DSHGEM-LIB　通信エンジンライブラリ(GEM+GEM300)

ソフトウェア・パッケージ

APP インタフェース
ライブラリ関数説明書

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

VOL-９／１５

３．１５　SUBSTRATE 基板情報アクセスサービス関数

２００９年6月

株式会社データマップ

文書番号　DSHGEM-LIB-09-30329-00

[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】

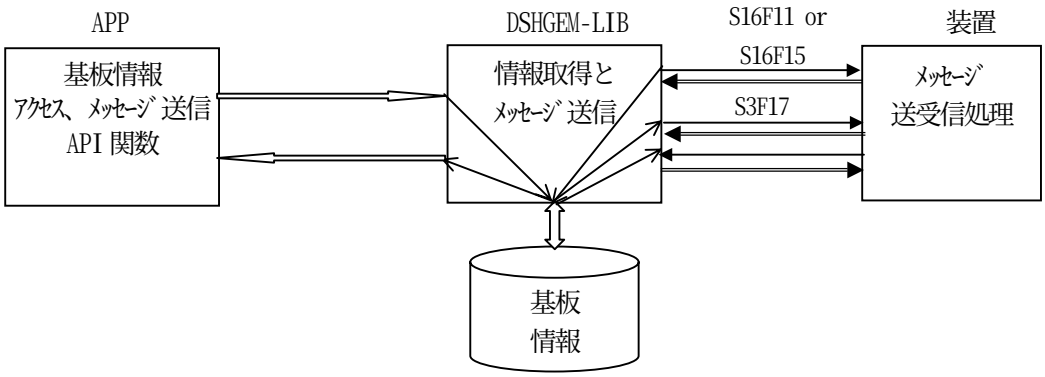
番号	改訂日付	項 目	概 略
1.	2009.6	改訂版	以前の DSHGEM-LIB-07-3032x-00 を全面改訂 .Net VB2008, C#2008 対応関数の説明を追加した。

目　次

(VOL－１０に続く)

３．15　SUBST 基板情報アクセスサービス関数

ここで述べる基板情報は、DSHGEM-LIB が管理します。従って、APP はこれらの情報をアクセスと関連メッセージを送信するために以下の DSHGEM-LIB API 関数を使用します。



（1）情報アクセスと送信 API 関数

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

	API 関数名	機能
1	GemAllocSubstInfo()	基板情報領域を割当て登録します。
2	GemSetSubstInfo()	基板情報を設定・変更します。
3	GemSetSubstInfoX()	基板情報ｲﾝﾃﾞｸｽ指定で基板情報を設定します。
4	GemGetSubstInfo()	SUBSTID 指定で基板情報を取得します。
5	GemGetSubstInfoX()	基板情報ｲﾝﾃﾞｸｽ指定で基板情報を取得します。
6	GemDelSubstInfo()	SUBSTID 指定で基板情報を削除します。
7	GemDelSubstInfoX()	基板情報ｲﾝﾃﾞｸｽ指定で基板情報を削除します。
8	GemSetSubstInfoState()	SUBSTID 指定で SUBST 情報の状態を設定します。
9	GemSetSubstInfoStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST 情報の状態を設定します。
10	GemGetSubstInfoState ()	SUBSTID 指定で SUBST 情報の状態を取得します。
11	GemGetSubstInfoStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST 情報の状態を取得します。
12	GemSetSubstIdState()	SUBSTID 指定で SUBST ID の状態を設定します。
13	GemSetSubstIdStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST ID の状態を設定します。
14	GemGetSubstIdState ()	SUBSTID 指定で SUBSTID の状態を取得します。
15	GemGetSubstIdStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST ID の状態を取得します。
16	GemSetSubstAcquiredID()	SUBSTID 指定で基板読取 ID を設定します。
17	GemSetSubstAcquiredIDX()	基板情報ｲﾝﾃﾞｸｽ指定で基板読取 ID を設定します。
18	GemGetSubstAcquiredID()	SUBSTID 指定で基板読取 ID を取得します。
19	GemGetSubstAcquiredIDX()	基板情報ｲﾝﾃﾞｸｽ指定で基板読取 ID を取得します。
20	GemSetSubstLotId()	SUBSTID 指定で基板の LotId を設定します。
21	GemSetSubstLotIdX()	基板情報ｲﾝﾃﾞｸｽ指定で基板の LotId を設定します。
22	GemGetSubstLotId()	SUBSTID 指定で基板の LotId を取得します。
23	GemGetSubstLotIdX()	基板情報ｲﾝﾃﾞｸｽ指定で基板の LotId を取得します。
24	GemSetSubstLocId()	SUBSTID 指定で基板の AcquiredIdId を設定します。
25	GemSetSubstLocIdX()	基板情報ｲﾝﾃﾞｸｽ指定で基板の AcquiredIdId を設定します。
26	GemGetSubstLocId()	SUBSTID 指定で基板の AcquiredIdId を取得します。
27	GemGetSubstLocIdX()	基板情報ｲﾝﾃﾞｸｽ指定で基板の AcquiredIdId を取得します。
28	GemSetSubstSource()	SUBSTID 指定で基板の移動元ﾛｹｰｼｮﾝを設定します。

29	GemSetSubstSourceX()	基板情報ｲﾝﾃﾞｸｽ指定で基板の移動元ﾛｹｰｼｮﾝを設定します。
30	GemGetSubstSource()	SUBSTID 指定で基板の移動元ﾛｹｰｼｮﾝを取得します。
31	GemGetSubstSourceX()	基板情報ｲﾝﾃﾞｸｽ指定で基板の移動元ﾛｹｰｼｮﾝを取得します。
32	GemSetSubstDestination()	SUBSTID 指定で基板の移動先ﾛｹｰｼｮﾝを設定します。
33	GemSetSubstDestinationX()	基板情報ｲﾝﾃﾞｸｽ指定で基板の移動先ﾛｹｰｼｮﾝを設定します。
34	GemGetSubstDestination()	SUBSTID 指定で基板の移動先ﾛｹｰｼｮﾝを取得します。
35	GemGetSubstDestinationX()	基板情報ｲﾝﾃﾞｸｽ指定で基板の移動先ﾛｹｰｼｮﾝを取得します。
36	GemSetSubstBatchLocId()	SUBSTID 指定で基板のﾊﾞｯﾁﾛｹｰｼｮﾝを設定します。
37	GemSetSubstBatchLocIdX()	基板情報ｲﾝﾃﾞｸｽ指定で基板のﾊﾞｯﾁﾛｹｰｼｮﾝを設定します。
38	GemGetSubstBatchLocId ()	SUBSTID 指定で基板のﾊﾞｯﾁﾛｹｰｼｮﾝを取得します。
39	GemGetSubstBatchLocIdX()	基板情報ｲﾝﾃﾞｸｽ指定で基板のﾊﾞｯﾁﾛｹｰｼｮﾝを取得します。
40	GemSetSubstPosInBatch()	SUBSTID 指定で基板のﾊﾞｯﾁ内位置を設定します。
41	GemSetSubstPosInBatchX()	基板情報ｲﾝﾃﾞｸｽ指定で基板のﾊﾞｯﾁ内位置を設定します。
42	GemGetSubstPosInBatch ()	SUBSTID 指定で基板のﾊﾞｯﾁ内位置を取得します。
43	GemGetSubstPosInBatchX()	基板情報ｲﾝﾃﾞｸｽ指定で基板のﾊﾞｯﾁ内位置を取得します。
44	GemSetSubstState()	SUBSTID 指定で SUBST の状態を設定します。
45	GemSetSubstStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST の状態を設定します。
46	GemGetSubstState()	SUBSTID 指定で SUBST の状態を取得します。
47	GemGetSubstStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST の状態を取得します。
48	GemSetSubstProcState()	SUBSTID 指定で SUBST の処理状態を設定します。
49	GemSetSubstProcStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST の処理状態を設定します。
50	GemGetSubstProcState()	SUBSTID 指定で SUBST の処理状態を取得します。
51	GemGetSubstProcStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST の処理状態を取得します。
52	GemGetSubstStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST の状態を取得します。
53	GemSetSubstLocState()	SUBSTID 指定で SUBST のﾛｹｰｼｮﾝ状態を設定します。
54	GemSetSubstLocStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST のﾛｹｰｼｮﾝ状態を設定します。
55	GemGetSubstLocState()	SUBSTID 指定で SUBST のﾛｹｰｼｮﾝ状態を取得します。
56	GemGetSubstLocStateX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBST のﾛｹｰｼｮﾝ状態を取得します。
57	GemSetSubstIdStatus()	SUBSTID 指定で SUBSTID の状態を設定します。
58	GemSetSubstIdStatusX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBSTID の状態を設定します。
59	GemGetSubstIdStatus()	SUBSTID 指定で SUBSTID の状態を取得します。
60	GemGetSubstIdStatusX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBSTID の状態を取得します。
61	GemSetSubstUsage()	SUBSTID 指定で SUBSTID の使用法を設定します。
62	GemSetSubstUsageX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBSTID の使用法を設定します。
63	GemGetSubstUsage()	SUBSTID 指定で SUBSTID の使用法を取得します。
64	GemGetSubstUsageX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBSTID の使用法を取得します。
65	GemSetSubstType()	SUBSTID 指定で SUBSTID のﾀｲﾌﾟを設定します。
66	GemSetSubstTypeX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBSTID のﾀｲﾌﾟを設定します。
67	GemGetSubstType()	SUBSTID 指定で SUBSTID のﾀｲﾌﾟを取得します。
68	GemGetSubstTypeX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBSTID のﾀｲﾌﾟを取得します。
69	GemSetSubstLocHistory()	SUBSTID 指定で SUBSTID のﾛｹｰｼｮﾝ履歴を設定します。
70	GemSetSubstLocHistoryX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBSTID のﾛｹｰｼｮﾝ履歴を設定します。
71	GemGetSubstLocHistory()	SUBSTID 指定で SUBSTID のﾛｹｰｼｮﾝ履歴を取得します。
72	GemGetSubstTLocHistoryX()	基板情報ｲﾝﾃﾞｸｽ指定で SUBSTID のﾛｹｰｼｮﾝ履歴を取得します。
73	GemGetSubstList()	基板 ID の一覧ﾘｽﾄを取得します。
74	GemGetSubstId()	指定した基板情報ｲﾝﾃﾞｸｽの SUBSTID を取得します。

75	GemGetSubstIdIndex()	指定した SUBSTID の情報ｲﾝﾃﾞｸｽを取得します。

SUBSTID インデクスは、DSHGEM-LIB が管理する各 SUBSTID 領域の番号です。このインデクスの値は、GemAllocSubstInfo()関数実行時に DSHGEM-LIB によって割当てられ、APP に渡されます。また、基板の取得時に、情報格納構造体のメンバー、 index に設定されます。

（2）ライブラリ関数

他に APP が使用できる基板情報処理用 API 関数として、以下の関数があります。

	API 関数名	機能
1	DshFreeTSUBST_INFO()	基板情報が格納されている TSUBST_INFO 構造体の内部で使用されているﾒﾓﾘを開放するための関数です。
2	DshCopyTSUBST_INFO()	TSUBST_INFO の基板情報を別の別の構造体にｺﾋﾟｰします。
3	DshFreeTSUBST_LOC_HIST()	TSUBST_LOC_HIST 構造体内のﾒﾓﾘを開放します。
4	DshCopyTSUBST_LOC_HIST()	TSUBST_LOC_HIST 構造体の内容を別の構造体にｺﾋﾟｰします。
5	DshInitSubstInfo()	TSUBST_INFO 構造体を初期設定します。
6	DshPutSubstLocHist()	TSUBST_INFO 構造体内に基板ﾛｹｰｼｮﾝ移動情報を１個追加設定します。
7	DshInitSubstLocHist()	TSUBST_INFO 内に設定するための履歴情報構造体 TSUBST_LOC_HIST を初期設定します。
8	DshAddSubstLocHist()	TSUBST_LOC_HIST 構造体内に履歴情報を１個追加します。

3．15．1　使用する情報格納構造体

基板情報を操作する関数は、共通の情報格納のための TSUBST_INFO 構造体を使用します。
基板情報に関連する構造体は下記のとおりです。

（1）TSUBST_INFO - Substrate Information

```
typedef struct{
    int         index;
    int         state;                  // 情報の状態です。
    char        *substid;               // ID
    char        *acquiredID;
    char        *lotID;
    char        *subst_locID;
    char        *subst_source;
    char        *subst_destination;
    char        *batch_locID;
    char        *subst_pos_in_batch;
    int         substIDstatus;
    int         material_status;
    int         subst_proc_state;
    int         subst_state;
    int         subst_loc_state;
    int         subst_usage;
    int         subst_type;
   TSUBST_LOC_HIST *hist;
  } TSUBST_INFO;
```

（2）TSLOT_LOC_HIST - Substrate Location History

```
typedef struct{
    int         hist_count;
    char        **subst_locID;
    char        **TimeIn;
    char        **TimeOut;
} TSUBST_LOC_HIST;
```

3．15．２　SUBST 基板情報アクセス関数

3．15．２．１　GemAllocSubstInfo() - 基板の登録

（1）呼出書式

[c,C++]

```
API int APIX GemAllocSubstInfo(
        int   eqid,                 // 通信対象装置 ID(0,1,2,...)
        char *substid,              // 基板 ID 格納領域のﾎﾟｲﾝﾀ
        int   *index                // 得られた情報領域ｲﾝﾃﾞｸｽ格納用ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemAllocSubstInfo (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef index As Int32) As Int32
```

[.NET C#]

```
int GemAllocSubstInfo(
        int eqid,
        byte[] substid,
        ref int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

登録したい基板 ID が格納されているポインタです。

index

登録された情報領域のインデクス値が格納される領域のポインタです。

（3）戻り値

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

（4）説明

基板を新規にシステムに登録するための関数です。
登録は、引数 substid で与えられる基板 ID をシステムに登録します。

正常に登録できた場合は、index で指定される領域に登録された情報領域のインデクスが設定返却されます。もし、substid に指定された基板が既に登録済みであった場合には関数の戻り値 1 を返却します。index には既に登録されている情報領域のインデクスが設定されます。

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

３．15．２．２　GemSetSubstInfo()　- 基板情報の設定
.　　GemSetSubstInfoX() – インデクスでの基板情報の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetSubstInfo(
        int          eqid,              // 通信対象装置 ID(0,1,2,...)
        TSUBST_INFO *pinfo              // 基板情報格納構造体のﾎﾟｲﾝﾀ
);

API int APIX GemSetSubstInfoX(
        int          eqid,              // 通信対象装置 ID(0,1,2,...)
        int          index,             // GemAllocSubstInfo()で得られた基板情報のインデクス
        TSUBST_INFO *pinfo              // 基板情報格納構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemSetSubstInfo (
        ByVal eqid As Int32,
        ByRef pinfo As dsh_info.TSUBST_INFO) As Int32

Function GemSetSubstInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef pinfo As dsh_info.TSUBST_INFO) As Int32
```

[.NET C#]

```
int GemSetSubstInfo(
        int eqid,
        ref TSUBST_INFO pinfo );

int GemSetSubstInfoX(
        int eqid,
        int index,
        ref TSUBST_INFO pinfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

pinfo

設定したい基板情報が格納されている格納構造体領域のポインタです。

index

基板 ID 情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。
インデクスは、SUBSTID から GemGetSubstIdIndex()関数で取得することができます。

（3）戻り値

戻り値	意味
0	正常に設定できた。

(-1)	設定できなかった。

（4）説明

本関数は、pinfo に格納されている基板情報の設定・変更に使用します。

引数 pinfo 内のメンバーsubstid に指定される基板 ID の情報として設定されます。

pinfo 内には SUBSTID の他、 スロット情報などの情報が含まれます。
指定した SUBSTID が既に登録済みである場合には、pinfo 内の情報にすべて書き換えられます。

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

３．15．２．３　GemGetSubstInfo() - 基板情報の取得
.　　GemGetSubstInfoX() - インデクス指定での基板情報の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetSubstInfo(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        char        *substid,               // SUBSTID が格納されている領域のﾎﾟｲﾝﾀ
        TSUBST_INFO *pinfo                  // 基板情報を格納する構造体ﾎﾟｲﾝﾀを格納するﾎﾟｲﾝﾀ
);

API int APIX GemGetSubstInfoX(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        int         index,                  // 基板情報のｲﾝﾃﾞｸｽ
        TSUBST_INFO *pinfo                  // 基板情報を格納するﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetSubstInfo (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef pinfo As dsh_info.TSUBST_INFO) As Int32

Function GemGetSubstInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef pinfo As dsh_info.TSUBST_INFO) As Int32
```

[.NET C#]

```
int GemGetSubstInfo(
        int eqid,
        byte[] substid,
        ref TSUBST_INFO pinfo );

int GemGetSubstInfoX(
        int eqid,
        int index,
        ref TSUBST_INFO pinfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

pinfo

取得した基板情報を格納する構造体領域ポインタです。

index

基板 ID 情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

substid または index に指定されている基板の情報を pinfo 構造体領域に格納します。

TSUBST_INFO 構造体の中に情報を格納するために必要なメモリは、DSHGEM-LIB が準備確保します。即ち、構造体のメンバーの中でポインタになっている情報の実体即ち、substid, スロット情報などのためのメモリは DSHGEM-LIB が準備します。
これらのメモリは、使用後、ユーザが DSHGEM-LIB の API 関数を使って次のように開放してください。

```
TSUBST_INFO          *pinfo;

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

３．15．２．４　GemDelSubstInfo()　- 基板の削除
.　　GemDelSubstInfoX() – インデクスでの基板の削除

（1）呼出書式

[c,C++]

```
API int APIX GemDelSubstInfo(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        char        *substid;               // SUBSTID が格納されている領域のポインタ
);

API int APIX GemDelSubstInfoX(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        int         index,                  // ｲﾝﾃﾞｸｽ(0,1,2....)
);
```

[.NET VB]

```
Function GemDelSubstInfo (
        ByVal eqid As Int32,
        ByVal substid As String) As Int32

Function GemDelSubstInfoX (
        ByVal eqid As Int32,
        ByVal index As Int32) As Int32
```

[.NET C#]

```
int GemDelSubstInfo(
        int eqid,
        byte[] substid );

int GemDelSubstInfoX(
        int eqid,
        int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

index

削除したい基板情報のインデクスです。

（3）戻り値

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

（4）説明

substid または index に指定されている基板 ID の情報を登録から削除します。

３．15．２．５　GemSetSubsInfoState() － 基板情報状態の設定
.　　GemSetSubstInfoStateX() － インデクスでの基板情報状態の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetSubstInfoState(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       state                 // 設定したい状態値
);

API int APIX GemSetSubstInfosStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       state                 // 設定したい状態値
);
```

[.NET VB]

```
Function GemSetSubstInfoState (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal state As Int32) As Int32

Function GemSetSubstInfoStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal state As Int32) As Int32
```

[.NET C#]

```
int GemSetSubstInfoState(
        int eqid,
        byte[] substid,
        int state );

int GemSetSubstInfoStateX(
        int eqid,
        int index,
        int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

state

設定したい SUBSTID 情報の状態値です。
値=(-1)は使用不可を意味する値になります。

index

基板情報のインデクスです。登録時にGemAllocSubstInfo()関数によって与えられます。
インデクスは、SUBSTIDからGemGetSubstIdIndex()関数で取得することができます。

（3）戻り値

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

（4）説明

基板情報の状態値を設定します。
状態値の意味合いは値=(-1)以外についてはユーザが定義し、使用します。

３．15．２．６　GemGetSubstInfoState() － 基板情報状態の取得
.　　GemGetSubstInfoStateX() － インデクスでの基板情報状態の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetSubstInfoState(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       *state                // 取得した状態値格納用領域ポインタ
);

API int APIX GemGetSubstInfoStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       *state                // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetSubstInfoState (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef state As Int32) As Int32

Function GemGetSubstInfoStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef state As Int32) As Int32
```

[.NET C#]

```
int GemGetSubstInfoState(
        int eqid,
        byte[] substid,
        ref int state );

int GemGetSubstInfoStateX(
        int eqid,
        int index,
        ref int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

state

取得した SUBSTID 情報の状態値を格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板情報の状態値を取得します。

状態値の意味合いは値=(-1)以外についてはユーザが定義し、使用します。

３．15．２．７　GemSetSubstIdStatus() － 基板 ID 読取り状態の設定
.　　GemSetSubstIdStatusX() － インデクスでの基板 ID 読取り状態の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetSubstIdStatus(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       status                // 設定したい状態値
);

API int APIX GemSetSubstIdStatusX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       status                // 設定したい状態値
);
```

[.NET VB]

```
Function GemSetSubstIdStatus (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal status As Int32) As Int32

Function GemSetSubstIdStatusX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal status As Int32) As Int32
```

[.NET C#]

```
int GemSetSubstIdStatus(
        int eqid,
        byte[] substid,
        int status );

int GemSetSubstIdStatusX(
        int eqid,
        int index,
        int status );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

status

設定したい SUBSTID の ID 読取り状態値です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板 ID 読取り状態値を設定します。
デフォルトの値は次のとおりです。

基板 ID 状態のデフォルト値

状態値記号	値
ST_SubstIdNotRead	0
ST_SubstIdWaitingForHost	1
ST_SubstIdVerificationOK	2
ST_SubstIdVerificationFail	3
ST_SubstIdNotExist	4

3．15．2．8　GemGetSubstIdStatus() – 基板 ID 読取り状態の取得
.　　GemGetSubstIdStatusX() – インデクスでの基板 ID 読取り状態の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetSubstIdStatus(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       *status               // 取得した状態値格納用領域ポインタ
);

API int APIX GemGetSubstIdStatusX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       *status               // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetSubstIdStatus (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef status As Int32) As Int32

Function GemGetSubstIdStatusX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef status As Int32) As Int32
```

[.NET C#]

```
int GemGetSubstIdStatus(
        int eqid,
        byte[] substid,
        ref int status );

int GemGetSubstIdStatusX(
        int eqid,
        int index,
        ref int status );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

status

取得した SUBSTID の ID 読取り状態値を格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板 ID 読取りの状態値を取得します。
デフォルトの値は次のとおりです。

基板 ID 状態のデフォルト値

状態値記号	値
ST_SubstIdNotRead	0
ST_SubstIdWaitingForHost	1
ST_SubstIdVerificationOK	2
ST_SubstIdVerificationFail	3
ST_SubstIdNotExist	4

3．15．２．９　GemSetSubstAcquiredId() － 読取られた ID の設定
.　　GemSetSubstAcquiredIdX() － インデクスでの読取られた ID の設定

（1）呼出書式

[c,C++]

```
API int    APIX GemSetSubstAcquiredId(
        int        eqid,                    // 通信対象装置 ID(0,1,2,...)
        char       *substid,                // SUBSTID が格納されている領域のポインタ
        char       *acquired_id             // 設定したい読取られた SUBSTID 格納ポインタ
);

API int    APIX GemSetSubstAcquiredIdX(
        int        eqid,                    // 通信対象装置 ID(0,1,2,...)
        int        index,                   // 基板情報のインデクス
        char       *acquired_id             // 設定したい読取られた SUBSTID 格納ポインタ
);
```

[.NET VB]

```
Function GemSetSubstAcquiredId (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal acqid As String) As Int32

Function GemSetSubstAcquiredIdX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal acqid As String) As Int32
```

[.NET C#]

```
int GemSetSubstAcquiredId(
        int eqid,
        byte[] substid,
        byte[] acqid );

int GemSetSubstAcquiredIdX(
        int eqid,
        int index,
        byte[] acqid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

acquired_id

設定したい読取られた SUBSTID 値（文字列）です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

読取られた SUBSTID(文字列)を設定します。

3．15．2．10 GemGetSubstAcquiredId() – 読取られた ID の取得
. GemGetSubstAcquiredIdX() – インデクスでの読取られた ID の取得

（1）呼出書式

[c,C++]

```
API int    APIX GemGetSubstAcquiredId(
        int        eqid,                    // 通信対象装置 ID(0,1,2,...)
        char       *substid,                // SUBSTID が格納されている領域のポインタ
        char       *acquired_id             // 取得した読取られた SUBSTID 格納ポインタ
);

API int    APIX GemGetSubstAcquiredIdX(
        int        eqid,                    // 通信対象装置 ID(0,1,2,...)
        int        index,                   // 基板情報のインデクス
        char       *acquired_id             // 取得した読取られた SUBSTID 格納ポインタ
);
```

[.NET VB]

```
Function GemGetSubstAcquiredId (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal acqid As String) As Int32

Function GemGetSubstAcquiredIdX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal acqid As String) As Int32
```

[.NET C#]

```
int GemGetSubstAcquiredId(
        int eqid,
        byte[] substid,
        byte[] acqid );

int GemGetSubstAcquiredIdX(
        int eqid,
        int index,
        byte[] acqid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

acquired_id

取得した "読取られた SUBSTID 値"を格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板の読取られた SUBSTID(文字列)を取得します。

3．15．2．11 GemSetSubstLotId() - LotId の設定
. GemSetSubstLotIdX() - インデクスでの LotId の設定

（1）呼出書式

[c,C++]

```
API int    APIX GemSetSubstLotId(
        int       eqid,                  // 通信対象装置 ID(0,1,2,...)
        char      *substid,              // SUBSTID が格納されている領域のポインタ
        char      *lotid                 // 設定したい lotid 格納ポインタ
);

API int    APIX GemSetSubstLotIdX(
        int       eqid,                  // 通信対象装置 ID(0,1,2,...)
        int       index,                 // 基板情報のインデクス
        char      *lotid                 // 設定したい lotid 格納ポインタ
);
```

[.NET VB]

```
Function GemSetSubstLotId (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal lotid As String) As Int32

Function GemSetSubstLotIdX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal lotid As String) As Int32
```

[.NET C#]

```
int GemSetSubstLotId(
        int eqid,
        byte[] substid,
        byte[] lotid );

int GemSetSubstLotIdX(
        int eqid,
        int index,
        byte[] lotid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

lotid

設定したい SUBSTID の lotid 値（文字列）です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 ID の lotid(文字列)を設定します。

3．15．2．12 GemGetSubstLotId() - LotId の取得
. GemGetSubstLotIdX() - インデクスでの LotId の取得

（1）呼出書式

[c,C++]

```
API int    APIX GemGetSubstLotId(
        int        eqid,                  // 通信対象装置 ID(0,1,2,...)
        char       *substid,              // SUBSTID が格納されている領域のポインタ
        char       *lotid                 // 取得した lotid 格納ポインタ
);

API int    APIX GemGetSubstLotIdX(
        int        eqid,                  // 通信対象装置 ID(0,1,2,...)
        int        index,                 // 基板情報のインデクス
        char       *lotid                 // 取得した lotid 格納ポインタ
);
```

[.NET VB]

```
Function GemGetSubstLotId (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal lotid As String) As Int32

Function GemGetSubstLotIdX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal lotid As String) As Int32
```

[.NET C#]

```
int GemGetSubstLotId(
        int eqid,
        byte[] substid,
        byte[] lotid );

int GemGetSubstLotIdX(
        int eqid,
        int index,
        byte[] lotid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

lotid

取得した SUBSTID の lotid 値を格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 Id の lotid(文字列)を取得します。

3．15．2．13 GemSetSubstLocId() - LocId の設定
. GemSetSubstLocIdX() - インデクスでの LocId の設定

（1）呼出書式

[c,C++]

```
API int    APIX GemSetSubstLocId(
        int       eqid,                  // 通信対象装置 ID(0,1,2,...)
        char      *substid,              // SUBSTID が格納されている領域のポインタ
        char      *location              // 設定したい location 格納ポインタ
);

API int    APIX GemSetSubstLocIdX(
        int       eqid,                  // 通信対象装置 ID(0,1,2,...)
        int       index,                 // 基板情報のインデクス
        char      *location              // 設定したい location 格納ポインタ
);
```

[.NET VB]

```
Function GemSetSubstLocId (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal location As String) As Int32

Function GemSetSubstLocIdX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal location As String) As Int32
```

[.NET C#]

```
int GemSetSubstLocId(
        int eqid,
        byte[] substid,
        byte[] location );

int GemSetSubstLocIdX(
        int eqid,
        int index,
        byte[] location );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

location

設定したい SUBSTID の location（文字列）です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 ID の location(文字列)を設定します。

3．15．2．14 GemGetSubstLocId() - LocId の取得
. GemGetSubstLocIdX() - インデクスでの LocId の取得

（1）呼出書式

[c,C++]

```
API int    APIX GemGetSubstLocId(
        int       eqid,                  // 通信対象装置 ID(0,1,2,...)
        char      *substid,              // SUBSTID が格納されている領域のポインタ
        char      *location              // 取得した location 格納ポインタ
);

API int    APIX GemGetSubstLocIdX(
        int       eqid,                  // 通信対象装置 ID(0,1,2,...)
        int       index,                 // 基板情報のインデクス
        char      *location              // 取得した location 格納ポインタ
);
```

[.NET VB]

```
Function GemGetSubstLocId (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal location As String) As Int32

Function GemGetSubstLocIdX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal location As String) As Int32
```

[.NET C#]

```
int GemGetSubstLocId(
        int eqid,
        byte[] substid,
        byte[] location );

int GemGetSubstLocIdX(
        int eqid,
        int index,
        byte[] location );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

location

取得した SUBSTID の location を格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 Id の location(文字列)を取得します。

3．15．２．15 GemSetSubstSource() － 移動元ロケーションの設定
. GemSetSubstSourceX() － インデクスでの移動元ロケーションの設定

（1）呼出書式

[c,C++]

```
API int    APIX GemSetSubstSource(
        int        eqid,                  // 通信対象装置 ID(0,1,2,...)
        char       *substid,              // SUBSTID が格納されている領域のポインタ
        char       *source                // 設定したい移動元ロケーション格納ポインタ
);

API int    APIX GemSetSubstSourceX(
        int        eqid,                  // 通信対象装置 ID(0,1,2,...)
        int        index,                 // 基板情報のインデクス
        char       *source                // 設定したい移動元ロケーション格納ポインタ
);
```

[.NET VB]

```
Function GemSetSubstSource (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal source As String) As Int32

Function GemSetSubstSourceX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal source As String) As Int32
```

[.NET C#]

```
int GemSetSubstSource(
        int eqid,
        byte[] substid,
        byte[] source );

int GemSetSubstSourceX(
        int eqid,
        int index,
        byte[] source );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

source

設定したい SUBSTID の移動元ロケーション（文字列）です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 ID の移動元ロケーション(文字列)を設定します。

3．15．2．16 GemGetSubstSource() –移動元ロケーションの取得
. GemGetSubstSourceX() – インデクスでの移動元ロケーションの取得

（1）呼出書式

[c,C++]

```
API int    APIX GemGetSubstSource(
        int        eqid,                  // 通信対象装置 ID(0,1,2,...)
        char       *substid,              // SUBSTID が格納されている領域のポインタ
        char       *source                // 取得した移動元ロケーション格納ポインタ
);

API int    APIX GemGetSubstSourceX(
        int        eqid,                  // 通信対象装置 ID(0,1,2,...)
        int        index,                 // 基板情報のインデクス
        char       *source                // 取得した移動元ロケーション格納ポインタ
);
```

[.NET VB]

```
Function GemGetSubstSource (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal source As String) As Int32

Function GemGetSubstSourceX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal source As String) As Int32
```

[.NET C#]

```
int GemGetSubstSource(
        int eqid,
        byte[] substid,
        byte[] source );

int GemGetSubstSourceX(
        int eqid,
        int index,
        byte[] source );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

source

取得した SUBSTID の移動元ロケーションを格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 Id の移動元ロケーション(文字列)を取得します。

3．15．2．17 GemSetSubstDestination() － 移動先ロケーションの設定
. GemSetSubstDestinationX() － インデクスでの移動先ロケーションの設定

（1）呼出書式

[c,C++]

```
API int    APIX GemSetSubstDestination(
        int       eqid,                  // 通信対象装置 ID(0,1,2,...)
        char      *substid,              // SUBSTID が格納されている領域のポインタ
        char      *dest                  // 設定したい移動先ロケーション格納ポインタ
);

API int    APIX GemSetSubstDestinationX(
        int       eqid,                  // 通信対象装置 ID(0,1,2,...)
        int       index,                 // 基板情報のインデクス
        char      *dest                  // 設定したい移動先ロケーション格納ポインタ
);
```

[.NET VB]

```
Function GemSetSubstDestination (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal dest As String) As Int32

Function GemSetSubstDestinationX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal dest As String) As Int32
```

[.NET C#]

```
int GemSetSubstDestination(
        int eqid,
        byte[] substid,
        byte[] dest );

int GemSetSubstDestinationX(
        int eqid,
        int index,
        byte[] dest );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

dest

設定したい SUBSTID の移動先ロケーション（文字列）です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 ID の移動先ロケーション(文字列)を設定します。

3．15．2．18 GemGetSubstDestination() –移動先ロケーションの取得
. GemGetSubstDestinationX() – インデクスでの移動先ロケーションの取得

（1）呼出書式

[c,C++]

```
API int    APIX GemGetSubstDestination(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        char      *dest                 // 取得した移動先ロケーション格納ポインタ
);

API int    APIX GemGetSubstDestinationX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        char      *dest                 // 取得した移動先ロケーション格納ポインタ
);
```

[.NET VB]

```
Function GemGetSubstDestination (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal dest As String) As Int32

Function GemGetSubstDestinationX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal dest As String) As Int32
```

[.NET C#]

```
int GemGetSubstDestination(
        int eqid,
        byte[] substid,
        byte[] dest );

int GemGetSubstDestinationX(
        int eqid,
        int index,
        byte[] dest );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

dest

取得した SUBSTID の移動先ロケーションを格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 Id の移動先ロケーション(文字列)を取得します。

3．15．2．19 GemSetSubstBatchLocId() – バッチロケーションの設定
. GemSetSubstBatchLocIdX() – インデクスでのバッチロケーションの設定

（1）呼出書式

[c,C++]

```
API int    APIX GemSetSubstBatchLocId(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        char      *blocid               // 設定したいバッチロケーション格納ポインタ
);

API int    APIX GemSetSubstBatchLocIdX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        char      *blocid               // 設定したいバッチロケーション格納ポインタ
);
```

[.NET VB]

```
Function GemSetSubstBatchLocId (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal blocid As String) As Int32

Function GemSetSubstBatchLocIdX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal blocid As String) As Int32
```

[.NET C#]

```
int GemSetSubstBatchLocId(
        int eqid,
        byte[] substid,
        byte[] blocid );

int GemSetSubstBatchLocIdX(
        int eqid,
        int index,
        byte[] blocid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

blocid

設定したい SUBSTID のバッチロケーション（文字列）です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 ID のバッチロケーション(文字列)を設定します。

3．15．2．20 GemGetSubstBatchLocId() – バッチロケーションの取得
. GemGetSubstBatchLocIdX() – インデクスでのバッチロケーションの取得

（1）呼出書式

[c,C++]

```
API int    APIX GemGetSubstBatchLocId(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        char      *blocid               // 取得したバッチロケーション格納ポインタ
);

API int    APIX GemGetSubstBatchLocIdX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        char      *blocid               // 取得したバッチロケーション格納ポインタ
);
```

[.NET VB]

```
Function GemGetSubstBatchLocId (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal blocid As String) As Int32

Function GemGetSubstBatchLocIdX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal blocid As String) As Int32
```

[.NET C#]

```
int GemGetSubstBatchLocId(
        int eqid,
        byte[] substid,
        byte[] blocid );

int GemGetSubstBatchLocIdX(
        int eqid,
        int index,
        byte[] blocid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

blocid

取得した SUBSTID のバッチロケーションを格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 Id のバッチロケーション(文字列)を取得します。

3．15．2．21 GemSetSubstPosInBatch() – バッチ内位置の設定
. GemSetSubstPosInBatchX() – インデクスでのバッチ内位置の設定

（1）呼出書式

[c,C++]

```
API int    APIX GemSetSubstPosInBatch(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        char      *posid                // 設定したいバッチ内位置格納ポインタ
);

API int    APIX GemSetSubstPosInBatchX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        char      *posid                // 設定したいバッチ内位置格納ポインタ
);
```

[.NET VB]

```
Function GemSetSubstPosInBatch (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal posid As String) As Int32

Function GemSetSubstPosInBatchX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal posid As String) As Int32
```

[.NET C#]

```
int GemSetSubstPosInBatch(
        int eqid,
        byte[] substid,
        byte[] posid );

int GemSetSubstPosInBatchX(
        int eqid,
        int index,
        byte[] posid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

posid

設定したい SUBSTID のバッチ内位置（文字列）です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 ID のバッチ内位置(文字列)を設定します。

3．15．2．22 GemGetSubstPosInBatch() – バッチ内位置の取得
. GemGetSubstPosInBatchX() – インデクスでのバッチ内位置の取得

（1）呼出書式

[c,C++]

```
API int    APIX GemGetSubstPosInBatch(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        char      *posid                // 取得したバッチ内位置格納ポインタ
);

API int    APIX GemGetSubstPosInBatchX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        char      *posid                // 取得したバッチ内位置格納ポインタ
);
```

[.NET VB]

```
Function GemGetSubstPosInBatch (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal posid As String) As Int32

Function GemGetSubstPosInBatchX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal posid As String) As Int32
```

[.NET C#]

```
int GemGetSubstPosInBatch(
        int eqid,
        byte[] substid,
        byte[] posid );

int GemGetSubstPosInBatchX(
        int eqid,
        int index,
        byte[] posid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

posid

取得した SUBSTID のバッチ内位置を格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

指定された基板 Id のバッチ内位置(文字列)を取得します。

３．15．２．23 GemSetSubsState() － 基板状態の設定
. GemSetSubstStateX() － インデクスでの基板状態の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetSubstState(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       state                 // 設定したい状態値
);

API int APIX GemSetSubstInfosStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       state                 // 設定したい状態値
);
```

[.NET VB]

```
Function GemSetSubstState (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal state As Int32) As Int32

Function GemSetSubstStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal state As Int32) As Int32
```

[.NET C#]

```
int GemSetSubstState(
        int eqid,
        byte[] substid,
        int state );

int GemSetSubstStateX(
        int eqid,
        int index,
        int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

state

設定したい基板の状態値です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板の状態値を設定します。

デフォルトの値は次のとおりです。

基板状態のデフォルト値

状態値記号	値
ST_StSource	0
ST_AtWork	1
ST_AtDestination	2

3．15．2．24 GemGetSubstState() － 基板情報状態の取得
. GemGetSubstStateX() － インデクスでの基板情報状態の取得

（1）呼出書式

[c,C++]

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

API int APIX GemGetSubstStateX(
        int         eqid,
        int         index,                  // 基板情報のインデクス
        int         *state                  // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetSubstState (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef state As Int32) As Int32

Function GemGetSubstStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef state As Int32) As Int32
```

[.NET C#]

```
int GemGetSubstState(
        int eqid,
        byte[] substid,
        ref int state );

int GemGetSubstStateX(
        int eqid,
        int index,
        ref int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

substid

基板IDが格納されている領域のポインタです。

state

取得したSUBSTID情報の状態値を格納する領域のポインタです。

index

基板情報のインデクスです。登録時にGemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板の状態値を取得します。

デフォルトの値は次のとおりです。

基板状態のデフォルト値

状態値記号	値
ST_StSource	0
ST_AtWork	1
ST_AtDestination	2

3．15．2．25 GemSetSubsMaterialStatus () － 基板処理品質状態の設定
. GemSetSubstMaterialStatusX() － インデクスでの基板処理品質状態の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetSubstMaterialStatus(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       status                // 設定したい状態値
);

API int APIX GemSetSubstInfosStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       status                // 設定したい状態値
);
```

[.NET VB]

```
Function GemSetSubstMaterialStatus (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal state As Int32) As Int32

Function GemSetSubstMaterialStatusX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal state As Int32) As Int32
```

[.NET C#]

```
int GemSetSubstMaterialStatus(
        int eqid,
        byte[] substid,
        int state );

int GemSetSubstMaterialStatusX(
        int eqid,
        int index,
        int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

status

設定したい基板の処理品質状態値です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板の処理品質状態値を設定します。

状態値の定義はユーザが行うことになります。

3．15．2．26 GemGetSubstMaterialStatus() – 基板処理品質状態の取得
. GemGetSubstMaterialStatusX() – インデクスでの基板処理品質状態の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetSubstMaterialStatus(
      int       eqid,                  // 通信対象装置 ID(0,1,2,...)
      char      *substid,              // SUBSTID が格納されている領域のポインタ
      int       *status                // 取得した状態値格納用領域ポインタ
);

API int APIX GemGetSubstMaterialStatusX(
      int       eqid,                  // 通信対象装置 ID(0,1,2,...)
      int       index,                 // 基板情報のインデクス
      int       *status                // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetSubstMaterialStatus (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef state As Int32) As Int32

Function GemGetSubstMaterialStatusX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef state As Int32) As Int32
```

[.NET C#]

```
int GemGetSubstMaterialStatus(
        int eqid,
        byte[] substid,
        ref int state );

int GemGetSubstMaterialStatusX(
        int eqid,
        int index,
        ref int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

status

取得した SUBSTID 情報の処理品質状態値を格納する領域のポインタです。

index

基板情報のインデクスです。登録時にGemAllocSubstInfo()関数によって与えられます。
インデクスは、SUBSTIDからGemGetSubstIdIndex()関数で取得することができます。

（3）戻り値

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

（4）説明

基板の処理品質状態値を取得します。

状態値の定義はユーザが行うことになります。

3．15．2．27 GemSetSubsProcState () － 基板処理状態の設定
. GemSetSubstProcStateX() － インデクスでの基板処理状態の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetSubstProcState(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       state                 // 設定したい状態値
);

API int APIX GemSetSubstInfosStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       state                 // 設定したい状態値
);
```

[.NET VB]

```
Function GemSetSubstProcState (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal state As Int32) As Int32

Function GemSetSubstProcStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal state As Int32) As Int32
```

[.NET C#]

```
int GemSetSubstProcState(
        int eqid,
        byte[] substid,
        int state );

int GemSetSubstProcStateX(
        int eqid,
        int index,
        int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

state

設定したい基板の処理状態値です。
値=(-1)は使用不可を意味する値になります。

index

基板情報のインデクスです。登録時にGemAllocSubstInfo()関数によって与えられます。
インデクスは、SUBSTIDからGemGetSubstIdIndex()関数で取得することができます。

（3）戻り値

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

（4）説明

基板の処理状態値を設定します。
デフォルトの値は次のとおりです。

基板処理状態のデフォルト値

状態値記号	値
ST_SUBST_NeedsProcessing	0
ST_SUBST_InProcessing	1
ST_SUBST_processed	2
ST_SUBST_Aborted	3
ST_SUBST_Stopped	4
ST_SUBST_Rejected	5
ST_SUBST_Lost	6

３．15．２．28 GemGetSubstProcState() － 基板処理状態の取得
. GemGetSubstProcStateX() － インデクスでの基板処理状態の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetSubstProcState(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       *state                // 取得した状態値格納用領域ポインタ
);

API int APIX GemGetSubstProcStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       *state                // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetSubstProcState (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef state As Int32) As Int32

Function GemGetSubstProcStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef state As Int32) As Int32
```

[.NET C#]

```
int GemGetSubstProcState(
        int eqid,
        byte[] substid,
        ref int state );

int GemGetSubstProcStateX(
        int eqid,
        int index,
        ref int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

state

取得した SUBSTID 情報の処理状態値を格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板の処理状態値を取得します。

デフォルトの値は次のとおりです。

基板処理状態のデフォルト値

状態値記号	値
ST_SUBST_NeedsProcessing	0
ST_SUBST_InProcessing	1
ST_SUBST_processed	2
ST_SUBST_Aborted	3
ST_SUBST_Stopped	4
ST_SUBST_Rejected	5
ST_SUBST_Lost	6

3．15．2．29 GemSetSubsLocState () － 基板ロケーション状態の設定
. GemSetSubstLocStateX() － インデクスでの基板ロケーション状態の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetSubstLocState(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       state                 // 設定したい状態値
);

API int APIX GemSetSubstInfosStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       state                 // 設定したい状態値
);
```

[.NET VB]

```
Function GemSetSubstLocState (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal state As Int32) As Int32

Function GemSetSubstLocStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal state As Int32) As Int32
```

[.NET C#]

```
int GemSetSubstLocState(
        int eqid,
        byte[] substid,
        int state );

int GemSetSubstLocStateX(
        int eqid,
        int index,
        int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

state

設定したい基板のロケーション状態値です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板のロケーション状態値を設定します。
デフォルトの値は次のとおりです。

基板ロケーション状態のデフォルト値

状態値記号	値
ST_SUBST_Unoccupied	0
ST_SUBST_Occupied	1

3．15．2．30 GemGetSubstLocState() － 基板ロケーション状態の取得
. GemGetSubstLocStateX() － インデクスでの基板ロケーション状態の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetSubstLocState(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       *state                // 取得した状態値格納用領域ポインタ
);

API int APIX GemGetSubstLocStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       *state                // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetSubstLocState (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef state As Int32) As Int32

Function GemGetSubstLocStateX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef state As Int32) As Int32
```

[.NET C#]

```
int GemGetSubstLocState(
        int eqid,
        byte[] substid,
        ref int state );

int GemGetSubstLocStateX(
        int eqid,
        int index,
        ref int state );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

state

取得した SUBSTID 情報のロケーション状態値を格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板のロケーション状態値を取得します。
デフォルトの値は次のとおりです。

基板ロケーション状態のデフォルト値

状態値記号	値
ST_SUBST_Unoccupied	0
ST_SUBST_Occupied	1

３．15．２．31 GemSetSubsUsage () － 基板 Usage の設定
. GemSetSubstUsageX() － インデクスでの基板 Usage の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetSubstUsage(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       usage                 // 設定したい Usage 値
);

API int APIX GemSetSubstInfosStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       usage                 // 設定したい Usage 値
);
```

[.NET VB]

```
Function GemSetSubstUsage (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal usage As Int32) As Int32

Function GemSetSubstUsageX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal usage As Int32) As Int32
```

[.NET C#]

```
int GemSetSubstUsage(
        int eqid,
        byte[] substid,
        int usage );

int GemSetSubstUsageX(
        int eqid,
        int index,
        int usage );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

usage

設定したい基板の Usage 値です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板の Usage 値を設定します。
デフォルトの値は次のとおりです。

基板 Usage のデフォルト値

状態値記号	値
SUBST_USAGE_Product	0
SUBST_USAGE_Test	1
SUBST_USAGE_Filler	2

３．15．２．32　GemGetSubstUsage()　－ 基板 Usage の取得
.　　GemGetSubstUsageX() － インデクスでの基板 Usage の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetSubstUsage(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       *usage                // 取得した Usage 値格納用領域ポインタ
);

API int APIX GemGetSubstUsageX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       *usage                // 取得した Usage 値格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetSubstUsage (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef usage As Int32) As Int32

Function GemGetSubstUsageX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef usage As Int32) As Int32
```

[.NET C#]

```
int GemGetSubstUsage(
        int eqid,
        byte[] substid,
        ref int usage );

int GemGetSubstUsageX(
        int eqid,
        int index,
        ref int usage );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

usage

取得した SUBSTID 情報の Usage 値を格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板の Usage 値を取得します。
デフォルトの値は次のとおりです。

基板 Usage のデフォルト値

状態値記号	値
SUBST_USAGE_Product	0
SUBST_USAGE_Test	1
SUBST_USAGE_Filler	2

３．15．２．33 GemSetSubsType () － 基板 Type の設定
. GemSetSubstTypeX() － インデクスでの基板 Type の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetSubstType(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       type                  // 設定したい Type 値
);

API int APIX GemSetSubstInfosStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       type                  // 設定したい Type 値
);
```

[.NET VB]

```
Function GemSetSubstType (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal stype As Int32) As Int32

Function GemSetSubstTypeX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal stype As Int32) As Int32
```

[.NET C#]

```
int GemSetSubstType(
        int eqid,
        byte[] substid,
        int stype );

int GemSetSubstTypeX(
        int eqid,
        int index,
        int stype );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

type

設定したい基板の Type 値です。
値=(-1)は使用不可を意味する値になります。

index

基板情報のインデクスです。登録時にGemAllocSubstInfo()関数によって与えられます。
インデクスは、SUBSTIDからGemGetSubstIdIndex()関数で取得することができます。

（3）戻り値

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

（4）説明

基板のType値を設定します。
デフォルトの値は次のとおりです。

基板Typeのデフォルト値

状態値記号	値
ST_SUBST_TYPE_Wafer	0
ST_SUBST_ TYPE_FlatPanel	1
ST_SUBST_ TYPE_CD	2
ST_SUBST_ TYPE_Mask	3

３．15．２．34 GemGetSubstType() － 基板 Type の取得
. GemGetSubstTypeX() － インデクスでの基板 Type の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetSubstType(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        int       *type                 // 取得した Type 値格納用領域ポインタ
);

API int APIX GemGetSubstTypeX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        int       *type                 // 取得した Type 値格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetSubstType (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef stype As Int32) As Int32

Function GemGetSubstTypeX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef stype As Int32) As Int32
```

[.NET C#]

```
int GemGetSubstType(
        int eqid,
        byte[] substid,
        ref int stype );

int GemGetSubstTypeX(
        int eqid,
        int index,
        ref int stype );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

type

取得した SUBSTID 情報の Type 値を格納する領域のポインタです。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板の Type 値を取得します。
デフォルトの値は次のとおりです。

基板 Type のデフォルト値

状態値記号	値
ST_SUBST_TYPE_Wafer	0
ST_SUBST_ TYPE_FlatPanel	1
ST_SUBST_ TYPE_CD	2
ST_SUBST_ TYPE_Mask	3

３．15．２．35 GemSetSubsLocHistory () － 基板通過ロケーション履歴の設定
. GemSetSubstLocHistoryX() － インデクスでの基板通過ロケーション履歴の設定

（1）呼出書式

[c,C++]

```
API int APIX GemSetSubstLocHistory(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        char        *substid,               // SUBSTID が格納されている領域のポインタ
        TSUBST_LOC_HIST *pinfo              // 設定したい通過ロケーション履歴情報のポインタ
);

API int APIX GemSetSubstInfosStateX(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        int         index,                  // 基板情報のインデクス
        TSUBST_LOC_HIST *pinfo              // 設定したい通過ロケーション履歴情報のポインタ
);
```

[.NET VB]

```
Function GemSetSubstLocHistory (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef hinfo As dsh_info.TSUBST_LOC_HIST) As Int32

Function GemSetSubstLocHistoryX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef hinfo As dsh_info.TSUBST_LOC_HIST) As Int32
```

[.NET C#]

```
int GemSetSubstLocHistory(
        int eqid,
        byte[] substid,
        ref TSUBST_LOC_HIST hinfo );

int GemSetSubstLocHistoryX(
        int eqid,
        int index,
        ref TSUBST_LOC_HIST hinfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

pinfo

設定したい基板の通過ロケーション履歴情報が格納されている領域のポインタです。
値=(-1)は使用不可を意味する値になります。

index

基板情報のインデクスです。登録時にGemAllocSubstInfo()関数によって与えられます。
インデクスは、SUBSTIDからGemGetSubstIdIndex()関数で取得することができます。

（3）戻り値

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

（4）説明

基板の基板通過ロケーション履歴情報を設定します。

3．15．2．36 GemGetSubstLocHistory() - 基板通過ロケーション履歴の取得
. GemGetSubstLocHistoryX() - インデクスでの基板通過ロケーション履歴の取得

（1）呼出書式

[c,C++]

```
API int APIX GemGetSubstLocHistory(
        int         eqid,                   // 通信対象装置ID(0,1,2,...)
        char        *substid,               // SUBSTIDが格納されている領域のポインタ
        TSUBST_LOC_HIST *pinfo              // 取得した通過ロケーション履歴情報格納用領域ポインタ
);

API int APIX GemGetSubstLocHistoryX(
        int         eqid,                   // 通信対象装置ID(0,1,2,...)
        int         index,                  // 基板情報のインデクス
        TSUBST_LOC_HIST *pinfo              // 取得した通過ロケーション履歴情報格納用領域ポインタ
);
```

[.NET VB]

```
Function GemGetSubstLocHistory (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef hinfo As dsh_info.TSUBST_LOC_HIST) As Int32

Function GemGetSubstLocHistoryX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByRef hinfo As dsh_info.TSUBST_LOC_HIST) As Int32
```

[.NET C#]

```
int GemGetSubstLocHistory(
        int eqid,
        byte[] substid,
        ref TSUBST_LOC_HIST hinfo );

int GemGetSubstLocHistoryX(
        int eqid,
        int index,
        ref TSUBST_LOC_HIST hinfo );
```

（2）引数

eqid

通信エンジンが通信する対象装置IDを指定します。装置IDは0から始まる番号です。

substid

基板IDが格納されている領域のポインタです。

pinfo

取得したSUBSTID情報の通過ロケーション履歴情報を格納する領域のポインタです。

index

基板情報のインデクスです。登録時にGemAllocSubstInfo()関数によって与えられます。

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

（3）戻り値

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

（4）説明

基板の基板通過ロケーション履歴情報を取得します。

3．15．2．37 GemAddSubsLocHistory () － 基板通過ロケーション履歴の追加
. GemAddSubstLocHistoryX() － インデクスでの基板通過ロケーション履歴の追加

（1）呼出書式

[c,C++]

```
API int APIX GemAddSubstLocHistory(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        char      *substid,             // SUBSTID が格納されている領域のポインタ
        char      *loc,                 // ﾛｹｰｼｮﾝ名
        char      *time_in,             // 入り時刻
        char      *time_out             // 出時刻
);

API int APIX GemAddSubstInfosStateX(
        int       eqid,                 // 通信対象装置 ID(0,1,2,...)
        int       index,                // 基板情報のインデクス
        char      *loc,                 // ﾛｹｰｼｮﾝ名
        char      *time_in,             // 入り時刻
        char      *time_out             // 出時刻
);
```

[.NET VB]

```
Function GemAddSubstLocHistory (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByVal loc As String,
        ByVal time_in As String,
        ByVal time_out As String) As Int32

Function GemAddSubstLocHistoryX (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal loc As String,
        ByVal time_in As String,
        ByVal time_out As String) As Int32
```

[.NET C#]

```
int GemAddSubstLocHistory(
        int eqid,
        byte[] substid,
        byte[] loc,
        byte[] time_in,
        byte[] time_out );

int GemAddSubstLocHistoryX(
        int eqid,
        int index,
        byte[] loc,
```

```
    byte[] time_in,
    byte[] time_out );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

基板 ID が格納されている領域のポインタです。

loc

通過ロケーション名です。

time_in

入り時刻です。

time_out

出時刻です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。
インデクスは、SUBSTID から GemGetSubstIdIndex()関数で取得することができます。

（3）戻り値

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

（4）説明

基板の基板通過ロケーション履歴情報を追加します。

3．15．2．38 GemGetSubstList() - 全登録基板 ID 取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetSubstList(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        TTEXT_DLIST **list                  // 取得ﾘｽﾄ格納ﾎﾟｲﾝﾀの格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetSubstList (
        ByVal eqid As Int32,
        ByRef list As IntPtr) As Int32
```

[.NET C#]

```
int GemGetSubstList(
        int eqid,
        IntPtr list );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

list

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

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

（4）説明

システムに登録されている SUBSTID(基板 ID)を TTEXT_DLIST 構造体に取出すための関数です。
info->name_list には NULL が設定されます。(定義名がありません。)
取得した情報の処理が終了した後、DshFreeTText_DLIST()関数で list 内部の情報格納用に使用されているメモリを開放してください。

TTEXT_DLIST 構造体は次のとおりです。

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

3．15．2．39 GemGetSubstId() - インデクスから基板 ID の取得

（1）呼出書式

[c,C++]

```
API int  APIX EgnGetSubstId(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        int     index,                  // 基板情報のインデクス
        char    *substid                // 取得した基板 ID を格納する領域のポインタ
);
```

[.NET VB]

```
Function GemGetSubstId (
        ByVal eqid As Int32,
        ByVal index As Int32,
        ByVal substid As String) As Int32
```

[.NET C#]

```
int GemGetSubstId(
        int eqid,
        int index,
        byte[] substid );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

index

基板情報のインデクスです。登録時に GemAllocSubstInfo()関数によって与えられます。
インデクスは、基板 ID から GemGetSubstIdIndex()関数で取得することができます。

substid

基板 ID を格納するための領域のポインタです。

（3）戻り値

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

（4）説明

基板情報のインデクスから基板 ID を取得し、substid に格納します。
正常に取得できた場合は関数戻り値として 0 が返却されます。

3．15．2．40　GemGetSubstIdIndex() -基板 ID からインデクスの取得

（1）呼出書式

[c,C++]

```
API int  APIX EgnGetSubstIdIndex(
        int     eqid,                   // 通信対象装置 ID(0,1,2,...)
        char    *substid,               // 基板 ID が格納されている領域のポインタ
        int     *index                  // 取得したインデクスを格納するための領域のポインタ
);
```

[.NET VB]

```
Function GemGetSubstIdIndex (
        ByVal eqid As Int32,
        ByVal substid As String,
        ByRef index As Int32) As Int32
```

[.NET C#]

```
int GemGetSubstIdIndex(
        int eqid,
        byte[] substid,
        ref int index );
```

（2）引数

eqid

通信エンジンが通信する対象装置 ID を指定します。装置 ID は 0 から始まる番号です。

substid

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

index

取得したインデクスの値を格納するための領域のポインタです。

（3）戻り値

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

（4）説明

substid に指定される基板 ID から基板情報インデクスを取得するための関数です。
取得されたインデクスは index で指定された領域に格納されます。
正常に取得できた場合は関数戻り値として 0 が返却されます。

３．15．３　SUBST 基板関連ライブラリ関数

３．15．３．１　DshFreeTSUBST_INFO() - 基板情報構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTSUBST_INFO(
      TSUBST_INFO *pinfo          // ﾒﾓﾘを開放したい SUBST 情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTSUBST_INFO (
        ByRef info As dsh_info.TSUBST_INFO)
```

[.NET C#]

```
void DshFreeTSUBST_INFO(
        ref TSUBST_INFO info );
```

（2）引数

pinfo

メモリを解放したい基板情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

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

3．15．3．2　DshCopyTSUBST_INFO() - 基板情報構造体メモリのコピー

（1）呼出書式

[c,C++]

```
API int APIX   DshCopyTSUBT_INFO(
        TSUBST_INFO *dinfo,                 // ｺﾋﾟｰ先のﾎﾟｲﾝﾀ
        TSUBST_INFO *sinfo                  // ｺﾋﾟｰ元のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshCopyTSUBST_INFO (
          ByRef dinfo As dsh_info.TSUBST_INFO,
          ByRef sinfo As dsh_info.TSUBST_INFO) As Int32
```

[.NET C#]

```
int DshCopyTSUBST_INFO(
          ref TSUBST_INFO dinfo,
          ref TSUBST_INFO sinfo);
```

（2）引数

dinfo

基板情報のコピー先構造体メモリのポインタです。

sinfo

コピー元の基板情報が格納されている構造体メモリのポインタです。

（3）戻り値

戻り値	意味
0	正常にｺﾋﾟｰできた。
(-1)	sinfo またはdinfo の値がNULL だったのでｺﾋﾟｰできなかった。

（4）説明

sinfo が指すTSUBST_INFO 構造体内に格納されている基板情報をdinfo が指定するTSUBST_INFO 構造体にコピーします。

dinfo 内のメンバーで新しいメモリが必要なものは本関数が取得します。

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

３．15．３．３　DshFreeTSUBST_LOC_HIST() - 通過ロケーション履歴構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTSUBST_LOC_HIST(
      TSUBST_LOC_HIST *pinfo          // ﾒﾓﾘを開放したいｱｸｼｮﾝ情報が格納されている構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTSUBST_LOC_HIST (
        ByRef pinfo As dsh_info.TSUBST_LOC_HIST)
```

[.NET C#]

```
void DshFreeTSUBST_LOC_HIST(
        ref TSUBST_LOC_HIST pinfo );
```

（2）引数

pinfo

メモリを解放したい通過ロケーション履歴構造体のポインタです。

（3）戻り値

なし。

（4）説明

TSUBST_LOC_HIST 構造体内で情報格納用に使用されているメモリを全て解放します。

３．15．３．４　DshCopyTSUBST_LOC_HIST() - 通過ロケーション履歴構造体メモリのコピー

（1）呼出書式

[c,C++]

```
API int APIX   DshCopyTSUBST_INFO(
        TSUBST_LOC_HIST *dinfo,                 // ｺﾋﾟｰ先のﾎﾟｲﾝﾀ
        TSUBST_LOC_HIST *sinfo                   // ｺﾋﾟｰ元のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshCopyTSUBST_LOC_HIST (
          ByRef dinfo As dsh_info.TSUBST_LOC_HIST,
          ByRef sinfo As dsh_info.TSUBST_LOC_HIST) As Int32
```

[.NET C#]

```
int DshCopyTSUBST_LOC_HIST(
          ref TSUBST_LOC_HIST dinfo,
          ref TSUBST_LOC_HIST sinfo );
```

（2）引数

dinfo

通過ロケーション履歴のコピー先構造体メモリのポインタを格納するポインタです。

sinfo

コピー元の通過ロケーション履歴が格納されている構造体メモリのポインタです。

（3）戻り値

戻り値	意味
0	正常にｺﾋﾟｰできた。
(-1)	sinfo またはdinfo の値がNULL だったのでｺﾋﾟｰできなかった。

（4）説明

sinfoが指すTSUBST_LOC_HIST構造体内に格納されている通過ロケーション履歴を新しく確保したメモリのTSUBST_LOC_HIST 構造体に構造体ごとコピーします。

dinfo には、新しく確保したメモリのポインタを格納し返却します。

3．15．3．5　DshInitSubstInfo - TSUBST_INFO の初期設定

（1）呼出書式

[c,C++]

```
API int  APIX DshInitSubstInfo(
        TSUBST_INFO  *info,                // TSUBST_INFO 構造体のﾎﾟｲﾝﾀ
        char         *substid,             // 基板 ID
        char         *acquierdID,          // 読み込まれた ID
        char         *lotID,               // 基板の lot id
        char         *subst_locID,         // 基板の現在 location
        char         *source,              // 移動元位置
        char         *destination,         // 移動先位置
        char         *batch_locID,         // ﾊﾞｯﾁの現在位置
        char         *subst_pos_in_batch,  // ﾊﾞｯﾁ内の位置
        int          substIDstatus,        // 基板 ID の読込み状態
        int          matrerial_status,     // 処理品質基板状態
        int          subst_pro_state,      // 処理状態
        int          subst_state,          // TRANSPORT 状態
        int          subst_loc_state,      // ﾛｹｰｼｮﾝの状態
        int          subst_usage,          // 基板の種類ｺｰﾄﾞ
        int          subst_type            // 基板のﾀｲﾌﾟｺｰﾄﾞ
);
```

[.NET VB]

```
Sub DshInitSubstInfo (
        ByRef info As dsh_info.TSUBST_INFO,
        ByVal substid As String,
        ByVal acquiredID As String,
        ByVal lotID As String,
        ByVal subst_locID As String,
        ByVal subst_source As String,
        ByVal subst_destination As String,
        ByVal batch_locID As String,
        ByVal subst_pos_in_batch As String,
        ByVal substIDstatus As Int32,
        ByVal material_status As Int32,
        ByVal subst_proc_state As Int32,
        ByVal subst_state As Int32,
        ByVal subst_loc_state As Int32,
        ByVal subst_usage As Int32,
        ByVal subst_type As Int32)
```

[.NET C#]

```
void DshInitSubstInfo(
        ref TSUBST_INFO info,
        byte[] substid,
        byte[] acquiredID,
        byte[] lotID,
```

```
    byte[] subst_locID,
    byte[] subst_source,
    byte[] subst_destination,
    byte[] batch_locID,
    byte[] subst_pos_in_batch,
    int substIDstatus,
    int material_status,
    int subst_proc_state,
    int subst_state,
    int subst_loc_state,
    int subst_usage,
    int subst_type );
```

（2）引数

info

TSUBST_INFO 構造体のポインタです。このメンバーを初期設定します。

substid

設定する基板 ID（文字列）です。

acquierdID

基板ﾘｰﾀﾞｰで読み込まれた基板 ID です。

lotID

基板のロット ID です。

subst_locID

基板の現在 location（バッチプロセス装置以外）

source

基板の移動元 location

destination

基板の移動先 location

batch_locID

バッチの現在 location (バッチプロセス装置)

subst_pos_in_batch

バッチ内の現在の基板 loction(バッチプロセス装置)

subst_IDstatus

基板の ID 読込み状態です。

material_status

基板の処理品質基板状態です。

subst_proc_status

基板の処理状態です。

subst_state

基板のトランスポート状態です。

subst_loc_state

基板の locatoin 状態です。

subst_usage

基板の種類コードです。

subst_type

基板のタイプコードです。

（3）戻り値

なし。

（4）説明

本関数は APP が OFFLINE で基板情報を生成する際に使用することができます。
最初に info 内をクリアします。そして、引数で指定された情報を info 内に設定します。
メモリが必要なメンバーについてはメモリを確保し情報をコピーします。
なお、バッチ処理装置かどうかによって引数の値を NULL にする引数もあります。
batch 名がついている引数はバッチプロセス装置用の引数です。

ユーザのシステムで不要な引数については、NULL または 0 を指定してください。

また、基板の移動履歴情報(TSUBST_LOC_HIST)の設定については次の DshPutSubstLocHist()関数を使ってください。この移動履歴情報のための領域を最大 64 個分予約してあります。

TSUBST_INFO 構造体の情報を使用終了後は DshFreeTSUBST_INFO()関数を使ってこ構造体内部で使用されたメモリを開放してください。

3．15．3．6　DshPutSubstLocHist() - 基板移動履歴情報の追加

（1）呼出書式

[c,C++]

```
API int APIX DshPutSubstLotHist(
        TSUBST_INFO  *info,              // 基板情報構造体のﾎﾟｲﾝﾀ
        char         *loc,               // 移動ﾛｹｰｼｮﾝ ID
        char         *time_in,           // 入時刻
        char         *time_out           // 出時刻
);
```

[.NET VB]

```
Function DshPutSubstLocHist (
        ByRef info As dsh_info.TSUBST_INFO,
        ByVal loc As String,
        ByVal time_in As String,
        ByVal time_out As String) As Int32
```

[.NET C#]

```
int DshPutSubstLocHist(
        ref TSUBST_INFO info,
        byte[] loc,
        byte[] time_in,
        byte[] time_out );
```

（2）引数

info

　　移動履歴を加える基板情報構造体のポインタです。

loc

　　基板移動ロケーション ID です。

time_in

　　指定されたロケーション ID 位置に入った時刻です。(YYYYMMDDhhmmsscc)

time_out

　　指定されたロケーション ID 位置から出た時刻です。(YYYYMMDDhhmmsscc)

（3）戻り値

戻り値	意味
0	正常に設定できた。
(-1)	予約されているｶｳﾝﾄ数を超えていたので設定できなかった。

（4）説明

先に DshInitSubstInfo()で初期設定された基板情報構造体の hist が示す TSUBST_LOC_HIST に１個のロケーション移動情報を加えます。

予め最大 64 個予約された数の履歴リスト上には本関数が実行される順にスロット情報を追加していきます。

60 個以内の追加であれば設定後 0 を返却します。

もし、64 個を超える場合は、設定しないで(-1)を返却します。

実際に設定された合計の履歴情報の数は TSUBST_LOC_HIST 内の hist_count に保存されています。

３．15．３．７　DshInitSubstLocHist() - 基板位置履歴情報の初期設定

（1）呼出書式

[c,C++]

```
API void APIX DshInitSubstLocHist(
        TSUBST_LOC_HIST   *info,            // 基板履歴情報構造体のﾎﾟｲﾝﾀ
        int    count                        // 基板位置履歴数
);
```

[.NET VB]

```
Sub DshInitSubstLocHist (
          ByRef info As dsh_info.TSUBST_LOC_HIST,
          ByVal count As Int32)
```

[.NET C#]

```
void DshInitSubstLocHist(
          ref TSUBST_LOC_HIST info,
          int count);
```

（2）引数

info

基板位置履歴情報を格納する構造体のポインタです。

count

格納する履歴数です。

（3）戻り値

なし。

（4）説明

info で指定される TSUBST_LOC_HIST 構造体を初期設定します。
info 内に格納する履歴数を count で指定します。本関数は、履歴メンバーに count 数分のポインタ配列領域を設けます。
TSUBST_INFO 構造体内に対する履歴情報の設定は DshAddSubstLocHist()関数を使用してください。

3．15．3．8　DshAddSubstLocHist() - 基板位置履歴情報の追加設定

（1）呼出書式

[c,C++]

```
API int APIX DshAddSubstLocHist(
        TSUBST_LOC_HIST   *info,            // 基板履歴情報構造体のﾎﾟｲﾝﾀ
        char       *loc,                    // 位置（名）
        char       *time_inf,               // 入時刻
        char       *time_out                // 出時刻
);
```

[.NET VB]

```
Function DshAddSubstLocHist (
          ByRef info As dsh_info.TSUBST_LOC_HIST,
          ByVal loc As String,
          ByVal time_in As String,
          ByVal time_out As String) As Int32
```

[.NET C#]

```
int DshAddSubstLocHist(
          ref TSUBST_LOC_HIST info,
          byte[] loc,
          byte[] time_in,
          byte[] time_out);
```

（2）引数

info

基板位置履歴情報を格納する構造体のポインタです。

loc

ロケーション名です。

time_in

入り時刻です。

timeout

出時刻です。

（3）戻り値

戻り値	意味
0	正常に設定できた。
(-1)	info にはもう追加できなかった

（4）説明

info で指定される TSUBST_LOC_HIST 構造体に履歴情報を追加します。

info 内に格納できる履歴数が既に満杯であった場合にはエラーを返します。

正常に追加できた場合は、０を返します。