマンド情報リストを設定します。
6	public int copy() public static int copy()	インスタンスを別のインスタンスにコピーします。

３．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．３．３．２　clear()- プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．３．３．３　set_set_ccode()- コマンドコードの設定

当該インスタンスにコマンドコードのフォーマット、サイズ、値を設定します。

【構文】

public int set_ccode(int fmt, int size, IntPtr c_code)
public int set_ccode(string c_code)

【引数】

fmt
　　コマンドコードのフォーマット
size
　　コード値サイズ
c_code
　　コマンドコードが保存されているメモリ
　　またはコマンド文字列

戻り値】

返却値	意　味
= 0	設定できた。
(-1)	設定できなかった

【説明】

当該インスタンス format, size, c_code プロパティに引数の値を設定します。

３．３．３．４　add_para() - コマンドパラメータの追加

コマンド情報を para_list リストに１個のパラメータを設定します。
指定する引数によって３種類のメソッドがあります。

【構文】

```
public int add_para(string para)
public int add_para( int fmt, int size, IntPtr paraval)
public int add_para( TFPP_PARA para_info)
```

【引数】

para
　　文字列のパラメータ
fmt
　　パラメータ値のフォーマット
size
　　パラメータ値のサイズ
paraval
　　パラメータ値が保存されているメモリ
para_info
　　TFPP_PARA クラスのインスタンス

【戻り値】

返却値	意　味
ccode_count の値	ccode_list リストに設定したコマンド情報の数です。

【説明】

para_list 配列リストの要素として１個のパラメータ情報を追加します。
TFPP_PARA のインスタンスを生成し、それを追加することになります。

追加したあと、para_count を +1 します。

３．３．３．５　set_para() - コマンドリスト情報の設定

パラメータ情報を para_list 配列リストに設定します。

【構文】

public int set_para(TFPP_PARA[] p_list, int p_count)

【引数】

p_list

　　設定したい TFPP_PARA クラスの配列リストです。

p_count

　　p_list に保存されているパラメータ数

【戻り値】

返却値	意　味
0	設定できた。

【説明】

最初に、プロパテイ para_list の内容をクリアします。

そして、p_list の内容を para_list コマンド情報リストに p_count 分だけコピーします。

３．３．３．６　copy() - インスタンスのコピー

TFPP_CCODE クラスのインスタンスをコピーします。

【構文】

public int copy(TFPP_CCODE src)

public static int copy(ref TFPP_CCODE dst, TFPP_CCODE src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。こちらは static メソッドです。

３．４　TFPP_PARA - FPP コマンドパラメータ情報保存クラス

TFPP_PARA クラスは、１個の FPP のコマンドパラメータ値を保存するために使用します。

３．４．１　コンストラクタ

３．４．１．１　コンストラクタ

TFPP_PARA クラスのインスタンスを生成します。

３．４．１．２　デストラクタ

TFPP_PARA のインスタンスを破棄します。
デストラクタはシステムから呼び出され、Dispose()メソッドを実行します。
Dispose()は、そのインスタンスが使用している資源（Unmanaged Memory）をシステムに返却します。

３．４．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int para_fmt	コマンドパラメータのフォーマット
2	public int para_size	コマンドパラメータ値のサイズ
3	public IntPtr para	コマンドパラメータ値保存メモリ

３．４．３　メソッド

TFPP_PARA のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int set()	コマンドパラメータ値を設定します。 format, size, para
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

３．４．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

３．４．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

３．４．３．３　set()　- コマンドパラメータコードの設定

当該インスタンスにコマンドパラメータのフォーマット、サイズ、値を設定します。

【構文】

```
public int set(int fmt, int size, IntPtr para)
public int set(string para)
```

【引数】

fmt

　　コマンドパラメータコードのフォーマット

size

　　コード値サイズ

para

　　コマンドパラメータコードが保存されているメモリ
　　またはコマンドパラメータ文字列

戻り値】

返却値	意　味
= 0	設定できた。
(-1)	設定できなかった

【説明】

当該インスタンス format, size, para プロパティに引数の値を設定します。
引数が string　para の場合は非管理メモリを確保し、文字列をコピーした上で設定します。

３．４．３．４　copy() - インスタンスのコピー

TFPP_PARA クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TFPP_PARA dst, TFPP_PARA src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src インスタンスから dst インスタンスへコピーします。static メソッドです。

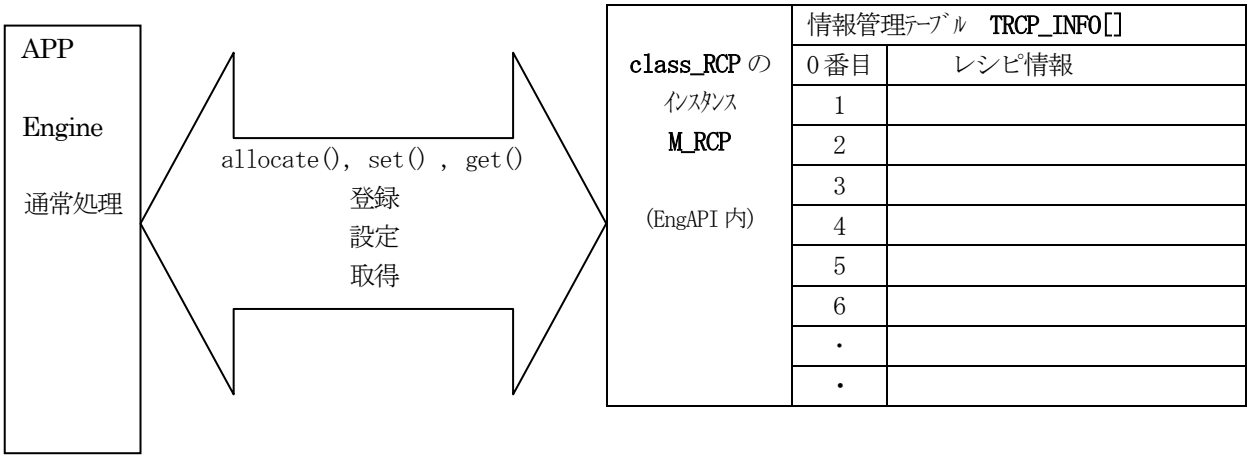
４．RCP － レシピ情報クラス

RCP（レシピ）クラスについて説明します。

４．１　class_RCP － レシピ

class_RCP クラスは全ての**レシピ**の登録、参照、管理サービスを行うためのクラスです。
各レシピ情報の保存には TRCP_INFO クラスを使用します。（４．２で説明します）

レシピ情報の構成と参照については概略以下の通りです。



４．１．１　コンストラクタ

	名前	説明
1.	public class_RCP(int max_id)	ｲﾝｽﾀﾝｽを生成します。　引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_RCP クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)
引数 max_id は管理する ID の最大数を指定します。プロパティ rcp_info_tab[] の配列サイズになります。

(本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_RCP を生成します。)

４．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1.	TRCP_INFO[] rcp_info_tab	レシピ情報の登録テーブル	

TRCP_INFO クラスについては、４．２　TRCP_INFO クラス　の説明を参照ください。

４．１．３　メソッド

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

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int allocate_id()	レシピ ID を登録します。
4	public int set()	指定 ID のレシピ情報を設定します。
5	public int get()	指定 ID のレシピ情報を取得します。
6	public int set_rcpbody()	指定 ID のレシピボディを設定します。
7	public string get_rcpbody()	指定 ID のレシピボディを取得します。
8	public int set_state()	指定 ID のレシピ状態を設定します。
9	public int get_state()	指定 ID のレシピ状態を取得します。
10	public int rename_rcpid()	レシピ名を変えます。
11	public void delete_all_id()	全登録 ID を削除します。
12	public void delete()	指定 ID の情報を削除します。

例えば、RCP_123 の情報を取得したい場合、次のようなコーディングになります。

```
TRCP_INFO info = new TRCP_INFO();
int result = EngAPI.M_RCP.get (RCP_123, ref info);
```

４．３．３．１　get_id_count() - 登録されている ID 数の取得

DSHEng5 内に登録されているレシピ ID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されているレシピ ID 数を取得します。

４．３．３．２　get_id_list() - 登録されている ID リストの取得

DSHEng5 内に登録されている全レシピ ID をリストに取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大容量

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list リスト内に取得します。

戻り値は、取得した ID 数です。

４．３．３．３　allocate() - レシピ ID の登録

指定されたレシピ ID を登録します。

【構文】

```
public int allocate_id(string id)
```

【引数】

id

　　登録したいレシピ ID

【戻り値】

返却値	意　味
0	登録できた。(新規登録)
1	ID が登録済であった。
(-1)	登録できなかった。

【説明】

指定されたレシピ ID を管理下に登録します。
もし、既に登録されていた場合、レシピ ID 以外のレシピ情報をクリアします。そして、戻り値=1 を返却します。
登録できるスペースが無かった場合は、(-1)を返却します。

４．３．３．４　set()　- レシピ情報の設定

指定された ID のレシピ情報を設定します。
未登録であれば、ID を allocate します。そして設定します。

【構文】

```
public int set(string id, TRCP_INFO src_info)
```

【引数】

id

　　設定したいレシピ ID

src_info

　　設定したいレシピ情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	src_info が null であった。

【説明】

指定された ID に、src_info で指定されたインスタンスの情報を設定します。

もし、ID が未登録の場合は、最初に allocate()メソッドを使ってエンジン内部で登録します。
既に登録済の場合は、元の情報をクリアした後にレシピ情報を設定します。

４．３．３．５　get() - レシピ情報の取得

指定された ID のレシピ情報を取得します。

【構文】

public int get(string id, ref TRCP_INFO dst_info)

【引数】

id

　　取得したいレシピ ID

dst_info

　　取得したレシピ情報を保存するインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたレシピ ID が登録されていなかった。

【説明】

指定された ID のレシピ情報を dst_info のインスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

４．３．３．６　set_rcpbody() - レシピボディの設定

指定された ID のレシピ情報のレシピボディを設定します。

【構文】

public int set_rcpbody(string id, string rcpbody)

【引数】

id

　　設定したいレシピ ID

rcpbody

　　設定したいモデル名

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたレシピ ID が登録されていなかった。

【説明】

指定された ID のレシピ情報のレシピボディに引数 rcpbody を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

４．３．３．７　get_rcpbody() - model 名の取得

指定された ID のレシピ情報のレシピボディを取得します。

【構文】

public int get_rcpbody(string id, ref string rcpbody))

【引数】

id

　　取得したいレシピ ID

rcpbody

　　取得したレシピボディの保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたレシピ ID が登録されていなかった。

【説明】

指定された ID のレシピ情報のレシピボディを取得します。
もし、ID が未登録の場合は、(-1)を返却します。

４．３．３．８　set_state() - レシピ状態語の設定

指定された ID のレシピ情報のレシピ状態語を設定します。

【構文】

public int set_state(string id, int state)

【引数】

id

　　設定したいレシピ ID

state

　　設定したいレシピ状態語

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたレシピ ID が登録されていなかった。

【説明】

指定された ID のレシピ情報のレシピ状態語をプロパティ state に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

４．３．３．９　get_state()-レシピ状態語の取得

指定された ID のレシピ情報のレシピ状態語を取得します。

【構文】

public int get_state(string id, ref int state)

【引数】

id

　　取得したいレシピ ID

state

　　取得したレシピ状態語の保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたレシピ ID が登録されていなかった。

【説明】

指定された ID のレシピ情報のレシピ状態語を取得します。
もし、ID が未登録の場合は、(-1)を返却します。

４．３．３．10　rename_rcpid()-ID 改名

指定された ID 名を別の ID 名に改名します。

【構文】

public int rename_rcpid(string id, string new_id)

【引数】

id

　　元のレシピ ID

new_id

　　新しいレシピ ID

【戻り値】

返却値	意　味
0	改名できた。
(-1)	指定されたレシピ ID が登録されていなかった。

【説明】

指定された ID のレシピ ID を新しい ID に変えます。

４．３．３．11　delete_all_id()－全レシピ情報の消去

登録されているすべてのレシピ情報を削除します。

【構文】

public void delete_all_id()

【引数】

なし。

【戻り値】

なし。

【説明】

登録されているすべてのレシピ情報を削除します。

４．３．３．12　delete()－レシピ情報の削除

指定された ID のレシピ情報を削除し、登録から外します。

【構文】

public void delete(string id)

【引数】

id

削除したいレシピ ID

【戻り値】

なし。

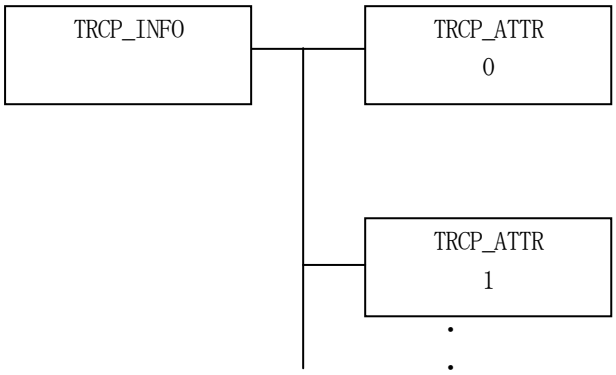
【説明】

指定された ID のレシピ情報を削除します。そして登録から外します。

４．２　TRCP_INFO - レシピ情報保存クラス

TRCP_INFO クラスは、１個のレシピ情報を保存するために使用します。

TRCP_INFO のクラス構成は以下のようになります。



４．２．１　コンストラクタ

TRCP_INFO クラスのインスタンスを生成します。

インスタンス rcp_info を生成する例です。

（1）空のインスタンスを生成

```
TRCP_INFO rcp_info = new TRCP_INFO();
```

（2）RCP_123 の ID のインスタンスを生成します。

```
string RCP_123 = “RCP100”;
TRCP_INFO pp_info = new TRCP_INFO(RCP_123);
```

（注）RCP_123 の RCP が登録されていない場合は空のインスタンスを生成します。

４．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string rcpid	レシピ ID
2	public string rcpbody	レシピボディ
3	public string state	レシピ状態語
4	public int para_count	コマンド情報の数
5	public TRCP_ATTR[] para_list	コマンド情報配列リスト
6	public bool update_flag	修正フラグ（ S15F13 の update 指定フラグ）

４．２．３　メソッド

レシピ情報クラス TRCP_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_rcpbody()	レシピボディを設定します。
4	public int add_para()	パラメータを 1 個 para_list[]に追加します。
5	public int set_para_list()	コマンド情報リストを設定します。
6	public int copy() public static int copy()	インスタンスを別のインスタンスにコピーします。

４．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

４．２．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

４．２．３．３　set_rcpbody() - レシピボディの設定

当該インスタンスにレシピボディ(recipe body)を設定します。

【構文】

public void set_rcpbody(string body)

【引数】

rcpbody

　　設定したいレシピボディ

【戻り値】

なし。

【説明】

当該インスタンスに、レシピボディを設定します。

４．２．３．４　add_para() - パラメータ情報の追加

コマンド情報を para_list 配列リストに追加します。

【構文】

public void add_para(string para_name, int format, int size, IntPtr para_val)
public void add_para(string para_name, string para_val)

【引数】

para_name

　　追加したいパラメータ名

format

　　パラメータ値のフォーマット

size

　　パラメータ値のデータサイズ

para_val

　　パラメータ値が」保存されているメモリ

【戻り値】

なし。

【説明】

para_list コマンド情報リストにパラメータを追加します。

パラメータ値は、別メモリを確保して設定します。
追加した後、プロパティ、 para_count +1 します。

パラメータ値が strin 型の para_val で与えられたメソッドについては、string を非管理メモリに置換して設定します。

４．２．３．５　set_para_list() - パラメータリスト情報の設定

パラメータ情報を para_list 配列リストに設定します。

【構文】

public int set_para_list(TRCP_ATTR[] info, int count)

【引数】

list

設定したい TRCP_ATTR クラスの配列リストです。

【戻り値】

返却値	意　味
0	設定できた。

【説明】

最初に、プロパテイ para_list の内容をクリアします。
そして、list の内容を para_list パラメータ情報リストにコピーします。

４．２．３．６　copy() - インスタンスのコピー

TRCP_INFO クラスのインスタンスをコピーします。

【構文】

public int copy(TRCP_INFO src)
public static int copy(ref TRCP_INFO dst, TRCP_INFO src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。こちらは static メソッドです。

４．３　TRCP_ATTR - レシピ属性情報保存クラス

TRCP_ATTR クラスは、１個のレシピ属性値を保存するために使用します。
本クラスは、S1５F13, S15F17 などで使用されます。

４．３．１　コンストラクタ

TRCP_ATTR クラスのインスタンスを生成します。

４．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string attrid	属性名
2	public int format	属性値のフォーマット
3	public int asize	属性値のサイズ
	public IntPtr attrdata	属性値保存メモリ

４．３．３　メソッド

TRCP_ATTR のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	属性名，属性データを設定します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

４．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

４．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

４．３．３．３　set()　- 属性名、属性値の設定

当該インスタンスに属性名と属性値を設定します。

【構文】

public void set(string attrid, int fmt, int size, IntPtr attrdata)
public void set(string attrid, string attrstr)

【引数】

attrid
　　属性名
fmt
　　属性値フォーマット
size
　　属性値サイズ
attrdata
　　属性値が保存されているメモリ
attrstr
　　文字列の属性値

【戻り値】

なし。

【説明】

当該インスタンス attrid, format, asize, attrdata プロパティに引数の値を設定します。
引数が attrstr の場合には、フォーマット ICODE_A で、メモリに移して設定します。

４．３．３．４　copy() - インスタンスのコピー

TRCP_ATTR クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TRCP_ATTR dst, TRCP_ATTR src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src インスタンスから dst インスタンスへコピーします。static メソッドです。

５．PRJ - プロセスジョブ情報クラス

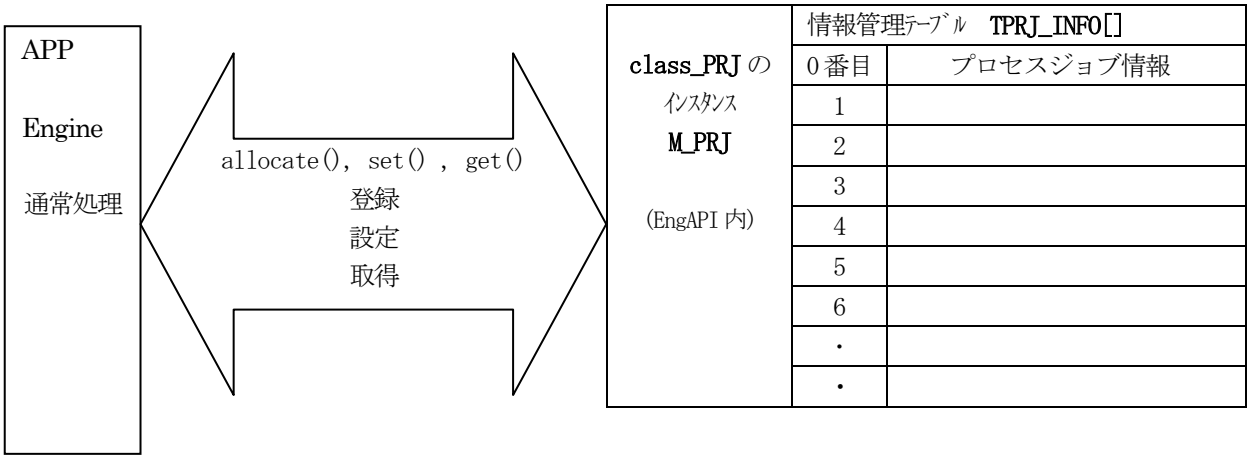
PRJ（プロセスジョブ）クラスについて説明します。

５．１　class_PRJ - プロセスジョブ

class_PRJ クラスは全ての**プロセスジョブ**の登録、参照、管理サービスを行うためのクラスです。

各プロセスジョブ情報の保存には TPRJ_INFO クラスを使用します。（後述します）

プロセスジョブ情報の構成と参照については概略以下の通りです。



５．１．１　コンストラクタ

	名前	説明
1.	public class_PRJ(int max_id)	ｲﾝｽﾀﾝｽを生成します。 引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_PRJ クラスのインスタンスを生成します。（DSHEng5 が生成します。APP が生成する必要はありません。）
引数 max_id は管理する ID の最大数を指定します。プロパティ prj_info_tab[] の配列サイズになります。

（本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_PRJ を生成します。）

５．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1.	TPRJ_INFO[] prj_info_tab	プロセスジョブ情報の登録テーブル	

TPRJ_INFO クラスについては、５．２　TPRJ_INFO クラス　の説明を参照ください。

５．１．３　メソッド

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

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int allocate_id()	プロセスジョブ ID を登録します。
4	public int set()	指定 ID のプロセスジョブ情報を設定します。
5	public int get()	指定 ID のプロセスジョブ情報を取得します。
6	public int set_mf()	指定 ID の MF 値を設定します。 (MF : Material Format)
7	public string get_mf()	指定 ID の MF 値を取得します。
8	public int get_car_count()	指定 ID の処理対象キャリア数を取得します。
9	public int add_car_list()	指定 ID の処理キャリアリストに 1 個キャリア情報を追加します。
10	public int get_car_list()	指定 ID の処理対象のキャリア情報リストを取得します。
11	public int get_mid_count()	指定 ID の処理対象 MID 数を取得します。
12	public int add_mid_list()	指定 ID の処理 MID リストに 1 個 MID 情報を追加します。
13	public int get_mid_list()	指定 ID の処理対象の MID 情報リストを取得します。
14	public int set_prrecipemethod()	指定 ID のレシピ処理方法を設定します。 (1=ﾚｼﾋﾟのみ、 2=可変ﾁｭｰﾆﾝｸﾞ付きﾚｼﾋﾟ)
15	public int get_prrecipemethod()	指定 ID のレシピ処理方法を取得します。

16	public int set_rcp_info()	指定 ID のレシピ情報を設定します。
17	public int get_rcp_info()	指定 ID のレシピ情報を取得します。
18	public int set_prprocessstart()	指定 ID の起動方法を設定します。 (true=自動スタート、 false=手動スタート)
19	public int get_prprocessstart()	指定 ID の起動方法を取得します。
20	public int get_ceid_count()	指定 ID の一時停止をする要因になるイベントリストに保存されている CEID 数を取得します。
21	public int add_ceid_list()	指定 ID の一時停止のイベントリストに CEID を追加します。
22	public int get_ceid_list()	指定 ID の一時停止の CE イベントリストの情報を取得します。
23	public int set_state()	指定 ID の state に値を設定します。
24	public int get_state()	指定 ID の state に値を取得します。
25	public void delete_all_id()	登録されている全登録 ID を削除します。
26	public void delete()	指定 ID を削除し、管理から外します。

例えば、PRJ_123 の情報を取得したい場合、次のようなコーディングになります。

```
TPRJ_INFO info = new TPRJ_INFO();
int result = EngAPI.M_PRJ.get (PRJ_123, ref info);
```

ID を指定するメソッドは、個々のプロセスジョブのプロセス情報プロパティを参照します。
個々のプロセスジョブ情報は、**５．２で説明する TPRJ_INFO** クラスに保存されています。

５．１．３．１　get_id_count()－登録されているID数の取得

DSHEng5内に登録されているプロセスジョブID数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID数	登録ID数

【説明】

登録されているプロセスジョブD数を取得します。

５．１．３．２　get_id_list()－登録されているIDリストの取得

DSHEng5内に登録されている全プロセスジョブIDをリストに取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　IDを保存するリスト

max_size

　　id_list配列の最大容量

【戻り値】

返却値	意　味
ID数	id_listリストに取得したID数

【説明】

登録されているIDをid_listリスト内に取得します。

戻り値は、取得したID数です。

５．１．３．３　allocate() - プロセスジョブ ID の登録

指定されたプロセスジョブ ID を登録します。

【構文】

public int allocate_id(string id)

【引数】

id

予約したいプロセスジョブ ID

【戻り値】

返却値	意　味
0	登録できた。(新規登録)
1	ID が登録済であった。
(-1)	登録できなかった。

【説明】

指定されたプロセスジョブ ID を管理下に登録します。
もし、既に登録されていた場合、プロセスジョブ ID 以外のプロセスジョブ情報をクリアします。戻り値=1 を返却します。
登録できるスペースが無かった場合は、(-1)を返却します。

５．１．３．４　set()　- プロセスジョブ情報の設定

指定された ID のプロセスジョブ情報を設定します。
未登録であれば、allocate します。そして設定します。

【構文】

public int set(string id, TPRJ_INFO src_info)

【引数】

id

設定したいプロセスジョブ ID

src_info

設定したいプロセスジョブ情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	src_info が null であった。

【説明】

指定された ID に、src_info で指定されたインスタンスの情報を設定します。
もし、ID が未登録の場合は、最初に allocate()メソッドを使ってエンジン内部で登録します。
既に登録済の場合は、元の情報をクリアした後に設定します。

５．１．３．５　get()－プロセスジョブ情報の取得

指定された ID のプロセスジョブ情報を取得します。

【構文】

```
public int get(string id, ref TPRJ_INFO dst_info)
```

【引数】

id

　　取得したいプロセスジョブ ID

dst_info

　　取得したプロセスジョブ情報を保存するインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報を dst_info のインスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．６　set_mf()－MF 値の設定

指定された ID のプロセスジョブ情報の MF 値を設定します。

【構文】

```
public int set_mf(string id, string mf)
```

【引数】

id

　　設定したいプロセスジョブ ID

mf

　　設定したい MF(Material Format)値

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報の中に引数 mf を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．７　get_mf()－MF の取得

指定された ID のプロセスジョブ情報の MF 値を取得します。

【構文】

public int set_mf(string id, int mf)

【引数】

id

取得したいプロセスジョブ ID

mf

取得したプロセスジョブ MF 値の保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報の MF 値を取得します。
もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．８　get_car_count()－キャリア数取得

処理対象のキャリアの数を取得します。

【構文】

public int get_car_count(string id)

【引数】

id

プロセスジョブ ID

【戻り値】

返却値	意　味
>=0	キャリア数
(-1)	指定された ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ ID の処理するキャリア数を取得します。
キャリア数は、TPRJ_INFO の car_count プロパティ値になります。

５．１．３．９　add_car_list() - キャリア情報の追加

指定された ID のプロセスジョブ情報に処理対象キャリアの情報を追加します。

【構文】

public int add_car_list(string id, TCAR_INFO car_info)

【引数】

id

　　設定したいプロセスジョブ ID

car_info

　　追加したいキャリアの情報

【戻り値】

返却値	意　味
0	追加できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報（TPRJ_INFO クラス）の中の car_list 配列リストに追加したいキャリア情報を追加します。

もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．10　get_car_list() - キャリア情報の取得

指定された ID のプロセスジョブ情報に処理対象になっているすべてのキャリアの情報を取得します。

【構文】

public int get_car_list(string id, ref TCAR_INFO[] car_list)

【引数】

id

　　設定したいプロセスジョブ ID

car_list

　　取得した情報を保存するキャリア情報保存配列リスト

【戻り値】

返却値	意　味
>=0	取得できたキャリアの数
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報（TPRJ_INFO クラス）の中の car_list 配列リストから全キャリア情報を取得し、引数の car_list に保存します。戻り値は取得し保存できたキャリアの数です。

もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．11　get_mid_count() - MID 数取得

処理対象の MID の数を取得します。

【構文】

```
public int get_mid_count( string id )
```

【引数】

id

　　プロセスジョブ ID

【戻り値】

返却値	意　味
>=0	MID 数
(-1)	指定された ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ ID の処理する MID 数を取得します。
MID 数は、TPRJ_INFO の mid_count プロパティ値になります。

５．１．３．12　add_mid_list() - MID 情報の追加

指定された ID のプロセスジョブ情報に処理対象 MID の情報を追加します。

【構文】

```
public int add_mid_list( string id, string mid)
```

【引数】

id

　　設定したいプロセスジョブ ID

mid

　　追加したい MID

【戻り値】

返却値	意　味
0	追加できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報（TPRJ_INFO クラス）の中の mid_list 配列リストに追加したい MID を追加します。

もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．13　get_mid_list()－MID 情報の取得

指定された ID のプロセスジョブ情報に処理対象になっているすべての MID の情報を取得します。

【構文】

public int get_mid_list(string id, ref string [] mid_list)

【引数】

id

　　設定したいプロセスジョブ ID

mid_list

　　取得した MID を保存する配列リスト

【戻り値】

返却値	意　味
>=0	取得できた MID の数
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報 (TPRJ_INFO クラス) の中の mid_list 配列リストから全 MID 情報を取得し、引数の mid_list に保存します。戻り値は取得し、保存できた MID の数です。

もし、プロセスジョブ ID が未登録の場合は、(-1)を返却します。

５．１．３．14　set_prrecipemethod()－レシピ処理方法値の設定

指定された ID のプロセスジョブ情報のレシピ処理方法値を設定します。

【構文】

public int set_prrecipemethod(string id, int method)

【引数】

id

　　設定したいプロセスジョブ ID

method

　　設定したいレシピ処理方法値　(1=ﾚｼﾋﾟのみ、 2=可変ﾁｭｰﾆﾝｸﾞ付きﾚｼﾋﾟ)

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報のレシピ処理方法として引数 prrecipemethod の値を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．15　get_prrecipemethod()–model 名の取得

指定された ID のプロセスジョブ情報のレシピ処理方法値を取得します。

【構文】

public int get_prrecipemethod(string id, ref string prrecipemethod))

【引数】

id

　　取得したいプロセスジョブ ID

prrecipemethod

　　取得したプロセスジョブレシピ処理方法値の保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報のレシピ処理方法値を取得します。
もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．16　set_rcp_info()- レシピ情報の設定

指定された ID のプロセスジョブ情報 TPRJ_INFO クラス）のプロパティ　rcp_info にレシピ情報を設定します。

【構文】

public int set_rcp_info(string id, TRCP_INFO r_info)

【引数】

id

　　プロセスジョブ ID

r_info

　　設定したいレシピ情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定 ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報クラスのインスタンスの中に、r_info で指定されたレシピ情報を設定します。

５．１．３．17　get_rcp_info() - レシピ情報の取得

指定された ID のプロセスジョブ情報 TPRJ_INFO クラス）のプロパティ　rcp_info に保存されているレシピ情報を取得します。

【構文】

public int get_rcp_info(string id, ref TRCP_INFO r_info)

【引数】

id

　　プロセスジョブ ID

dst_info

　　取得したレシピ情報を保存するインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	①指定されたプロセスジョブ ID が登録されていなかった。 ②指定されたプロセスジョブの中にレシピ情報が無かった。

【説明】

指定された ID のプロセスジョブ情報クラスのインスタンスの中のレシピ情報を r_info に取得します。
もし、プロセス ID が未登録の場合は、(-1)を返却します。また、プロセスジョブの中にレシピ情報が設定されなかった場合も(-1)を返却します。

５．１．３．18　set_prprocessstart() - プロセスジョブ開始方法の設定

指定された ID のプロセスジョブ開始方法を設定します。

【構文】

public int set_prprocessstart(string id, int prprocessstart)

【引数】

id

　　設定したいプロセスジョブ ID

prprocessstart

　　設定したいプロセスジョブ開始方法値　(1=自動開始、 0=手動開始)

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ開始方法として引数 prprocessstart の値を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．19　get_prprocessstart()－プロセスジョブ開始方法の取得

指定された ID のプロセスジョブ開始方法の設定値を取得します。

【構文】

```
public int get_prprocessstart(string id, ref int prprocessstart)
```

【引数】

id

　　プロセスジョブ ID

prprocessstart

　　取得したプロセスジョブ開始方法設定値の保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ開始方法の設定値を取得します。
もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．20　get_ceid_count()－CEID 数取得

指定された ID のプロセスジョブが使用する CEID(イベント ID)の数を取得します。

【構文】

```
public int get_ceid_count( string id )
```

【引数】

id

　　プロセスジョブ ID

【戻り値】

返却値	意　味
>=0	CEID 数
(-1)	指定された ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ ID が使用する CEID 数を取得します。
CEID 数は、TPRJ_INFO の ceid_count プロパティ値になります。

５．１．３．21　add_ceid_list() - CEID 情報の追加

指定された ID のプロセスジョブが使用する CEID の情報を追加します。

【構文】

public int add_ceid_list(string id, UInt32 ceid)

【引数】

id

　　設定したいプロセスジョブ ID

ceid

　　追加したい CEID

【戻り値】

返却値	意　味
0	追加できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報（TPRJ_INFO クラス）の中の ceid_list 配列リストに追加したい CEID を追加します。

もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．22　get_ceid_list() - CEID 情報の取得

指定された ID のプロセスジョブ情報が使用するすべての CEID の情報を取得します。

【構文】

public int get_ceid_list(string id, ref UInt32[] ceid_list)

【引数】

id

　　設定したいプロセスジョブ ID

ceid_list

　　取得した CEID 情報を保存する配列リスト

【戻り値】

返却値	意　味
>=0	取得できた CEID の数
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報（TPRJ_INFO クラス）の中の ceid_list リストから全 CEID を取得し、引数の ceid_list に保存します。戻り値は取得し、保存できた CEID の数です。
もし、プロセスジョブ ID が未登録の場合は、(-1)を返却します。

５．１．３．23　set_state() - プロセスジョブ状態語の設定

指定された ID の状態語を設定します。

【構文】

public int set_state(string id, int state)

【引数】

id

　　設定したいプロセスジョブ ID

state

　　設定したい状態語

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報の状態語をプロパティ state に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．23　set_state() - プロセスジョブ状態語の設定

指定された ID の状態語を設定します。

【構文】

public int set_state(string id, int state)

【引数】

id

　　設定したいプロセスジョブ ID

state

　　設定したい状態語

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報の状態語をプロパティ state に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．24　get_state()－プロセスジョブ状態語の取得

指定された ID のプロセスジョブ情報の状態語を取得します。

【構文】

```
public int get_state(string id)
```

【引数】

id

　　取得したいプロセスジョブ ID

state

　　取得したプロセスジョブ状態語の保存先

【戻り値】

返却値	意　味
状態語の値	取得できた。
(-1)	指定されたプロセスジョブ ID が登録されていなかった。

【説明】

指定された ID のプロセスジョブ情報のプロセスジョブ状態語を取得し、それを返却します。
もし、ID が未登録の場合は、(-1)を返却します。

５．１．３．25　delete()－プロセスジョブ情報の削除

指定された ID のプロセスジョブ情報を削除し、登録から外します。

【構文】

```
public void delete(string id)
```

【引数】

id

　　削除したいプロセスジョブ ID

【戻り値】

なし。

【説明】

指定された ID のプロセスジョブ情報を削除します。そして登録から外します。

５．１．３．26　delete_all_id() 全プロセスジョブ情報の消去

登録されているすべてのプロセスジョブ情報を削除します。

【構文】

```
public void delete_all_id()
```

【引数】

なし。

【戻り値】

なし。

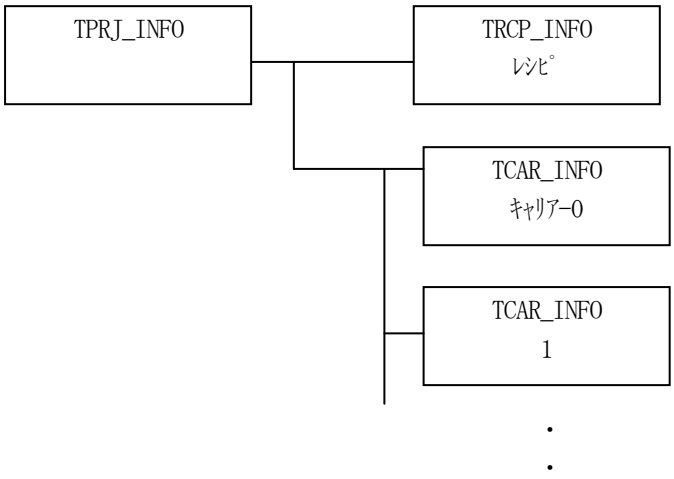
【説明】

登録されているすべてのプロセスジョブ情報を削除します。

５．２　TPRJ_INFO - プロセスジョブ情報保存クラス

TPRJ_INFO クラスは、１個のプロセスジョブ情報を保存するために使用します。

TPRJ_INFO のクラス構成は以下のようになります。



５．２．１　コンストラクタ

TPRJ_INFO クラスのインスタンスを生成します。

インスタンス prj_info を生成する例です。

（1）空のインスタンスを生成

```
TPRJ_INFO prj_info = new TPRJ_INFO();
```

（2）PRJ_123 の ID のインスタンスを生成します。

```
string PRJ_123 = “PRJ100”;
TPRJ_INFO prj_info = new TPRJ_INFO(PRJ_123);
```

（注）PRJ_123 の PRJ が登録されていない場合は空のインスタンスを生成します。

５．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string prjid	プロセスジョブ ID
2	public int state	プロセスジョブ状態語
3	public int cj_index	cj_index (not used)
4	public string mf	material format (13=carid, 14=mid)
5	public int car_count	キャリアの数 (mf=13)
6	public TCAR_INFO[] car_list	キャリア情報配列リスト(mf=13)
7	int mid_count	MID の数 (mf=14)
8	string[] mid_list	MID 配列リスト (mf=14)
9	public int prrecipemethod	レシピ処理方法
10	public TRCP_INFO rcp_info	レシピ情報
11	public int prprocessstart	処理スタート方法(1=自動, 0=手動)
12	public int ceid_count	CEID 数
13	public UInt32[] pause_ceid_list	CEID リスト （一時停止時用に使用されるイベント）

５．２．３　メソッド

プロセスジョブ情報クラス TPRJ_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void init_set()	prjid, mf などのプロパティを初期設定します。
4	public int add_car_list()	キャリア ID を 1 個 car_list[]に追加します。
5	public void add_mid_list()	MID を一個 mid_list[]に追加します。
6	public int set_rcp_info()	レシピ情報を rcp_info に設定します。
7	public void add_ceid_list()	CEID を pause_ced_list に追加します。
8	public int copy() public static int copy()	インスタンスを別のインスタンスにコピーします。

５．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．２．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

５．２．３．３　init_set()　- prjid, mf, rcpid など の設定

当該インスタンスにプロセスジョブ ID などの情報を設定します。

【構文】

public void init_set(string prjid, int mf, int prrecipemethod, string rcpid, int prprocessstart)

【引数】

prjid

　　設定したいプロセスジョブ I

mf

　　設定したい MF(material Format)値

prrecipemethod

　　レシピ処理方法

rcpid

　　レシピ ID

prprocessstart

　　処理スタート方法

【戻り値】

なし。

【説明】

当該インスタンスに、プロセスジョブ ID、MF、レシピ ID、処理方法、処理スタート方法を設定します。

５．２．３．４　add_car_list() - キャリア情報の追加

キャリア情報を car_list 配列リストに追加します。

【構文】

public void add_car_list(TCAR_INFO car_info)

【引数】

car_info

　　キャリア情報が保存されている TCAR_INFO クラスのインスタンス

【戻り値】

なし。

【説明】

car_info で指定されてたキャリア情報をプロパティ car_list リストに追加します。

追加した後、プロパティ、car_count +1 します。

５．２．３．６　set_rcp_info() - レシピ情報の設定

当該 TPRJ_INFO クラスのインスタンスのプロパティ　rcp_info にレシピ情報を設定します。

【構文】

public int set_rcp_info(TRCP_INFO src_info)

【引数】

src_info

設定したいレシピ情報　(TRCP_INFO クラスのインスタンス)

【戻り値】

返却値	意　味
0	設定できた。
(-1)	src_info が null であった。

【説明】

src_info で与えられたレシピ情報を　プロパティ rcp_info に設定します。

５．２．３．７　add_ceid_list() - CEID の追加

CEID を ceid_list 配列リストに追加します。

【構文】

public void add_ceid_list(UInt32 ceid)

【引数】

ceid

CEID

【戻り値】

なし。

【説明】

ceid をプロパティ ceid_list リストに追加します。
追加した後、プロパティ、ceid_count +1 します。

５．２．３．８　copy() - インスタンスのコピー

TPRJ_INFO クラスのインスタンスをコピーします。

【構文】

public int copy(TPRJ_INFO src)
public static int copy(ref TPRJ_INFO dst, TPRJ_INFO src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

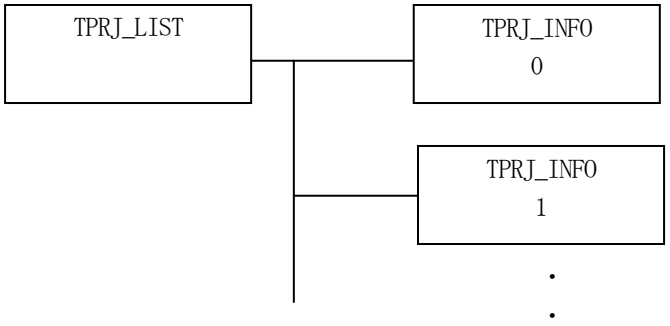
引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。

５．３　TPRJ_LIST - プロセスジョブ情報保存リストクラス

TPRJ_LIST クラスは、複数個のプロセスジョブ情報を保存するために使用する配列リストです。
メッセージ S16F15 の情報を保存するために使用します。

TPRJ_LIST のクラス構成は以下のようになります。



５．３．１　コンストラクタ

TPRJ_LIST クラスのインスタンスを生成します。

５．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int prj_count	prj_list リスト内に保存されている PRJ 情報の数です。
2	public TPRJ_INFO[] prj_list	TPRJ_INFO クラスを要素とするリストです。

５．３．３　メソッド

プロセスジョブ情報クラス TPRJ_LIST のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int add_info()	プロセスジョブ情報を prj_list[]に追加します。
4	public int set()	配列リストに保存されているプロセスジョブを prj_list[]に設定します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

５．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

５．３．３．３　add_info() - プロセスジョブ情報の追加

プロセスジョブ情報を prj_list 配列リストに追加します。

【構文】

public int add_info(TPRJ_INFO prj_info)

【引数】

prj_info

　　プロセスジョブ情報が保存されている TPRJ_INFO クラスのインスタンス

戻り値】

返却値	意　味
= 0	追加できた。
(-1)	prj_info が null であった。

【説明】

prj_info で指定されてたプロセスジョブ情報をプロパティ prj_list[]リストに追加します。

追加した後、プロパティ、prj_count +1 します。

５．３．３．４　set() - プロセスジョブ情報リストの設定

プロセスジョブ情報配列リストの情報を prj_list に設定します。

【構文】

public int set(TPRJ_INFO[] list, int p_count)

【引数】

list

　　プロセスジョブ情報が保存されている配列リスト

p_count

　　プロセスジョブ情報の数

【戻り値】

返却値	意　味
0	設定できた。
(-1)	list が null または無効の情報を検出した。

【説明】

配列リスト list に保存されている 1 個以上のプロセスジョブ情報を prj_list 配列リストに設定します。

設定した後、p_count をプロパティ prj_count に代入します。

５．３．３．５　copy() - インスタンスのコピー

TPRJ_LIST リストに含まれる PRJ_INFO クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TPRJ_LIST dst_info, TPRJ_LIST src_info)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。こちらは static メソッドです。

５．４　TPRJ_DEQ_INFO - プロセスジョブ削除 ID リストクラス

TPRJ_DEQ_INFO クラスは、削除したいプロセスジョブ ID の配列リストです。
メッセージ S16F17 の情報を保存するために使用します。

５．４．１　コンストラクタ

TPRJ_DEQ_INFO クラスのインスタンスを生成します。

５．４．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int prj_count	prj_list リスト内に保存されているプロセスジョブ ID の数です。
2	public string[] prj_list	削除対象プロセスジョブ ID の配列リストです。

５．４．３　メソッド

プロセスジョブ情報クラス TPRJ_DEQ_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int add()	プロセスジョブ ID を prj_list[]に追加します。
4	public int set()	配列リストに保存されているプロセスジョブ ID を prj_list[]に設定します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

５．４．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．４．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

５．４．３．３　add() - プロセスジョブ ID 情報の追加

プロセスジョブ ID を prj_list 配列リストに追加します。

【構文】

public int add(string prjid)

【引数】

prjid

　　プロセスジョブ ID

【戻り値】

なし。

【説明】

prj_info で指定されてたプロセスジョブ ID をプロパティ prj_list[]リストに追加します。
追加した後、プロパティ、prj_count +1 します。

５．４．３．４　set()　- プロセスジョブ ID リストの設定

プロセスジョブ ID 配列リストの内容を prj_list に設定します。

【構文】

public void set(string[] list, int count)

【引数】

list

　　プロセスジョブ ID が保存されている配列リスト

count

　　プロセスジョブ ID の数

【戻り値】

返却値	意　味
0	設定できた。
(-1)	list が null または無効の情報を検出した。

【説明】

list 配列リストに保存されている 1 個以上のプロセスジョブ ID を prj_list 配列リストに設定します。

設定した後、count をプロパティ prj_count に代入します。

５．４．３．５　copy() - インスタンスのコピー

TPRJ_DEQ_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

public static int copy(ref TPRJ_DEQ_INFO dst_info, TPRJ_DEQ_INFO src_info)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。こちらは static メソッドです。

５．５　TPRJ_STATE_LIST - プロセスジョブ状態情報リストクラス

TPRJ_STATE_LIST クラスは、プロセスジョブの状態情報の配列リストです。
プロセスジョブ状態情報の保存には TPRJ_STATE クラスが使用されます。

S16F20 がプロセスジョブ状態情報の通知のためのメッセージです。

５．５．１　コンストラクタ

TPRJ_STATE_LIST クラスのインスタンスを生成します。

５．５．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	prj_state_list リスト内に保存されている状態情報の数です。
2	public TPRJ_STATE [] prj_state_list	プロセスジョブ状態情報保存用の配列リストです。

５．５．３　メソッド

プロセスジョブ情報クラス TPRJ_STATE_LIST のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int add()	プロセスジョブ ID を prj_state_list[]に追加します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

５．５．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．５．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

５．５．３．３　add() - プロセスジョブ状態情報の追加

プロセスジョブ状態情報を prj_state_list 配列リストに追加します。

【構文】

public int add(TPRJ_STATE info)

【引数】

info

プロセスジョブ状態情報を保存している TSTATE_INFO クラスのインスタンスです。

【戻り値】

返却値	意　味
0	設定できた。
(-1)	info が null であった。

【説明】

prj_info で指定されてたプロセスジョブ ID をプロパティ prj_state_list[]リストに追加します。
追加した後、プロパティ、count +1 します。

５．５．３．４　copy() - インスタンスのコピー

TPRJ_STATE_LIST のインスタンスを他のインスタンスにコピーします。

【構文】

public static int copy(ref TPRJ_STATE_LIST dst_info, TPRJ_STATE_LIST src_info)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。こちらは static メソッドです。

５．６　TPRJ_STATE - プロセスジョブ状態情報クラス

TPRJ_STATE クラスは、1 個のプロセスジョブの状態情報の保存に使用されます。

５．６．１　コンストラクタ

TPRJ_STATE クラスのインスタンスを生成します。

５．６．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string prjid	プロセスジョブ ID です。
2	public int state	状態語です。

５．６．３　メソッド

プロセスジョブ情報クラス TPRJ_STATE のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	ID と状態語を設定します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

５．６．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．６．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

５．６．３．３　set()　- プロセスジョブ状態情報の設定

プロセスジョブ ID と状態語を設定します。

【構文】

public void set(string id, int st)

【引数】

id

　　プロセスジョブ ID

st

　　状態語です。

【戻り値】

なし。

【説明】

プロセスジョブ ID と状態語をプロパティに設定します。

５．６．３．４　copy() - インスタンスのコピー

TPRJ_STATE のインスタンスを他のインスタンスにコピーします。

【構文】

public static int copy(ref TPRJ_STATE dst_info, TPRJ_STATE src_info)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。こちらは static メソッドです。

５．７　TPRJ_CMD_INFO - プロセスジョブ・コマンド情報クラス

TPRJ_CMD_INFO クラスは、プロセスジョブに対するコマンド情報の保存に使用されます。

S16F5 がプロセスジョブに対するコマンド情報通知のメッセージです。

５．７．１　コンストラクタ

TPRJ_CMD_INFO クラスのインスタンスを生成します。

５．７．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string prjid	プロセスジョブ ID です。
2	public string cmd	コマンド名です。
3	public int cmd_index	cmd の数値表現コードです。
4	public int count	コマンドパラメータ数です。
5	public TCMD_PARA[] list	コマンドパラメータ保存用配列リストです。 （TCMD_PARA は ６．12　コマンドパラメータ情報クラス参照）

５．７．３　メソッド

プロセスジョブ情報クラス TPRJ_CMD_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	コマンド情報を設定します。
4	public void add_para()	コマンドパラメータをコマンド情報リスト list[]に追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

５．７．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

５．７．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイをすべてクリアします。

５．７．３．３　set()　- プロセスジョブコマンド情報の設定

プロセスジョブ ID とコマンドを設定します。

【構文】

public void set(string id, string cmd)

【引数】

id

　　プロセスジョブ ID

cmd

　　コマンド名です。

【戻り値】

なし。

【説明】

プロセスジョブ ID とコマンドをプロパティに設定します。

５．７．３．４　add_para() - コマンドパラメータ情報の追加

コマンドパラメータ情報を配列リスト list[]に追加します。

【構文】

public void add_para(string pname, int format, int size, IntPtr val)
public void add_para(string pname, string val)

【引数】

pname
　　追加したいパラメータ名
format
　　パラメータ値のフォーマット
size
　　パラメータ値のデータサイズ
val
　　パラメータ値が」保存されているメモリ

【戻り値】

なし。

【説明】

para_list コマンド情報リストにパラメータを追加します。

パラメータ値は、別メモリを確保して設定します。
追加した後、プロパティ、 para_count +1 します。

パラメータ値が strin 型の para_val で与えられたメソッドについては、string を非管理メモリに置換して設定します。

５．７．３．５　copy() - インスタンスのコピー

TPRJ_CMD_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

public static int copy(ref TPRJ_CMD_INFO dst_info, TPRJ_CMD_INFO src_info)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。本メソッドは static メソッドです。

６．CJ - コントロールジョブ情報クラス

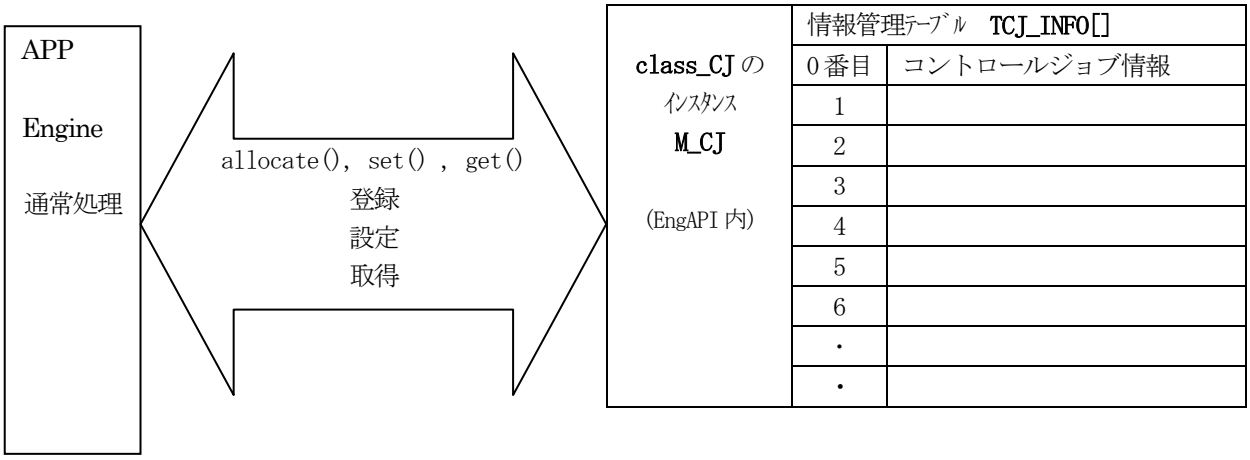
CJ（コントロールジョブ）クラスについて説明します。

６．１　class_CJ - コントロールジョブ

class_CJ クラスは全ての**コントロールジョブ**の登録、参照、管理サービスを行うためのクラスです。

各コントロールジョブ情報の保存には TCJ_INFO クラスを使用します。（６．２で説明します）

コントロールジョブ情報の構成と参照については概略以下の通りです。



６．１．１．　コンストラクタ

	名前	説明
1.	public class_CJ(int max_id)	ｲﾝｽﾀﾝｽを生成します。 引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_CJ クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)
引数 max_id は管理する ID の最大数を指定します。プロパティ cj_info_tab[] の配列サイズになります。

(本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_CJ を生成します。)

６．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1.	TCJ_INFO[] cj_info_tab	コントロールジョブ情報の登録テーブル	

TCJ_INFO クラスについては、６．１．２　TCJ_INFO クラス　の説明を参照ください。

６．１．３　メソッド

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

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int allocate_id()	コントロールジョブ ID を登録します。
4	public int set()	指定 ID のコントロールジョブ情報を設定します。
5	public int get()	指定 ID のコントロールジョブ情報を取得します。
6	public void delete()	指定 ID のコントロールジョブを削除します。
7	public int set_TMTRL_OUT_STAT()	指定 ID の MtrlOutByStatus 情報を設定します。
8	public int get_TMTRL_OUT_STAT()	指定 ID の MtrlOutByStatus 情報を取得します。
9	public int set_state()	指定 ID の state を設定します。
10	public int get_state()	指定 ID の state を取得します。
11	public int set_start_method()	指定 ID の StartMethod を設定します。
12	public int get_start_method()	指定 ID の StartMethod を取得します。
13	public int set_processing_order_mgmt()	指定 ID の ProcessingOrderMgmt を設定します。
14	public int get_processing_order_mgmt()	指定 ID の ProcessingOrderMgmt を取得します。
15	public int set_data_collect_plan()	指定 ID の DataCollectionPlan を設定します。

16	public int set_data_collect_plan()	指定 ID の DataCollectionPlan を取得します。
17	public int set_TMTRL_OUT_SPEC()	指定 ID の MtrlOutSpec を設定します。
18	public int get_TMTRL_OUT_SPEC()	指定 ID の MtrlOutSpec を取得します。
19	public int set_TPAUSE_EVENT()	指定 ID の PauseEvent を設定します。
20	public int get_ TPAUSE_EVENT()	指定 ID の PauseEvent を取得します。
21	public int set_TCTRL_SPEC_LIST()	指定 ID の ProcessingCtrlSpec を設定します。
22	public int get_ TCTRL_SPEC_LIST()	指定 ID の ProcessingCtrlSpec を取得します。
23	public int set_TPRJ_STATE_LIST()	指定 ID の PRJobStatusList を設定します。
24	public int get_ set_TPRJ_STATE_LIST()	指定 ID の PRJobStatusList を取得します。

例えば、CJ_123 の情報を取得したい場合、次のようなコーディングになります。

```
TCJ_INFO info = new TCJ_INFO();
int result = EngAPI.M_CJ.get (CJ_123, ref info);
```

コントロールジョブ情報は、**６．２で説明する TCJ_INFO** クラスに保存されています。

６．１．３．１　get_id_count() - 登録されているID数の取得

DSHEng5 内に登録されているコントロールジョブ ID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されているコントロールジョブ D 数を取得します。

６．１．３．２　get_id_list() - 登録されているIDリストの取得

DSHEng5 内に登録されている全コントロールジョブ ID をリストに取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大容量

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list リスト内に取得します。

戻り値は、取得した ID 数です。

６．１．３．３　allocate() - コントロールジョブ IDの登録

指定されたコントロールジョブ ID を登録します。

【構文】

```
public int allocate_id(string id)
```

【引数】

id

　　予約したいコントロールジョブ ID

【戻り値】

返却値	意　味
0	登録できた。(新規登録)
1	ID が登録済であった。
(-1)	登録できなかった。

【説明】

指定されたコントロールジョブ ID を管理下に登録します。
もし、既に登録されていた場合、コントロールジョブ ID 以外のコントロールジョブ情報をクリアします。そして戻り値=1 を返却します。登録できるスペースが無かった場合は、(-1)を返却します。

６．１．３．４　set()　- コントロールジョブ情報の設定

指定された ID のコントロールジョブ情報を設定します。
未登録であれば、allocate します。そして設定します。

【構文】

```
public int set(string id, TCJ_INFO src_info)
```

【引数】

id

　　設定したいコントロールジョブ ID

src_info

　　設定したいコントロールジョブ情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	src_info が null であった。

【説明】

指定された ID に、src_info で指定されたインスタンスの情報を設定します。
もし、ID が未登録の場合は、最初に allocate()メソッドを使ってエンジンに登録します。
既に登録済の場合は、元の情報をクリアした後に設定します。

６．１．３．５　get() - コントロールジョブ情報の取得

指定された ID のコントロールジョブ情報を取得します。

【構文】

public int get(string id, ref TCJ_INFO dst_info)

【引数】

id

　　取得したいコントロールジョブ ID

dst_info

　　取得したコントロールジョブ情報を保存するインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報を dst_info のインスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．６　set_TMTRL_OUT_STAT()　- MtrlOutByStatus リスト情報の設定

指定された ID のコントロールジョブ情報の MtrlOutByStatus リスト情報を設定します。

【構文】

```
public int set_TMTRL_OUT_STAT( string id, TMTRL_OUT_STAT_LIST list )
```

【引数】

id

　　設定したいコントロールジョブ ID

list

　　設定したい MtrlOutByStatus リスト情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報の中に、引数 list の MtrlOutByStatus リスト情報を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．７　get_TMTRL_OUT_STAT()　- MtrlOutByStatus リスト情報の取得

指定された ID のコントロールジョブ情報の MtrlOutByStatus リスト情報を取得します。

【構文】

```
public int get_TMTRL_OUT_STAT( string id, ref TMTRL_OUT_STAT_LIST list )
```

【引数】

id

　　取得したいコントロールジョブ ID

list

　　取得し、保存したい MtrlOutByStatus リスト

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報を、引数 list の MtrlOutByStatus インスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．８　set_state() - state 情報の設定

指定された ID のコントロールジョブの state を設定します。

【構文】

public int set_state(string id, int state)

【引数】

id

　　設定したいコントロールジョブ ID

state

　　設定したい state

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

引数 state を、指定された ID のコントロールジョブ情報の中のプロパティ state に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．９　get_state() - Cj state の取得

指定された ID のコントロールジョブの state を取得します。

【構文】

public int get_state(string id, ref int state)

【引数】

id

　　取得したいコントロールジョブ ID

state

　　取得したい state 保存領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブのプロパティ state を取得します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．10　set_start_method() - Cj start_method 情報の設定

指定された ID のコントロールジョブの start_method を設定します。

【構文】

public int set_start_method(string id, int start_method)

【引数】

id

　　設定したいコントロールジョブ ID

start_method

　　設定したい Cj start_method　(1=自動、0=手動)

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

引数 start_method を、指定された ID のコントロールジョブ情報の中のプロパティ start_method に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．11　get_start_method()　- Cj start_method の取得

指定された ID のコントロールジョブ情報の Cj start_method を取得します。

【構文】

public int get_start_method(string id, ref int start_method)

【引数】

id

　　取得したいコントロールジョブ ID

start_method

　　取得したい Cj start_method 保存領域　(1=自動、0=手動)

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報の中に、引数 start_method の Cj start_method を取得します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．12　set_processing_order_mgmt() - processing_order_mgmt 情報の設定

指定された ID のコントロールジョブの processing_order_mgmt を設定します。

【構文】

public int set_processing_order_mgmt(string id, int processing_order_mgmt)

【引数】

id

　　設定したいコントロールジョブ ID

processing_order_mgmt

　　設定したい Cj processing_order_mgmt

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報の中に、引数 processing_order_mgmt を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．13　get_processing_order_mgmt() - Cj processing_order_mgmtの取得

指定された ID のコントロールジョブ情報の processing_order_mgmt を取得します。

【構文】

public int get_processing_order_mgmt(string id, ref int processing_order_mgmt)

【引数】

id

　　取得したいコントロールジョブ ID

processing_order_mgmt

　　取得したい Cj processing_order_mgmt 保存領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報の中に引数 processing_order_mgmt の processing_order_mgmt を取得します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．14　set_data_collect_plan() - data_collect_plan情報の設定

指定された ID のコントロールジョブの data_collect_plan を設定します。

【構文】

public int set_data_collect_plan(string id, string data_collect_plan)

【引数】

id

　　設定したいコントロールジョブ ID

data_collect_plan

　　設定したい Cj data_collect_plan

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報の中に、引数 data_collect_plan を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．15　get_data_collect_plan() - Cj data_collect_planの取得

指定された ID のコントロールジョブ情報の data_collect_plan を取得します。

【構文】

public int get_data_collect_plan(string id, ref string data_collect_plan)

【引数】

id

　　取得したいコントロールジョブ ID

data_collect_plan

　　取得したい Cj data_collect_plan 保存領域

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報の中に引数 data_collect_plan の data_collect_plan を取得します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．16　set_TMTRL_OUT_SPEC() - MtrlOutSpec リスト情報の設定

指定された ID のコントロールジョブ情報の MtrlOutSpec リスト情報を設定します。

【構文】

```
public int set_TMTRL_OUT_SPEC( string id, TMTRL_OUT_SPEC_LIST list )
```

【引数】

id

　　設定したいコントロールジョブ ID

list

　　設定したい MtrlOutSpec リスト情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報の中に、引数 list の MtrlOutSpec リスト情報を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．17　get_TMTRL_OUT_SPEC() - MtrlOutSpec リスト情報の取得

指定された ID のコントロールジョブ情報の MtrlOutSpec リスト情報を取得します。

【構文】

```
public int get_TMTRL_OUT_SPEC( string id, ref TMTRL_OUT_SPEC_LIST list )
```

【引数】

id

　　取得したいコントロールジョブ ID

list

　　取得し、保存したい MtrlOutSpec リスト

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報を、引数 list の MtrlOutSpec インスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．18　set_TPAUSE_EVENT() - PauseEvent リスト情報の設定

指定された ID のコントロールジョブ情報の PauseEvent リスト情報を設定します。

【構文】

```
public int set_TPAUSE_EVENT( string id, TPAUSE_EVENT_LIST list )
```

【引数】

id

　　設定したいコントロールジョブ ID

list

　　設定したい PauseEvent リスト情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報の中に、引数 list の PauseEvent リスト情報を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．19　get_TPAUSE_EVENT() - PauseEvent リスト情報の取得

指定された ID のコントロールジョブ情報の PauseEvent リスト情報を取得します。

【構文】

```
public int get_TPAUSE_EVENT( string id, ref TPAUSE_EVENT_LIST list )
```

【引数】

id

　　取得したいコントロールジョブ ID

list

　　取得し、保存したい PauseEvent リスト

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報を、引数 list の PauseEvent インスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．20　set_TCTRL_SPEC_LIST() - ProcessingCtrlSpec リスト情報の設定

指定された ID のコントロールジョブ情報の ProcessingCtrlSpec リスト情報を設定します。

【構文】

```
public int set_TCTRL_SPEC_LIST( string id, TCTRL_SPEC_LIST list )
```

【引数】

id

　　設定したいコントロールジョブ ID

list

　　設定したい ProcessingCtrlSpec リスト情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報の中に、引数 list の ProcessingCtrlSpec リスト情報を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．21　get_TCTRL_SPEC_LIST() - ProcessingCtrlSpec リスト情報の取得

指定された ID のコントロールジョブ情報の ProcessingCtrlSpec リスト情報を取得します。

【構文】

```
public int get_TCTRL_SPEC_LIST( string id, ref TCTRL_SPEC_LIST list )
```

【引数】

id

　　取得したいコントロールジョブ ID

list

　　取得し、保存したい ProcessingCtrlSpec リスト

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報を、引数 list の ProcessingCtrlSpec インスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．22　set_TPRJ_STATE_LIST() - PRJobStatusList リスト情報の設定

指定された ID のコントロールジョブ情報の PRJobStatusList リスト情報を設定します。

【構文】

```
public int set_TPRJ_STATE_LIST( string id, TPRJ_STATE_LIST list )
```

【引数】

id

　　設定したいコントロールジョブ ID

list

　　設定したい PRJobStatusList リスト情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報の中に、引数 list の PRJobStatusList 情報を設定します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．23　get_TPRJ_STATE_LIST() - PRJobStatusList リスト情報の取得

指定された ID のコントロールジョブ情報の PRJobStatusList リスト情報を取得します。

【構文】

```
public int get_TPRJ_STATE_LIST( string id, ref TPRJ_STATE_LIST list )
```

【引数】

id

　　取得したいコントロールジョブ ID

list

　　取得し、保存したい PRJobStatusList リスト

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたコントロールジョブ ID が登録されていなかった。

【説明】

指定された ID のコントロールジョブ情報を、引数 list の PRJobStatusList インスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

６．１．３．24　delete() - コントロールジョブの削除

指定された ID のコントロールジョブを削除します。

【構文】

public void delete(string id)

【引数】

id

削除したいコントロールジョブ ID

【戻り値】

なし。

【説明】

指定された ID のコントロールジョブ情報を削除します。そして、ID を管理から外します。

６．１．３．25　delete_all_id() - 全コントロールジョブ情報の消去

登録されているすべてのコントロールジョブ情報を削除します。

【構文】

public void delete_all_id()

【引数】

なし。

【戻り値】

なし。

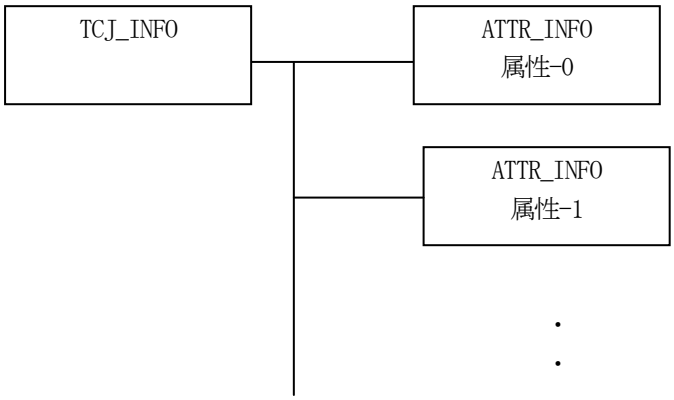
【説明】

登録されているすべてのコントロールジョブ情報を削除します。

６．２　TCJ_INFO - コントロール・ジョブ情報保存クラス

TCJ_INFO クラスは、１個のプロセスジョブ情報を保存するために使用します。

TCJ_INFO のクラス構成は以下のように属性情報から成ります。



６．２．１　コンストラクタ

TCJ_INFO クラスのインスタンスを生成します。

インスタンス cj_info を生成する例です。

（1）空のインスタンスを生成

```
TCJ_INFO cj_info = new TCJ_INFO();
```

（2）CJ_123 の ID のインスタンスを生成します。

```
string CJ_123 = “CJ100”;
TCJ_INFO cj_info = new TCJ_INFO(CJ_123);
```

（注）CJ_123 の CJ が登録されていない場合は空のインスタンスを生成します。

６．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string cjid	コントロールジョブ ID
2	public int state	コントロールジョブ状態語
3	public string objspec	OBJSPEC
4	public string objtype	OBJTYPE
5	public int attr_count	属性数
6	public TOBJ_ATTR_INFO[] attr_list	属性情報リスト （TOBJ_ATTR_INFO クラスは：　６．３　を参照）

６．２．３　メソッド

コントロールジョブ情報クラス TPRJ_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void init_set()	cjid, objspec, objtype を設定します。
4	public int add_attr()	属性情報を attr_list に追加します。
5	public int copy() public static int copy()	インスタンスを別のインスタンスにコピーします。

６．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．２．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．２．３．３　init_set() - cjid,, objspc, objtype など の設定

当該インスタンスに cjid, objspec, objtype を設定します。

【構文】

public void init_set(string cjid, string objspec, string objtype)

【引数】

cjid

　　コントロールジョブ ID

objspec

　　OBJSPEC

objtype

　　OBJTYPE

【戻り値】

なし。

【説明】

当該インスタンスに、CJID, OBJSPEC, OBJTYPE を設定します。

６．２．３．４　add_attr() - 属性情報の追加

属性情報を attr_list 配列リストに追加します。

【構文】

public int add_attr(TOBJ_ATTR_INFO info)

【引数】

info

　　追加したい TOBJ_ATTR_INFO クラスのインスタンスです。

【戻り値】

返却値	意　味
attr_count 値	attr_list リストに設定した属性情報の数です。
(-1)	info が null であった。

【説明】

attr_list 属性情報リストに TOBJ_ATTR_INFO クラスのインスタンスを追加します。
追加した後、attr_list リストに設定できた属性情報数を返却します。

６．２．３．５　copy() - インスタンスのコピー

TCJ_INFO クラスのインスタンスをコピーします。

【構文】

public int copy(TCJ_INFO src)
public static int copy(ref TCJ_INFO dst, TCJ_INFO src)

【引数】

dst
　　コピー先のインスタンス
src
　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

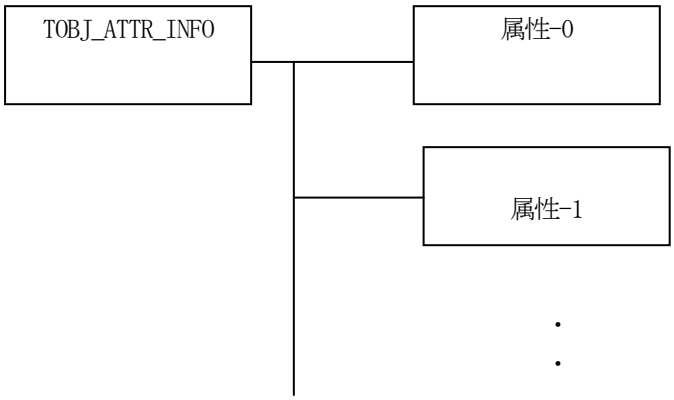
引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。こちらは static メソッドです。

６．３　TOBJ_ATTR_INFO - コントロール・ジョブ情報保存クラス

TOBJ_ATTR_INFO クラスは、１個のプロセスジョブ情報を保存するために使用します。

TOBJ_ATTR_INFO のクラス構成は以下のように属性情報から成ります。



６．３．１　コンストラクタ

TOBJ_ATTR_INFO クラスのインスタンスを生成します。

インスタンス attr_info を生成する例です。

（1）空のインスタンスを生成

```
TOBJ_ATTR_INFO attr_info = new TOBJ_ATTR_INFO();
```

（2）CJ_123 の ID のインスタンスを生成します。

```
string attr_id = “attr_list”;
int attr_index = class_const.EN_CarrierInputSpec;
TOBJ_ATTR_INFO attr_info = new TOBJ_ATTR_INFO(attr_id, attr_index);
```

属性インデクスと属性の対応を ６．3.2 プロパティ

６．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string attrid	属性 ID
2	public int attr_index	属性 ID を示すインデクス番号 index 値と属性 ID は下表に示します。
3	public string cjid	コントロールジョブ ID 属性情報
4	public TOBJ_TEXT_INFO car_input_spec_list	CarrierInputSpec 属性情報
5	public TOBJ_TEXT_INFO curr_prjob_list	CurrentPRJob 属性情報
6	public string data_collect_plan	データ収集プラン属性情報
7	public TMTRL_OUT_STAT_LIST mtrl_out_state_list	MtrlOutByStatus 属性情報
8	public TMTRL_OUT_SPEC_LIST mtrl_out_spec_list	MtrlOutSpec 属性情報
9	public TPAUSE_EVENT pause_event_list	PauseEvent 属性情報
10	public TCTRL_SPEC_LIST ctrl_spec_list	ProcessingCtrlSpec 属性情報
11	public TPRJ_STATE_LIST prj_state_list	PRJobStatusList 属性情報
12	public int processing_order_mgmt	ProcessingOrderMgmt 属性情報
13	public int start_method	StartMethod 属性情報
14	public int state	State 属性情報

次ページに属性インデクス(attr_index) 値と属性との対応については、次ページの
“☆　index 値と属性名対応表” を参照ください。

index 値と属性名対応表

class_const クラスに定数として定義されています。

index 値	属性指定記号
0	EN_ObjID
1	EN_CarrierInputSpec
2	EN_CurrentPRJob
3	EN_DataCollectionPlan
4	EN_MtrlOutByStatus
5	EN_MtrlOutSpec
6	EN_PauseEvent
7	EN_ProcessingCtrlSpec
8	EN_ProcessingOrderMgmt
9	EN_PRJobStatusList
10	EN_StartMethod
11	EN_State

６．３．３　メソッド

コントロールジョブ情報クラス TOBJ_ATTR_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_cjid()	コントロールジョブ ID を設定します。
4	public int add_attr()	属性情報を attr_list に追加します。
5	public void set_data_collection_plan()	データ収集プランを設定します。
6	public void set_car_input_spec_list()	CarrierInputSpec 属性情報を設定します。
7	public void set_curr_prjob_list()	CurrentPRJob 属性情報を設定します。
8	public void set_mtrl_out_state_list()	MtrlOutByStatus 属性情報を設定します。
9	public void set_mtrl_out_spec_list()	MtrlOutSpec 属性情報を設定します。
10	public void set_pause_event_list()	PauseEvent 属性情報を設定します。
11	public void set_ctrl_spec_list()	ProcessingCtrlSpec 属性情報を設定します。
12	public void set_prj_state_list()	PRJobStatusList 属性情報を設定します。
13	public void set_processing_order_mgmt()	ProcessingOrderMgmt 属性値を設定します。
14	public void set_start_method()	StartMethod 属性値を設定します。
15	public void set_state()	State 属性値を設定します。
16	public int copy() public static int copy()	インスタンスを別のインスタンスにコピーします。

６．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．３．３．３　set_cjid() - コントロールジョブIDの設定

当該インスタンスにコントロール ID を設定します。

【構文】

public void set_cjid(string id)

【引数】

cjid

設定したいコントロールジョブ ID

【戻り値】

なし。

【説明】

当該インスタンスに、コントロールジョブ ID を設定します。

６．３．３．４　set_data_collection_plan() - DataCollectionPlan の設定

当該インスタンスにデータ収集プランを設定します。

【構文】

public void set_data_collection_plan(string plan)

【引数】

plan

設定したいデータ収集プラン

【戻り値】

なし。

【説明】

当該インスタンスに、データ収集プラン plan を設定します。

６．３．３．５　set_car_input_spec_list() - CarrierInputSpec 情報の設定

当該インスタンスに CarrierInputSpec 属性情報を設定します。

【構文】

public void set_car_input_spec_list(TOBJ_TEXT_INFO list)

【引数】

list

設定したい CarrierInputSpec 属性情報

【戻り値】

なし。

【説明】

当該インスタンスに、引数の CarrierInputSpec 属性情報 list を car_input_spec_list に設定します。

６．３．３．６　set_curr_prjob_list() - CurrentPRJob情報の設定

当該インスタンスに CurrentPRJob 属性情報を設定します。

【構文】

public void set_curr_prjob_list(TOBJ_TEXT_INFO list)

【引数】

list

設定したい CurrentPRJob 属性情報

【戻り値】

なし。

【説明】

当該インスタンスに、引数の CurrentPRJob 属性情報 list を curr_prjob_list に設定します。

６．３．３．７　set_mtrl_out_state_list() - MtrlOutByStatus 情報の設定

当該インスタンスに MtrlOutByStatus 属性情報を設定します。

【構文】

public void set_mtrl_out_state_list(TMTRL_OUT_STAT_LIST list)

【引数】

list

設定したい MtrlOutByStatus 属性情報

【戻り値】

なし。

【説明】

当該インスタンスに、引数の MtrlOutByStatus 属性情報 list を mtrl_out_state_list に設定します。

６．３．３．８　set_mtrl_out_spec_list() - MtrlOutSpec情報の設定

当該インスタンスに MtrlOutSpec 属性情報を設定します。

【構文】

public void set_mtrl_out_spec_list(TMTRL_OUT_SPEC_LIST list)

【引数】

list

設定したい MtrlOutSpec 属性情報

【戻り値】

なし。

【説明】

当該インスタンスに、引数の MtrlOutSpec 属性情報 list を mtrl_out_spec_list に設定します。

６．３．３．９　sset_pause_event_list() - PauseEvent 情報の設定

当該インスタンスに PauseEvent 属性情報を設定します。

【構文】

public void set_pause_event_list(TPAUSE_EVENT ev_list)

【引数】

list

設定したい PauseEvent 属性情報

【戻り値】

なし。

【説明】

当該インスタンスに、引数の PauseEvent 属性情報 ev_list を pause_event_list に設定します。

６．３．３．10　set_ctrl_spec_list() - ProcessingCtrlSpec情報の設定

当該インスタンスに ProcessingCtrlSpec 属性情報を設定します。

【構文】

public void set_ctrl_spec_list(TCTRL_SPEC_LIST list)

【引数】

list

設定したい ProcessingCtrlSpec 属性情報

【戻り値】

なし。

【説明】

当該インスタンスに、引数の ProcessingCtrlSpec 属性情報 list を ctrl_spec_list に設定します。

６．３．３．11　set_prj_state_list() - PRJobStatusList情報の設定

当該インスタンスにPRJobStatusList属性情報を設定します。

【構文】

public void set_prj_state_list(TPRJ_STATE_LIST list)

【引数】

list

設定したいPRJobStatusList属性情報

【戻り値】

なし。

【説明】

当該インスタンスに、引数のPRJobStatusList属性情報listをprj_state_listに設定します。

６．３．３．12　set_start_method() - StartMethodの設定

当該インスタンスにStartMethod属性情報を設定します。

【構文】

public void set_start_method(int method)

【引数】

method

設定したいStartMethod

【戻り値】

なし。

【説明】

当該インスタンスに、引数のmethodをstart_methodに設定します。

６．３．３．13　set_state()－Stateの設定

当該インスタンスにState属性情報を設定します。

【構文】

public void set_state(int st)

【引数】

st

設定したいState

【戻り値】

なし。

【説明】

当該インスタンスに、引数のstをstateに設定します。

６．３．３．14　copy()－インスタンスのコピー

TOBJ_ATTR_INFOクラスのインスタンスをコピーします。

【構文】

public static int copy(ref TOBJ_ATTR_INFO dst, TOBJ_ATTR_INFO src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	srcがnull であった。

【説明】

srcからdstインスタンスへコピーします。

６．４　TOBJ_TEXT_INFO - テキスト・リスト情報クラス

TOBJ_TEXT_INFO クラスは、テキスト・リスト情報を保存するために使用します。

６．４．１　コンストラクタ

６．４．１．１　コンストラクタ

TOBJ_TEXT_INFO クラスのインスタンスを生成します。

６．４．１．２　デストラクタ

TOBJ_TEXT_INFO のインスタンスを破棄します。
デストラクタはシステムから呼び出され、Dispose()メソッドを実行します。
Dispose()は、そのインスタンスが使用している資源（Unmanaged Memory）をシステムに返却します。

６．４．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int text_count	保存されているテキスト（文字列）数
2	public string[] text_list	テキスト保存リスト

６．４．３　メソッド

コントロールジョブ情報クラス TOBJ_TEXT_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void add_text()	テキストを text_list に追加します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

６．４．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．４．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．４．３．３　add_text() - テキストの追加

テキスト１個を text_list 配列リストに追加します。

【構文】

public void add_text (string text)

【引数】

text

テキスト（文字列）

【戻り値】

なし。

【説明】

テキストをプロパティ text_list リストに追加します。
追加した後、プロパティ、text_count +1 します。

６．３．４．４　copy() - インスタンスのコピー

TOBJ_TEXT_INFO クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TOBJ_TEXT_INFO dst, TOBJ_TEXT_INFO src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。

６．５　TMTRL_OUT_STAT_LIST - MtrlOutByStatus情報リスト保存クラス

TMTRL_OUT_STAT_LIST クラスは、１個以上の MtrlOutByStatus 情報を保存するためのリストです。

TMTRL_OUT_STAT_LIST リストの要素は、TMTRL_OUT_STAT クラスのインスタンスになります。

６．５．１　コンストラクタ

TMTRL_OUT_STAT_LIST クラスのインスタンスを生成します。

６．５．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	mtr_list リストに保存されている MtrlOutByStatus 情報の数
2	public TMTRL_OUT_STAT[] mtr_list	MtrlOutByStatus 情報の配列リスト

６．５．３　メソッド

TMTRL_OUT_STAT_LIST クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int add()	MtrlOutByStatus 情報を mtr_list に追加します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

６．５．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．５．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．５．３．３　add() - MtrlOutByStatus 情報の追加

MtrlOutByStatus 情報を mtr_list 配列リストに追加します。

【構文】

public int add(TMTRL_OUT_STAT info)

【引数】

info

MtrlOutByStatus 情報（TMTRL_OUT_STAT クラスのインスタンス）

【戻り値】

なし。

【説明】

引数 info の情報を mtrl_list リストに追加します。

追加した後、プロパティ、count +1 します。

６．３．４．４　copy() - インスタンスのコピー

TMTRL_OUT_STAT_LIST クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TMTRL_OUT_STAT_LIST dst, TMTRL_OUT_STAT_LIST src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。

６．６　TMTRL_OUT_STAT - MtrlOutByStatus情報保存クラス

TMTRL_OUT_STAT クラスは、MtrlOutByStatus 情報を保存します。

６．６．１　コンストラクタ

TMTRL_OUT_STAT クラスのインスタンスを生成します。

６．６．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int mtrl_status	状態値
2	public string carid	キャリア ID
3	public int slot_count	キャリアのスロット数
4	public int[] slotid_list	スロットの ID

６．６．３　メソッド

TMTRL_OUT_STAT クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void initial_set()	材料状態とキャリア ID を設定します。
4	public void add_slotid_list()	スロット ID を dlotid_list に追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

６．６．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．６．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．６．３．３　initial_set()　- 材料状態とキャリア ID の設定

当該インスタンスに材料状態とキャリア ID を設定します。

【構文】

public void initial_set(int mstatus, string carid)

【引数】

mstatus

material status

carid

キャリア ID

【戻り値】

なし。

【説明】

当該インスタンスに、材料状態とキャリア ID を設定します。

６．６．３．４　add_slotid_list() - スロット ID の追加

スロット ID を slotid_list に追加します。

【構文】

public void add_slotid_list(int slotid)

【引数】

slotid

スロット ID

【戻り値】

なし。

【説明】

引数 slotid の情報を slotid_list に追加します。

追加した後、プロパティ、slot_count +1 します。

６．６．３．５　copy() - インスタンスのコピー

TMTRL_OUT_STAT クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TMTRL_OUT_STAT dst, TMTRL_OUT_STAT src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。

６．７　TMTRL_OUT_SPEC_LIST - MtrlOutSpec情報リスト保存クラス

TMTRL_OUT_SPEC_LIST クラスは、１個以上の MtrlOutSpec 情報を保存するためのリストです。

TMTRL_OUT_SPEC_LIST リストの要素は、TMTRL_OUT_SPEC クラスのインスタンスです。

６．７．１　コンストラクタ

TMTRL_OUT_SPEC_LIST クラスのインスタンスを生成します。

６．７．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int count	mtrs_list リストに保存されている MtrlOutSpec 情報の数
2	public TMTRL_OUT_SPEC[] mtrs_list	MtrlOutSpec 情報の配列リスト

６．７．３　メソッド

TMTRL_OUT_SPEC_LIST クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public int add()	MtrlOutSpec 情報を mtrs_list に追加します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

６．７．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．７．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．７．３．３　add() - MtrlOutSpec情報の追加

MtrlOutSpec 情報を mtrs_list 配列リストに追加します。

【構文】

public int add(TMTRL_OUT_SPEC info)

【引数】

info

MtrlOutSpec 情報（TMTRL_OUT_SPEC クラスのインスタンス）

【戻り値】

なし。

【説明】

引数 info の情報を mtrs_list リストに追加します。
追加した後、プロパティ、count +1 します。

６．３．４．４　copy() - インスタンスのコピー

TMTRL_OUT_SPEC_LIST クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TMTRL_OUT_SPEC_LIST dst, TMTRL_OUT_SPEC_LIST src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。

６．８　TMTRL_OUT_SPEC - MtrlOutSpec情報保存クラス

TMTRL_OUT_SPEC クラスは、MtrlOutSpec 情報を保存します。

６．８．１　コンストラクタ

TMTRL_OUT_SPEC クラスのインスタンスを生成します。

６．８．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string src_carid	元のキャリア ID
2	public int src_slot_count	元のキャリアのスロット数
3	public int[] src_slotid_list	元のスロット ID リスト
4	public string dst_carid	行先のキャリア ID
5	public int dst_slot_count	行先のキャリアのスロット数
6	public int[] dst_slotid_list	行先のスロット ID リスト

６．８．３　メソッド

TMTRL_OUT_SPEC クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void initial_set()	材料状態とキャリア ID を設定します。
4	public void add_slotid_list()	スロット ID を dlotid_list に追加します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

６．８．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．８．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．８．３．３　initial_set()　- 材料状態とキャリアID の設定

当該インスタンスに材料状態とキャリア ID を設定します。

【構文】

public void initial_set(string srcid, string dstid)

【引数】

src_carid

　　元のキャリア ID

carid

　　行先キャリア ID

【戻り値】

なし。

【説明】

当該インスタンスに、soruce と destination のキャリア ID を設定します。

６．８．３．４　add_src_slotid_list ()- 元のスロットIDの追加

スロット ID を src_slotid_list に追加します。

【構文】

public void add_src_slotid_list(int slotid)

【引数】

slotid

　　スロット ID

【戻り値】

なし。

【説明】

引数 slotid の情報を src_slotid_list に追加します。

追加した後、プロパティ、src_slot_count +1 します。

６．８．３．５　add_dst_slotid_list()- 行先のスロットIDの追加

スロット ID を dst_slotid_list に追加します。

【構文】

public void add_dst_slotid_list(int slotid)

【引数】

slotid

スロット ID

【戻り値】

なし。

【説明】

引数 slotid の情報を dst_slotid_list に追加します。

追加した後、プロパティ、dst_slot_count +1 します。

６．８．３．６　copy()- インスタンスのコピー

TMTRL_OUT_SPEC クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TMTRL_OUT_SPEC dst, TMTRL_OUT_SPEC src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。

６．９　TPAUSE_EVENT - 一時停止イベント情報クラス

TPAUSE_EVENT クラスは一時停止イベント情報を保存するために使用します。

６．９．１　コンストラクタ

TPAUSE_EVENT クラスのインスタンスを生成します。

６．９．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public int ce_count	ceid_list に保存されている CEID) 数
2	public UInt32[] ceid_list	CE イベント ID 保存リスト

６．９．３　メソッド

コントロールジョブ情報クラス TPAUSE_EVENT のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void add_ceid()	テキストを ceid_list に追加します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

６．９．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．９．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．９．３．３　add_ceid_list() - CEID 情報の追加

指定された ID のプロセスジョブが使用する CEID の情報を追加します。

【構文】

public void add_ceid(uint ceid)

【引数】

ceid

追加したい CEID

【戻り値】

なし。

【説明】

ceid_list 配列リストに CEID を 1 個追加します。

６．９．３．４　copy() - インスタンスのコピー

TMTRL_OUT_SPEC クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TPAUSE_EVENT dst, TPAUSE_EVENT src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。

６．10　TCTRL_SPEC - ProcessingCtrlSpec情報保存クラス

TCTRL_SPEC クラスは、ProcessingCtrlSpec 情報を保存します。

６．10．１　コンストラクタ

TCTRL_SPEC クラスのインスタンスを生成します。

６．10．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string prjid	プロセスジョブ ID
2	public int ctrl_rule_count	プロセスジョブのルール数
3	public TCTRL_RULE [] ctrl_rule_list	ルール ID リスト
4	public int out_rule_count	行先のプロセスジョブのルール数
5	public TOUT_RULE [] out_rule_list	行先のルール ID リスト

６．10．３　メソッド

TCTRL_SPEC クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void initial_set()	プロセスジョブ ID を設定します。
4	public int add_ctrl_rule()	TCTRL_RULE 情報を ctrl_rule_list に追加します。
5	public int set_ctrl_rule()	TCTRL_RULE 情報リストを ctrl_rule_list に設定します。
6	public int add_out_rule()	TOUT_RULE 情報を out_rule_list に追加します。
7	public int set_out_rule()	TOUT_RULE 情報リストを out_rule_list に設定します。
8	public static int copy()	インスタンスを別のインスタンスにコピーします。

６．10．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．10．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．10．３．３　initial_set()　–プロセスジョブID の設定

当該インスタンスにプロセスジョブ ID を設定します。

【構文】

public void initial_set(string id)

【引数】

id

プロセスジョブ ID

【戻り値】

なし。

【説明】

当該インスタンスにプロセスジョブ ID を設定します。

６．10．３．４　add_ctrl_rule ()－CTRLルール情報の追加

ルール情報を ctrl_rule_list に追加します。

【構文】

public int add_ctrl_rule(TCTRL_RULE tr_info)

【引数】

tr_info

ルール情報 TCTRL_RULE クラスのインスタンス

戻り値】

返却値	意　味
= 0	追加できた。
(-1)	tr_info が null であった。

【説明】

引数 tr_info 情報を ctrl_rule_list に追加します。

追加した後、プロパティ、ctrl_rule_count +1 します。

６．10．３．５　set_ctrl_rule()-CTRLルール情報リストの設定

TCTRL_RULE リスト情報を ctrl_rule_list に設定します。

【構文】

public int set_ctrl_rule(TCTRL_RULE[] ctrl_rule_list, int size)

【引数】

ctrl_rule_list

TCTRL_RULE クラスのインスタンスの配列リスト

size

ctrl_rule_list 内の要素数

戻り値】

返却値	意　味
= 0	設定できた。
(-1)	ctrl_rule_list が null であった。

【説明】

引数 ctrl_rule_list リスト内の情報を ctrl_rule_list に設定します。

設定した後、プロパティ、ctrl_rule_count　= size　にします。

６．10．３．６　add_rule()-OUTルール情報の追加

ルール情報を out_rule_list に追加します。

【構文】

public int add_out_rule(TOUT_RULE tr_info)

【引数】

tr_info

ルール情報 TOUT_RULE クラスのインスタンス

戻り値】

返却値	意　味
= 0	追加できた。
(-1)	tr_info が null であった。

【説明】

引数 tr_info 情報を out_rule_list に追加します。

追加した後、プロパティ、out_rule_count +1 します。

６．10．３．７　set_out_rule()-CTRLルールリスト情報の設定

TOUT_RULE リスト情報を out_rule_list に設定します。

【構文】

public int set_out_rule(TOUT_RULE[] out_rule_list, int size)

【引数】

out_rule_list

　　TOUT_RULE クラスのインスタンスの配列リスト

size

　　out_rule_list 内の要素数

戻り値】

返却値	意　味
= 0	設定できた。
(-1)	out_rule_list が null であった。

【説明】

引数 out_rule_list リスト内の情報を out_rule_list に設定します。

設定した後、プロパティ、out_rule_count　= size　にします。

６．10．３．８　copy()- インスタンスのコピー

TOUT_SPEC クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TOUT_SPEC dst, TOUT_SPEC src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピーインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。

６．11　TCJ_CMD_INFO - コントロールジョブ・コマンド情報クラス

TCJ_CMD_INFO クラスは、コントロールジョブに対するコマンド情報の保存に使用されます。

本クラスはメッセージ S16F27 の情報を保存するために使用します。

６．11．１　コンストラクタ

TCJ_CMD_INFO クラスのインスタンスを生成します。

６．11．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string ctljobid	コントロールジョブ ID です。
2	public string cmd	コマンド名です。
3	public TCMD_PARA　cmd_info	コマンドパラメータ情報です。

６．11．３　メソッド

コントロールジョブ情報クラス TCJ_CMD_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	コマンド情報を設定します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

６．11．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．11．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．11．３．３　set()　- コントロールジョブコマンド情報の設定

コントロールジョブ ID とコマンドを設定します。

【構文】

public void set(　string id, int cmd)
public void set(string id, int cmd, TCMD_PARA cmd_para)

【引数】

id
　　コントロールジョブ ID
cmd
　　コマンド名です。
cmd_para
　　コマンドパラメータです。

【戻り値】

なし。

【説明】

コントロールジョブ ID とコマンドをプロパティに設定します。
引数に cmd_para が指定されていない場合は、パラメータ無しのコマンドになります。
引数に cmd_para が指定されている場合には、引数 cmd_para の内容をプロパティに設定します。

６．11．３．４　copy() - インスタンスのコピー

TCJ_CMD_INFO のインスタンスを他のインスタンスにコピーします。

【構文】

public static int copy(ref TCJ_CMD_INFO dst_info, TCJ_CMD_INFO src_info)

【引数】

dst
　　コピー先のインスタンス
src
　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。本メソッドは static メソッドです。

６．12　TCMD_PARA - コマンドパラメータ情報クラス

TCMD_PARA クラスは、コントロールジョブ/プロセスジョブのコマンドパラメータ情報を保存するために使用します。

６．12．１　コンストラクタ

TCMD_PARA クラスのインスタンスを生成します。

６．12．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string cpname	コマンドパラメータ名です。
2	public int cpval_fmt	コマンドパラメータ値のフォーマットです。
3	public int cpval_size	コマンドパラメータ値のデータサイズです。
4	public IntPtr cpval	コマンドパラメータ値が保存されているメモリです。

６．12．３　メソッド

コマンドパラメータ TCMD_PARA クラスのメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set_para()	コマンドパラメータ情報を設定します。
4	public void set_para_val()	コマンドパラメータの値を設定します。
5	public static int copy()	インスタンスを別のインスタンスにコピーします。

６．12．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

６．12．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

６．12．３．３　set_para()- コマンドパラメータの設定

コントロールジョブ ID とコマンドを設定します。

【構文】

public int set_para(string name, string para)
public void set_para(string name, int fmt, int size, IntPtr val)

【引数】

name
　　コマンドパラメータ名
para
　　コマンドパラメータ値（文字列データ）です。
fmt
　　コマンドパラメータ値のフォーマットです。
size
　　コマンドパラメータ値のデータサイズです。
val
　　コマンドパラメータ値が保存されているメモリです。

【戻り値】

なし。

【説明】

コマンドパラメータ名とコマンドデータをプロパティに設定します。
2 番目の引数が para の場合には、フォーマット = A、size = para 文字列のバイト長で非管理メモリを確保してプロパティに設定します。

６．12．３．４　set_para_val() - コマンドパラメータ値の設定

コントロールジョブ ID とコマンドを設定します。

【構文】

public int set_para(string para)
public void set_para(int fmt, int size, IntPtr val)

【引数】

para
　　コマンドパラメータ値（文字列データ）です。
fmt
　　コマンドパラメータ値のフォーマットです。
size
　　コマンドパラメータ値のデータサイズです。
val
　　コマンドパラメータ値が保存されているメモリです。

【戻り値】

なし。

【説明】

コマンドパラメータ値だけをプロパティに設定します。
2 番目の引数が para の場合には、フォーマット = A、size = para 文字列のバイト長で非管理メモリを確保してプロパティに設定します。

６．12．３．５　copy() - インスタンスのコピー

TCMD_PARA のインスタンスを他のインスタンスにコピーします。

【構文】

public static int copy(ref TCMD_PARA dst_info, TCMD_PARA src_info)

【引数】

dst
　　コピー先のインスタンス
src
　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。本メソッドは static メソッドです。

7. Carrier - キャリア情報クラス

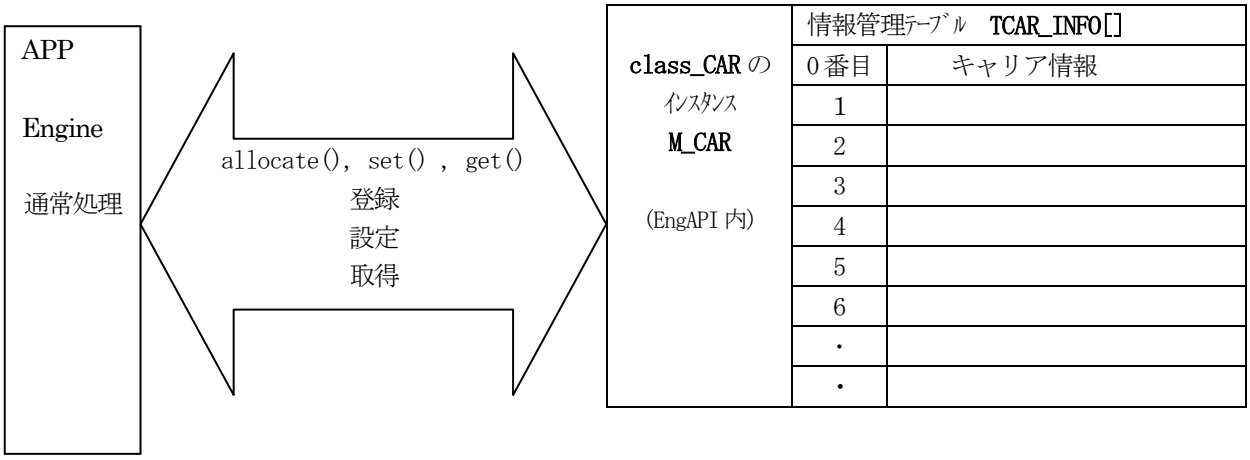
キャリア関連クラスについて説明します。

7. 1 class_CAR - キャリア

class_CAR クラスは全ての**キャリア**の登録、参照、管理サービスを行うためのクラスです。

各キャリア ID 情報の保存には TCAR_INFO クラスを使用します。（7. 2 で説明します）

キャリア情報の構成と参照については概略以下の通りです。



7. 1. 1 コンストラクタ

	名前	説明
1.	public class_CAR(int max_id)	ｲﾝｽﾀﾝｽを生成します。 引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_CAR クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)
引数 max_id は管理する ID の最大数を指定します。プロパティ car_info_tab[] の配列サイズになります。

(本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_CAR を生成します。)

７．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1.	TCAR_INFO[] car_info_tab	キャリア情報の登録テーブル	

TCAR_INFO クラスについては、７．２　TCAR_INFO クラス　の説明を参照ください。

７．１．３　メソッド

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

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int allocate_id()	キャリア ID を登録します。
4	public int set()	指定 ID のキャリア情報を設定します。
5	public int get()	指定 ID のキャリア情報を取得します。
6	public int set_state()	指定 ID の state を設定します。
7	public int get_state()	指定 ID の state を取得します。
8	public int set_capacity()	指定 ID の収納可能スロット数を設定します。
9	public int get_capacity()	指定 ID の収納可能スロット数を取得します。
10	public int set_map_status()	指定 ID の Map Status を設定します。
11	public int get_map_status()	指定 ID の Map Status を取得します。
12	public int set_id_status()	指定 ID の ID Status を設定します。
13	public int get_id_status()	指定 ID の ID Status を取得します。
14	public int set_acc_status()	指定 ID の Access Status を設定します。
15	public int get_acc_status()	指定 ID の Access Status を取得します。

16	public int set_usage()	指定 ID の Usage を設定します。
17	public string get_usage()	指定 ID の Usage を取得します。
18	public int set_location()	指定 ID の Location を設定します。
19	public string get_location()	指定 ID の Location を取得します。
20	public int add_slot()	指定 ID の指定スロットに状態、基板 ID、ロケーションを設定する。
21	public int set_slot_list()	指定 ID のスロットリストに複数のスロット情報を設定します。
22	public int get_slot_list(	指定 ID の複数のスロット情報をスロットリストに取得します。
23	public void delete_all_id()	全登録 ID を削除します。
24	public void delete()	指定 ID の情報を削除します。

例えば、CAR_123 の情報を取得したい場合、次のようなコーディングになります。

```
TCAR_INFO info = new TCAR_INFO();
int result = EngAPI.M_CAR.get (CAR_123, ref info);
```

７．１．３．１　get_id_count() - 登録されている ID 数の取得

DSHEng5 内に登録されているキャリア ID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されているキャリア ID 数を取得します。

７．１．３．２　get_id_list() - 登録されている ID リストの取得

DSHEng5 内に登録されている全キャリア ID をリストに取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大容量

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list リスト内に取得します。

戻り値は、取得した ID 数です。

７．１．３．３　allocate() - CAR ID の登録

指定されたキャリア ID を登録します。

【構文】

```
public int allocate_id(string id)
```

【引数】

id

　　予約したいキャリア ID

【戻り値】

返却値	意　味
0	登録できた。(新規登録)
1	ID が登録済であった。
(-1)	登録できなかった。

【説明】

指定されたキャリア ID を管理下に登録します。
もし、既に登録されていた場合、ID 以外のキャリア情報をクリアします。戻り値=1 を返却します。
登録できるスペースが無かった場合は、(-1)を返却します。

７．１．３．４　set()　- キャリア情報の設定

指定された ID のキャリア情報を設定します。
未登録であれば、allocate します。そして設定します。

【構文】

```
public int set(string id, TCAR_INFO src_info)
```

【引数】

id

　　設定したいキャリア ID

src_info

　　設定したいキャリア情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	src_info が null であった。

【説明】

指定された ID に、src_info で指定されたインスタンスの情報を設定します。

もし、ID が未登録の場合は、最初に allocate()メソッドを使ってエンジン内部で登録します。
既に登録済の場合は、元の情報をクリアした後に設定します。

７．１．３．５　get()－キャリア情報の取得

指定された ID のキャリア情報を取得します。

【構文】

public int get(string id, ref TCAR_INFO dst_info)

【引数】

id

　　取得したいキャリア ID

dst_info

　　取得したキャリア情報を保存するインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定された ID のキャリア情報を dst_info のインスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．６　set_state()－状態の設定

指定された ID のキャリア情報の状態を設定します。

【構文】

public int set_state(string id, int state)

【引数】

id

　　設定したいキャリア ID

state

　　状態

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定された ID のキャリア情報の状態(state)をプロパティ state に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．７　get_state() - 状態の取得

指定されたのキャリア ID の状態(state)を取得します。

【構文】

public int get_state(string id, ref int state)

【引数】

id

　　取得したいキャリア ID

state

　　状態値保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID の状態(state)をプロパティ state から取得し、引数 state に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．８　set_capacity() - 収納数の設定

指定された ID のキャリア情報の収納数を設定します。

【構文】

public int set_capacity(string id, int capacity)

【引数】

id

　　設定したいキャリア ID

capacity

　　収納数

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定された ID のキャリア情報の収納数(capacity)をプロパティ capacity に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．９　get_capacity() - 収納数の取得

指定されたのキャリア ID の収納数(capacity)を取得します。

【構文】

public int get_capacity(string id, ref int capacity)

【引数】

id

　　取得したいキャリア ID

capacity

　　収納数値保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID の収納数(capacity)をプロパティ capacity から取得し、引数 capacity に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．10　set_map_status() - マップ状態の設定

指定されたのキャリア ID のマップ状態を設定します。

【構文】

public int set_map_status(string id, int map_status)

【引数】

id

　　設定したいキャリア ID

map_status

　　マップ状態

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID のマップ状態(map_status)をプロパティ map_status に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．11　get_map_status() - マップ状態の取得

指定されたのキャリア ID のマップ状態(map_status)を取得します。

【構文】

public int get_map_status(string id, ref int map_status)

【引数】

id

　　取得したいキャリア ID

map_status

　　マップ状態値保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID のマップ状態(map_status)をプロパティ map_status から取得し、引数 map_status に設定します。

もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．12　set_id_status() - ID 状態の設定

指定されたのキャリア ID の ID 状態を設定します。

【構文】

public int set_id_status(string id, int id_status)

【引数】

id

　　設定したいキャリア ID

id_status

　　ID 状態

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID の ID 状態(id_status)をプロパティ id_status に設定します。

もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．13　get_id_status() - ID 状態の取得

指定されたのキャリア ID の ID 状態(id_status)を取得します。

【構文】

public int get_id_status(string id, ref int id_status)

【引数】

id

　　取得したいキャリア ID

id_status

　　ID 状態値保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID の ID 状態(id_status)をプロパティ id_status から取得し、引数 id_status に設定します。

もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．14　set_acc_status() - アクセス状態の設定

指定されたのキャリア ID のアクセス状態を設定します。

【構文】

public int set_acc_status(string id, int acc_status)

【引数】

id

　　設定したいキャリア ID

acc_status

　　アクセス状態

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID のアクセス状態(acc_status)をプロパティ acc_status に設定します。

もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．15　get_acc_status() - アクセス状態の取得

指定されたのキャリア ID のアクセス状態(acc_status)を取得します。

【構文】

public int get_acc_status(string id, ref int acc_status)

【引数】

id

　　取得したいキャリア ID

acc_status

　　アクセス状態値保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID のアクセス状態(acc_status)をプロパティ acc_status から取得し、引数 acc_status に設定します。

もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．16　set_usage() - Usage の設定

指定されたのキャリア ID の Usage を設定します。

【構文】

public int set_usage(string id, string usage)

【引数】

id

　　設定したいキャリア ID

usage

　　Usage 名

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID の Usage(usage)をプロパティ usage に設定します。

もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．17　get_usage() - Usage の取得

指定されたのキャリア ID の Usage を取得します。

【構文】

```
public int get_usage(string id, ref string usage)
```

【引数】

id

　　取得したいキャリア ID

usage

　　Usage 値保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID の Usage をプロパティ usage から取得し、引数 usage に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．18　set_location() - Location の設定

指定されたのキャリア ID の Location を設定します。

【構文】

```
public int set_location(string id, string location)
```

【引数】

id

　　設定したいキャリア ID

location

　　Location 名

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID の Location(location)をプロパティ location に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．19　get_location() - Location の取得

指定されたのキャリア ID の Location を取得します。

【構文】

public int get_location(string id, ref string location)

【引数】

id

　　取得したいキャリア ID

location

　　Location 値保存先

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID の Location をプロパティ location から取得し、引数 location に設定します。
もし、ID が未登録の場合は、(-1)を返却します。

７．１．３．20　add_slot()－Slot 情報の追加

指定されたのキャリア ID の Slot 情報をリストに追加します。

【構文】

public int add_slot(string id, int slot_id, int status, string substid, string mid, string s_loc)

【引数】

id

　　設定したいキャリア ID

slot_id

　　スロット ID

status

　　スロットの状態

substid

　　基板 ID

mid

　　材料 ID

s_loc

　　スロットのロケーション

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

引数に指定されたのキャリア ID のプロパティ TCAR_INFO クラスの slot_list リストプロパティに、スロット ID 等の情報を追加します。(TCAR_INFO クラスの 、TSLOT_INFO[] slot_list に追加設定します。)

もし、キャリア ID が未登録の場合は、(-1)を返却します。

７．１．３．21　set_slot_list() - Slot 情報リストの設定

指定されたのキャリア ID の Slot 情報リストにスロットリスト情報を設定します。

【構文】

public int set_slot_list(string id, TSLOT_INFO[] list, int count)

【引数】

id

　　設定したいキャリア ID

list

　　スロット ID 情報リスト

count

　　list に含まれるスロット数

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

引数 list に保存されているスロット情報を指定されたのキャリア ID のプロパティ slot_list に設定します。
(TCAR_INFO クラスの 、TSLOT_INFO[] slot_list に設定します。)
設定するスロット情報の数は引数 count になります。また, プロパティ slot_count = count にします。

もし、キャリア ID が未登録の場合は、(-1)を返却します。

７．１．３．22　get_slot_list() - Slot 情報リストの取得

指定されたのキャリア ID の Slot 情報リストにスロットリスト情報を取得します。

【構文】

public int get_slot_list(string id, ref TSLOT_INFO[] list, ref int count)

【引数】

id

　　取得したいキャリア ID

list

　　スロット ID 情報保存リスト

count

　　list に含まれるスロット数

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

指定されたのキャリア ID のプロパティ slot_list からスロット情報を引数 list に取得します。
取得できたスロット情報の数は引数 count に設定します。

もし、キャリア ID が未登録の場合は、(-1)を返却します。

７．１．３．23　delete() - キャリア情報の削除

指定された ID のキャリア情報を削除し、登録から外します。

【構文】

public void delete(string id)

【引数】

id

削除したいキャリア ID

【戻り値】

なし。

【説明】

指定された ID のキャリア情報を削除します。そして登録から外します。

７．１．３．24　delete_all_id() - 全キャリア情報の消去

登録されているすべてのキャリア情報を削除します。

【構文】

public void delete_all_id()

【引数】

なし。

【戻り値】

なし。

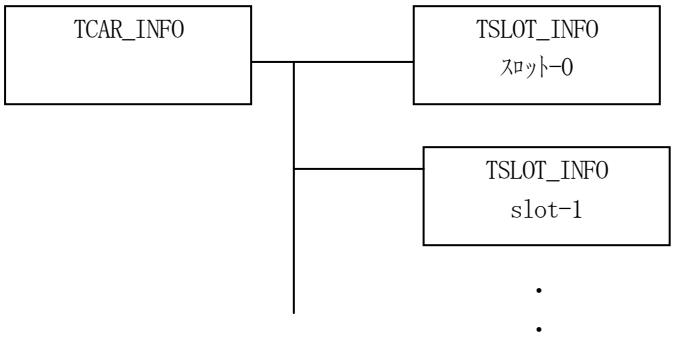
【説明】

登録されているすべてのキャリア情報を削除します。

７．２　TCAR_INFO - キャリア情報保存クラス

TCAR_INFO クラスは、１個のキャリア情報を保存するために使用します。

TCAR_INFO のクラス構成は以下のようになります。



７．２．１　コンストラクタ

TCAR_INFO クラスのインスタンスを生成します。

インスタンス car_info を生成する例です。

（1）空のインスタンスを生成

```
TCAR_INFO car_info = new TCAR_INFO();
```

（2）CAR_123 の ID のインスタンスを生成します。

```
string CAR_123 = “CAR100”;
TCAR_INFO car_info = new TCAR_INFO(CAR_123);
```

（注）CAR_123 の CAR が登録されていない場合は空のインスタンスを生成します。

７．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string carid	キャリア ID
2	public int state	キャリア状態語
3	public int capacity	キャリアの材料収納数
4	public string usage	Usage(用途種類)
5	public int map_status	マップ状態
6	public int id_status	ID 状態
7	public int acc_status	アクセス状態
8	public string location	Location
9	public int slot_count	スロット情報リストのスロット数
10	public TSLOT_INFO[] slot_list	スロット情報リスト

７．２．３　メソッド

キャリア情報クラス TCAR_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void initial_set()	slot 情報以外のプロパティ情報を設定する。
4	public void add_slot()	スロット情報を slot_list[]に追加します。
5	public int add_slot_info()	TSLOT_INFO クラスのインスタンスの内容を slot_listni 追加します。
6	public int set_slot_info()	スロット情報を slot_list[]に設定します。
7	public int set_slot_list()	スロット情報リストを slot_list[]に設定します。
8	public int copy() public static int copy()	インスタンスを別のインスタンスにコピーします。

７．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

７．２．３．２　clear()－プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

７．２．３．３　initial_set()－キャリア情報の初期設定

当該インスタンスにキャリア ID、capacity、usage などの情報を初期設定します。

【構文】

```
public void initial_set(string carid, int capacity , string usage,
                        int map_status, int id_status, int acc_status, string location)
```

【引数】

carid
　　キャリア ID
capacity
　　材料収納数
usage
　　材料の用途
map_status
　　マップ状態
id_status
　　ID 状態
acc_status
　　アクセス状態
location
　　ロケーション

【戻り値】

なし。

【説明】

当該インスタンスにキャリア ID をはじめ、引数で与えられた情報をプロパティに設定します。

７．２．３．４　add_slot() - スロット情報の追加

当該インスタンスの slot_list プロパティにスロット情報を 1 個追加します。

【構文】

public void add_slot(int slot_id, int state, string substid, string mid, string s_loc)

【引数】

slot_id
　　スロット ID
state
　　状態
substid
　　基板 ID
mid
　　材料 ID
id_status
　　ID 状態
s_loc
　　ロケーション

【戻り値】情報を

なし。

【説明】

当該インスタンスのスロット情報リスト slot_list[]に、引数で与えられたスロット情報の TSLOT_INFO クラスのインスタンスを生成し、それを追加設定します。
追加後、slot_count + 1 します。

７．２．３．５　add_slot_info() - スロット情報の追加

当該インスタンスの slot_list プロパティにスロット情報を 1 個追加します。

【構文】

public int add_slot_info(TSLOT_INFO sinfo)

【引数】

sinfo

スロット情報が保存されている TSLOT_INFO クラスのインスタンス

【戻り値】

返却値	意　味
= 0	追加できた。
(-1)	info が null であった。

【説明】

当該インスタンスのスロット情報リスト slot_list[]に、引数で与えられた SLOT_INFO クラスのインスタンスの内容を追加設定します。
追加後、slot_count + 1 します。

７．２．３．６　add_info() - スロット情報の追加

当該インスタンスの slot_list プロパティにスロット情報を 1 個追加します。

【構文】

public int add_slot(TSLOT_INFO sinfo)

【引数】

sinfo

スロット情報が保存されている TSLOT_INFO クラスのインスタンス

【戻り値】

返却値	意　味
= 0	追加できた。
(-1)	info が null であった。

【説明】

当該インスタンスのスロット情報リスト slot_list[]に、引数で与えられた SLOT_INFO クラスのインスタンスの内容を追加します。
追加後、slot_count + 1 します。

７．２．３．７　set_slot_list() - Slot 情報リストの設定

Slot 情報リスト slot_list[] にスロットリスト情報を設定します。

【構文】

public int set_slot_list(TSLOT_INFO[] list, int count)

【引数】

list

　　スロット ID 情報リスト

count

　　list に含まれるスロット数

【戻り値】

返却値	意　味
0	設定できた。
(-1)	指定されたキャリア ID が登録されていなかった。

【説明】

引数 list に保存されているスロット情報をプロパティ slot_list に設定します。
(当該クラスの 、TSLOT_INFO[] slot_list に設定します。)
設定するスロット情報の数は引数 count になります。また, プロパティ slot_count = count にします。

もし、キャリア ID が未登録の場合は、(-1)を返却します。

７．２．３．８　copy() - インスタンスのコピー

TCAR_INFO クラスのインスタンスをコピーします。

【構文】

public int copy(TCAR_INFO src)
public static int copy(ref TCAR_INFO dst, TCAR_INFO src)

【引数】

dst
　　コピー先のインスタンス
src
　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。こちらは static メソッドです。

７．３　TSLOT_INFO - スロット情報保存クラス

TSLOT_INFO クラスは、１個のスロット情報を保存するために使用します。

７．３．１　コンストラクタ

TSLOT_INFO クラスのインスタンスを生成します。

７．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string slotid	スロット ID
2	public int state	スロット状態語
3	public int mid	材料 ID
4	public string substid	基板 ID
5	public string location	Location

７．３．３　メソッド

スロット情報クラス TSLOT_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	slot 情報を設定します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。

７．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

７．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

７．３．３．３_set()　- スロット情報の設定

当該クラスのインスタンスにスロット ID 等の情報を設定します。

【構文】

public void set(int slot_id, int state, string substid, string mid, string s_loc)

【引数】

slotid

スロット ID

state

状態

substid

基板 ID

mid

材料 ID

s_loc

ロケーション

【戻り値】

なし。

【説明】

当該インスタンスにスロット ID をはじめ、引数で与えられた情報をプロパティに設定します。

７．３．３．４　copy()- インスタンスのコピー

TSLOT_INFO クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TSLOT_INFO dst, TSLOT_INFO src)

【引数】

dst

コピー先のインスタンス

src

コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへスロット情報をコピーします。

８．　Substrate - 基板情報クラス

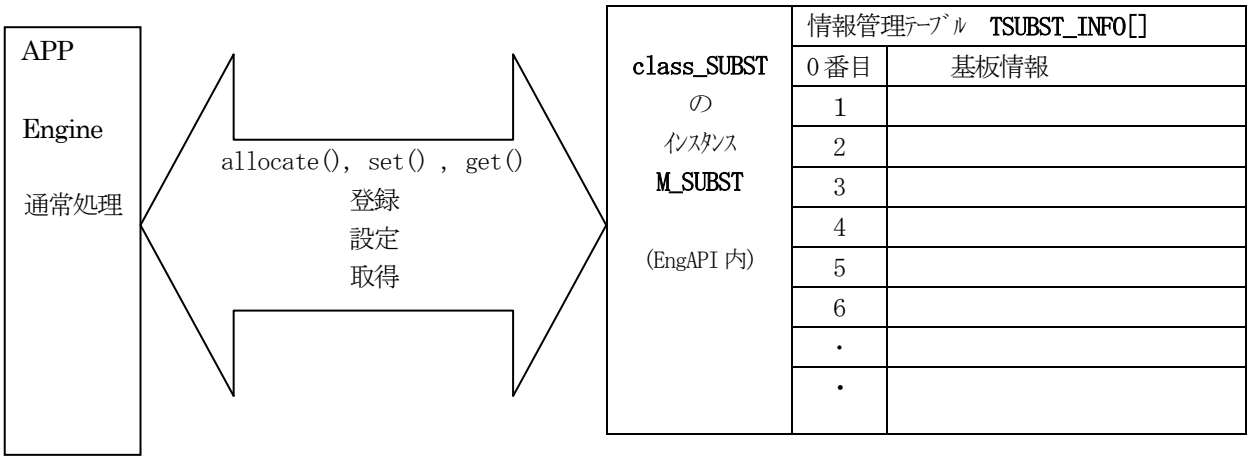
基板関連クラスについて説明します。

８．１　class_SUBST - 基板

class_SUBST クラスは全ての**基板**の登録、参照、管理サービスを行うためのクラスです。

各基板 ID 情報の保存には TSUBST_INFO クラスを使用します。（８．２で説明します）

基板情報の構成と参照については概略以下の通りです。



８．１．１　コンストラクタ

	名前	説明
1.	public class_SUBST(int max_id)	ｲﾝｽﾀﾝｽを生成します。 引数 max_id は ID 最大数 DSHEng5 が開始時に生成します。

class_SUBST クラスのインスタンスを生成します。(DSHEng5 が生成します。APP が生成する必要はありません。)
引数 max_id は管理する ID の最大数を指定します。プロパティ subst_info_tab[] の配列サイズになります。

(本クラスの生成は EngAPI クラスが、APP からの start()メソッドによるエンジン開始時にインスタンス M_SUBST を生成します。)

８．１．２　プロパティ

下表のプロパティを所有しています。

	プロパティ名	説明	ﾃﾞﾌｫﾙﾄ値
1.	TSUBST_INFO[] subst_info_tab	基板情報の登録テーブル	

TSUBST_INFO クラスについては、**８．２　TSUBST_INFO** クラス　の説明を参照ください。

８．１．３　メソッド

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

	メソッド名	説明
1	public int get_id_count()	登録されている ID 数を取得します。
2	public int get_id_list()	登録されている ID リストを取得します。
3	public int allocate_id()	基板 ID を登録します。
4	public int set()	指定 ID の基板情報を設定します。
5	public int get()	指定 ID の基板情報を取得します。
6	public void delete()	指定 ID の情報を削除します。
7	public void delete_all_id()	全登録 ID を削除します。

例えば、SUBST_123 の情報を取得したい場合、次のようなコーディングになります。

```
TSUBST_INFO info = new TSUBST_INFO();
int result = EngAPI.M_SUBST.get (SUBST_123, ref info);
```

８．１．３．１　get_id_count() - 登録されている ID 数の取得

DSHEng5 内に登録されている基板 ID 数を取得します。

【構文】

public int get_id_count()

【引数】

なし。

【戻り値】

返却値	意　味
ID 数	登録 ID 数

【説明】

登録されている基板 ID 数を取得します。

８．１．３．２　get_id_list() - 登録されている ID リストの取得

DSHEng5 内に登録されている全基板 ID をリストに取得します。

【構文】

public int get_id_list(UInt32[] id_list, int max_size)

【引数】

id_list

　　ID を保存するリスト

max_size

　　id_list 配列の最大容量

【戻り値】

返却値	意　味
ID 数	id_list リストに取得した ID 数

【説明】

登録されている ID を id_list リスト内に取得します。

戻り値は、取得した ID 数です。

８．１．３．３　allocate() - SUBST ID の登録

指定された基板 ID を登録します。

【構文】

public int allocate_id(string id)

【引数】

id

予約したい基板 ID

【戻り値】

返却値	意　味
0	登録できた。(新規登録)
1	ID が登録済であった。
(-1)	登録できなかった。

【説明】

指定された基板 ID を管理下に登録します。
もし、既に登録されていた場合、ID 以外の基板情報をクリアします。戻り値=1 を返却します。
登録できるスペースが無かった場合は、(-1)を返却します。

８．１．３．４　set()　- 基板情報の設定

指定された ID の基板情報を設定します。
未登録であれば、allocate します。そして設定します。

【構文】

public int set(string id, TSUBST_INFO src_info)

【引数】

id

設定したい基板 ID

src_info

設定したい基板情報

【戻り値】

返却値	意　味
0	設定できた。
(-1)	src_info が null であった。

【説明】

指定された ID に、src_info で指定されたインスタンスの情報を設定します。

もし、ID が未登録の場合は、最初に allocate()メソッドを使ってエンジン内部で登録します。
既に登録済の場合は、元の情報をクリアした後に設定します。

８．１．３．５　get()－基板情報の取得

指定された ID の基板情報を取得します。

【構文】

public int get(string id, ref TSUBST_INFO dst_info)

【引数】

id

　　取得したい基板 ID

dst_info

　　取得した基板情報を保存するインスタンス

【戻り値】

返却値	意　味
0	取得できた。
(-1)	指定された基板 ID が登録されていなかった。

【説明】

指定された ID の基板情報を dst_info のインスタンスに取得します。
もし、ID が未登録の場合は、(-1)を返却します。

８．１．３．６　delete()－基板情報の削除

指定された ID の基板情報を削除し、登録から外します。

【構文】

public void delete(string id)

【引数】

id

　　削除したい基板 ID

【戻り値】

なし。

【説明】

指定された ID の基板情報を削除します。そして登録から外します。

８．１．３．７　delete_all_id() - 全基板情報の消去

登録されているすべての基板情報を削除します。

【構文】

```
public void delete_all_id()
```

【引数】

なし。

【戻り値】

なし。

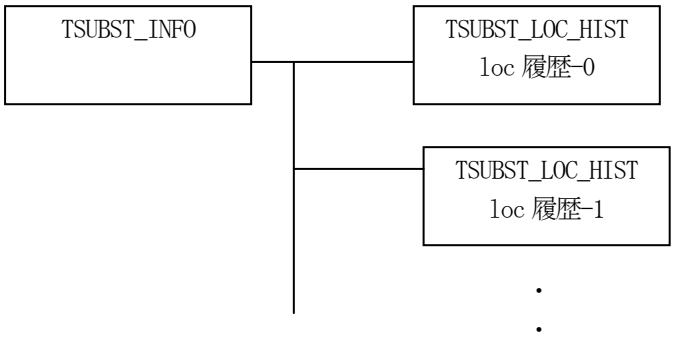
【説明】

登録されているすべての基板情報を削除します。

８．２　TSUBST_INFO - 基板情報保存クラス

TSUBST_INFO クラスは、１個の基板情報を保存するために使用します。

TSUBST_INFO のクラス構成は以下のようになります。



８．２．１　コンストラクタ

TSUBST_INFO クラスのインスタンスを生成します。

インスタンス subst_info を生成する例です。

（1）空のインスタンスを生成

```
TSUBST_INFO subst_info = new TSUBST_INFO();
```

（2）SUBST_123 の ID のインスタンスを生成します。

```
string SUBST_123 = “SUBST100”;
TSUBST_INFO subst_info = new TSUBST_INFO(SUBST_123);
```

（注）SUBST_123 の SUBST が登録されていない場合は空のインスタンスを生成します。

８．２．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string substid	基板 ID
2	public int state	基板状態語
3	public string acquired_id	取得 ID
4	public string lotid	Lot ID
5	public string location;	現在位置
6	public string source	入口位置
7	public string destination	行先位置
8	public string batch_location	バッチロケーション
9	public string pos_in_batch	バッチ内の順位
10	public int id_status	ID 状態
11	public int material_status	材料状態
12	public int proc_state	処理状態
13	public int subst_state	基板状態
14	public int loc_state	ロケーション状態
15	public int usage	用途
16	public int type	タイプ
17	public int hist_count	ロケーション履歴数（hist_list の中の）
18	public TSUBST_LOC_HIST[] hist_list	ロケーション履歴情報保存リスト

８．２．３　メソッド

基板情報クラス TSUBST_INFO のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void clear_hist_list()	hist_list[]プロパティを空にします。
4	public int set()	基板情報を設定します。
5	public int add_TSUBST_LOC_HIST()	ロケーション履歴情報を追加します。
6	public int set_TSUBST_LOC_HIST_list()	ロケーション履歴情報リストを設定します。
7	public int copy() public static int copy()	インスタンスを別のインスタンスにコピーします。

８．２．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

public void Dispose()

【戻り値】

なし。

【説明】

当該インスタンスのプロパテイを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

８．２．３．２　clear()－プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

public void clear()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

８．２．３．３　clear_list()－ロケーション履歴リストのクリア

プロパティ hist_list の内容をすべて消去します。

【構文】

public void clear_hist()

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティ hist_list の内容をすべてクリアします。

８．２．３．４　set()－基板情報の設定

基板情報を設定します。

【構文】

public int set(TSUBST_INFO info, TSUBST_LOC_HIST[] list, int count)

【引数】

info

　　設定したいロケーション履歴情報を除く基板情報

list

　　設定したいロケーション履歴情報リスト

count

　　ロケーション履歴情報の数

【戻り値】

返却値	意　味
0	設定できた。
(-1)	info が null であった。

【説明】

以下の２つの情報を設定します。

①引数 info で指定された TSUBST_INFO のインスタンスの情報を設定します。(substid ほかのプロパティ)

②引数 list で指定されたロケーション履歴リストの内容を hist_list プロパティにコピーします。

８．２．３．５　add_TSUBST_LOC_HIST()－ロケーション履歴情報の追加

ロケーション履歴情報を hist_list リストに追加します。

【構文】

public int add_TSUBST_LOC_HIST(TSUBST_LOC_HIST info)

【引数】

info

　　追加したいロケーション履歴情報

【戻り値】

返却値	意　味
0	追加できた。
(-1)	info が null であった。

【説明】

引数 info のロケーション履歴情報をプロパティ hist_list リストに追加します。

追加した後は、hist_count + 1 します。

８．２．３．６　set_TSUBST_LOC_HIST_list() - ロケーション履歴情報リストの設定

ロケーション履歴リスト情報を hist_list リストに設定します。

【構文】

public int　set_TSUBST_LOC_HIST_list(TSUBST_LOC_HIST[] list, int count)

【引数】

list

　　設定したいロケーション履歴リスト情報

count

　　list に保存されたロケーション履歴の数

【戻り値】

返却値	意　味
0	設定できた。
(-1)	info が null であった。

【説明】

引数 info のロケーション履歴情報をプロパティ hist_list リストに追加します。
追加した後は、hist_count + 1 します。

８．２．３．７　copy() - インスタンスのコピー

TSUBST_INFO クラスのインスタンスをコピーします。

【構文】

public int copy(TSUBST_INFO src)
public static int copy(ref TSUBST_INFO dst, TSUBST_INFO src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

引数 dst が無いメソッドは、src のインスタンスの内容を当該インスタンスにコピーします。

引数 dst があるメソッドは src から dst インスタンスへコピーします。こちらは static メソッドです。

８．３　TSUBST_LOC_HIST - 基板ロケーション履歴情報保存クラス

TSUBST_LOC_HIST クラスは、１個の基板ロケーション履歴情報を保存するために使用します。

８．３．１　コンストラクタ

TSUBST_LOC_HIST クラスのインスタンスを生成します。

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

（2）public TSUBST_LOC_HIST(string locid, strng timein, string timeout)
インスタンスを生成し、引数 locid, timein, timeout をそれぞれのプロパティに設定します。

８．３．２　プロパティ

プロパティを下表に示します。

	プロパティ名	説明
1	public string location	ロケーション名
2	public string time_in	入時刻 - YYYYDDHHMMSSFF - FF : 10ms 単位
3	public string time_out	出時刻 "

８．３．３　メソッド

基板ロケーション履歴情報クラス TSUBST_LOC_HIST のメソッドは下表のとおりです。

	メソッド名	説明
1	public void Dispose()	インスタンスを破棄します。
2	public void clear()	すべてのプロパティの値をクリアします。
3	public void set()	基板ロケーション履歴情報を設定します。
4	public static int copy()	インスタンスを別のインスタンスにコピーします。
5	public static int copy_list()	TSUBST_LOC_HIST クラスの配列リストを別の配列リストのコピーします。

８．３．３．１　Dispose() - インスタンスの破棄

当該インスタンスの内容をすべて消去し、破棄します。

【構文】

```
public void Dispose()
```

【戻り値】

なし。

【説明】

当該インスタンスのプロパティを clear()メソッドによってすべてクリアします。
そして、破棄します。
Dispose()の後、このインスタンスを使用することはできません。

８．３．３．２　clear() - プロパティのクリア

当該インスタンスの内容をすべて消去します。

【構文】

```
public void clear()
```

【引数】

なし。

【戻り値】

なし。

【説明】

当該インスタンスのプロパティをすべてクリアします。

８．３．３．３　set() - ロケーション情報の設定

引数のロケーション履歴情報をプロパティに設定します。

【構文】

public void set(string locid, string t_in, string t_out)

【引数】

locid

　　ロケーション名

t_in

　　入時刻

t_out

　　出時刻

【戻り値】

なし。

【説明】

引数のロケーション履歴情報 locid, t_in, t_out をそれぞれプロパティに location, time_in, time_out に設定します。

８．３．３．４　copy() - TSUBST_LOC_HIST クラスのインスタンスのコピー

TSUBST_LOC_HIST クラスのインスタンスをコピーします。

【構文】

public static int copy(ref TSUBST_LOC_HIST dst, TSUBST_LOC_HIST src)

【引数】

dst

　　コピー先のインスタンス

src

　　コピー元のインスタンス

戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src が null であった。

【説明】

src から dst インスタンスへコピーします。

８．３．３．５　copy list() - TSUBST_LOC_HIST リストのコピー

TSUBST_LOC_HIST 配列リストをコピーします。

【構文】

public static int copy_list(ref TSUBST_LOC_HIST[] dst_list, TSUBST_LOC_HIST[] src_list, int count)

【引数】

dst_list

　　コピー先のリスト

src_list

　　コピー元のリスト

count

　　コピーする配列要素数

【戻り値】

返却値	意　味
= 0	コピーできた。
(-1)	src_list が null であった。

【説明】

src_list リストから count 分だけ、dst_list リストへ TSUBST_LOC_HIST のインスタンスをコピーします。