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-２／１５

３．３　変数（EC, SV, DVVAL）情報アクセスと通信サービス

２００９年6月

株式会社データマップ

文書番号　DSHGEM-LIB-09-30322-00

[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】

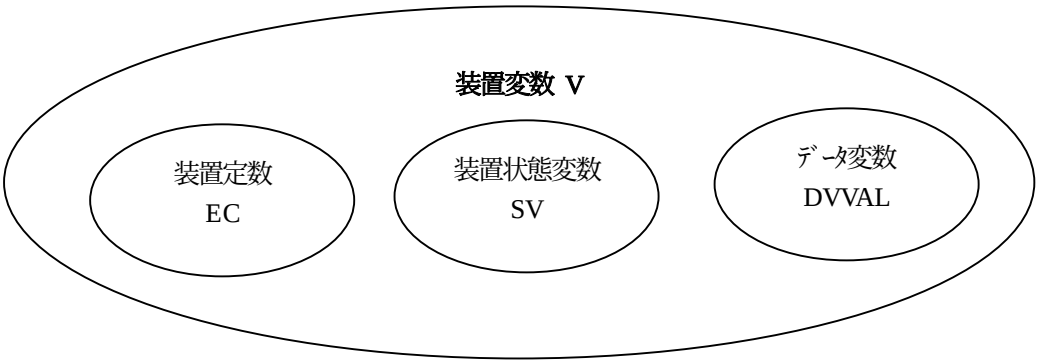
番号	改訂日付	項 目	概 略
1.	2009.6	改訂版	以前の DSHGEM-LIB-07-3032x-00 を全面改訂 .Net VB2008, C#2008 対応関数の説明を追加した。

目　次

(VOL－３に続く)

３．３　変数（EC，SV，DVVAL）情報アクセスと通信サービス

装置変数には以下の３種類の変数があります。



ここで述べる情報は、DSHGEM-LIB が管理します。従って、APP はこれらの情報をアクセスする場合、以下の API 関数を呼出してアクセスすることになります。また、装置に対し変数値要求、定数変更などのメッセージの送信のための関数も準備されています。

（1）情報アクセスと送信 API 関数

装置変数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。

	情報名と API 関数名		機能
1	**変数(EC,SV,DVVAL)**		EC,SV,DVVAL 全変数が対象です。
	(1)	GemSetVVal()	VID で指定された変数の値を設定変更します。
	(2)	GemGetVVal()	VID で指定された変数の値を取得します。
	(3)	GemGetVName()	VID で指定された変数の Vﾃﾞｰﾀ名を取得します。
	(4)	GemGetVUnits()	VID で指定された変数の単位名を取得します。
	(5)	GemGetVFormat()	VID で指定された変数の値の Format と単位ﾃﾞｰﾀﾊﾞｲﾄ長を取得します。
	(6)	GemGetVArraySize()	VID で指定された変数の値のﾃﾞｰﾀの配列ｻｲｽﾞを取得します。
	(7)	GemGetVMin()	VID で指定された変数の値の最小値を取得します。
	(8)	GemGetVMax()	VID で指定された変数の値の最大値を取得します。
	(9)	GemGetVNominal()	VID で指定された変数の値の初期設定値を取得します。
	(10)	GemCheckVVal()	VID で指定された変数の値をﾁｪｯｸします。
	(11)	GemGetVList()	全変数の一覧ﾘｽﾄを取得します。
	(12)	GemGetVSizeMinMax()	変数値の最小、最大配列ｻｲｽﾞを取得します。
	(13)	GemSetMulVLimit()	複数個の変数に対します。ﾘﾐｯﾄ値を設定します。3.10.2.1 で説明します。
	(14)	GemSetVLimit()	１個の変数に対します。ﾘﾐｯﾄ値を設定します。 3.10.2.2 で説明します。
	(15)	GemGetVLimit()	変数に対します。ﾘﾐｯﾄ値を取得します。 3.10.2.3 で説明します。
	(16)	GemDelVLimit()	変数に対します。ﾘﾐｯﾄ値を消去します。 3.10.2.4 で説明します。z
	(17)	GemCheckVLimit()	変数値のﾘﾐｯﾄﾁｪｯｸを行う。 3.10.2.5 で説明します。
	(18)	GemSendS2F45()	変数ﾘﾐｯﾄ属性定義 S2F45 ﾒｯｾｰｼﾞを送信します。 3.10.2.6 で説明します。
	(19)	GemSendS2F47()	変数ﾘﾐｯﾄ属性要求 S2F47 ﾒｯｾｰｼﾞを送信します。 3.10.2.7 で説明します。
2	**EC(装置定数)関連**		対象は EC 変数だけです。
	(1)	GemSetEcVal()	ECID で指定された定数の値を設定変更します。
	(2)	GemGetEcVal()	ECID で指定された定数の値を取得します。
	(3)	GemGetEcName()	ECID で指定された定数の EC 名を取得します。

	(4)	GemGetEcUnits()	ECID で指定された定数の単位名を取得します。
	(5)	GemGetEcFormat()	ECID で指定された定数の値の Format と単位ﾃﾞｰﾀﾊﾞｲﾄ長を取得します。
	(6)	GemGetEcArraySize()	ECID で指定された定数の値のﾃﾞｰﾀ配列ｻｲｽﾞを取得します。
	(7)	GemGetEcMin()	ECID で指定された定数の値の最小値を取得します。
	(8)	GemGetEcMax()	ECID で指定された定数の値の最大値を取得します。
	(9)	GemGetEcNominal()	ECID で指定された定数の値の初期設定値を取得します。
	(10)	GemCheckEcVal()	ECID で指定された定数の値をﾁｪｯｸします。
	(11)	GemGetEcList()	全 EC 変数の一覧ﾘｽﾄを取得します。
	(12)	GemGetEcSizeMinMax()	装置定数値の最小、最大配列ｻｲｽﾞを取得します。
	(13)	GemSendS2F13()	S2F13(装置定数要求)を送信します。
	(14)	GemSendS2F15()	S2F15(装置定数変更)を送信します。
	(15)	GemSendS2F29()	S2F29(装置定数名一覧要求)を送信します。
3	**SV(装置状態変数)関連**		対象は SV 変数だけです。
	(1)	GemSetSvVal()	SVID で指定された状態変数の値を設定変更します。
	(2)	GemGetSvVal()	SVID で指定された状態変数の値を取得します。
	(3)	GemGetSvName()	SVID で指定された状態変数の SV 名を取得します。
	(4)	GemGetSvUnits()	SVID で指定された状態変数の単位名を取得します。
	(5)	GemGetSvFormat()	SVID で指定された状態変数の値の Format と単位ﾃﾞｰﾀﾊﾞｲﾄ長を取得します。
	(6)	GemGetSvArraySize()	SVID で指定された状態変数の値のﾃﾞｰﾀの配列ｻｲｽﾞを取得します。
	(7)	GemGetSvMin()	SVID で指定された状態変数の値の最小値を取得します。
	(8)	GemGetSvMax()	SVID で指定された状態変数の値の最大値を取得します。
	(9)	GemGetSvNominal()	SVID で指定された状態変数の値の初期設定値を取得します。
	(10)	GemCheckEcVal()	SVID で指定された状態変数の値をﾁｪｯｸします。
	(11)	GemGetSvList()	全 SV 変数の一覧ﾘｽﾄを取得します。
	(12)	GemGetSvSizeMinMax()	状態変数値の最小、最大配列ｻｲｽﾞを取得します。
	(13)	GemSendS1F3()	S1F3(装置状態要求)を送信します。
	(14)	GemSendS1F11()	S1F11(状態変数一覧要求)を送信します。
4	**DVVAL(装置ﾃﾞｰﾀ変数)関連**		対象は DVVAL 変数だけです。
	(1)	GemSetDvVal()	DVID で指定されたﾃﾞｰﾀ変数の値を設定変更します。
	(2)	GemGetDvVal()	DVID で指定されたﾃﾞｰﾀ変数の値を取得します。
	(3)	GemGetDvName()	DVID で指定されたﾃﾞｰﾀ変数の Vﾃﾞｰﾀ名を取得します。
	(4)	GemGetDvUnits()	DVID で指定されたﾃﾞｰﾀ変数の単位名を取得します。
	(5)	GemGetDvFormat()	DVID で指定されたﾃﾞｰﾀ変数の値の Format と単位ﾃﾞｰﾀﾊﾞｲﾄ長を取得します。
	(6)	GemGetDvArraySize()	DVID で指定されたﾃﾞｰﾀ変数の値のﾃﾞｰﾀの配列ｻｲｽﾞを取得します。
	(7)	GemGetDvMin()	DVID で指定されたﾃﾞｰﾀ変数の値の最小値を取得します。
	(8)	GemGetDvMax()	DVID で指定されたﾃﾞｰﾀ変数の値の最大値を取得します。
	(9)	GemGetDvNominal()	DVID で指定されたﾃﾞｰﾀ変数の値の初期設定値を取得します。
	(10)	GemCheckDvVal()	DVID で指定されたﾃﾞｰﾀ変数の値をﾁｪｯｸします。
	(11)	GemGetDvList()	全 DVVAL 変数の一覧ﾘｽﾄを取得します。
	(12)	GemGetDvSizeMinMax()	ﾃﾞｰﾀ変数値の最小、最大配列ｻｲｽﾞを取得します。

（2）ライブラリ関数

APP が使用できる変数情報処理関連ライブラリ関数として、以下の関数があります。

	API 関数名	機能
1	DshInitTV_VALUE_LIST()	S2F15 装置定数変更ﾒｯｾｰｼﾞ送信に使用する TV_VALUE_LIST 構造体の初期設定を行います。
2	DshAddTV_VALUE_LIST()	S2F15 装置定数変更ﾒｯｾｰｼﾞ送信に使用する TV_VALUE_LIST 構造体に 1 個の変数情報を設定するための関数です。
3	DshFreeTV_VALUE_LIST()	TV_VALUE_LIST 構造体内に使用されているﾒﾓﾘを開放します。
4	DshFreeTEC_NAME_LIST()	定数名一覧情報、TEC_NAME_LIST 構造体内に使用されているﾒﾓﾘを開放します。
5	DshFreeTSV_NAME_LIST()	状態変数名一覧情報、TSV_NAME_LIST 構造体内に使用されているﾒﾓﾘを開放します。
6	DshEncodeVidList()	VID 格納ﾘｽﾄと VID 数に従って DSHMSG 構造体内に VID ﾘｽﾄ情報をｴﾝｺｰﾄﾞします。 S1F3，S1F13，S2F13，S2F29，S2F47 用です。
7	DshEncodeVarValMsg()	TV_VALUE_LIST 構造体内の変数値情報を DSHMSG 内にｴﾝｺｰﾄﾞします。 S2F15 用です。
8	DshDecodeVarValMsg()	DSHMSG に含まれるﾒｯｾｰｼﾞから変数の値を TV_VALUE_LIST 構造体内に取得します。 S1F4，S2F14 用です。
9	DshGetTV_VALUE_LIST_vid()	TV_VALUE_LIST 構造体内の指定された位置の変数 ID,ﾃﾞｰﾀﾌｫｰﾏｯﾄ、ﾃﾞｰﾀｻｲｽﾞを取得します。
10	DshPutTV_VALUE_LIST_vid()	TV_VALUE_LIST 構造体の指定位置の情報に変数 ID を設定します。 (S2F13,14，S1F3，S1F4 ﾒｯｾｰｼﾞの処理に使用します。

３．３．１　使用する情報構造体

（1）S1F3 ならびに S2F13 要求に対して応答された変数値を格納するために以下の構造体を使用します。
　　また、S2F29 定数値変更時にも設定、送信したい値を与えるために使用します。

```
typedef struct{
        TVID        vid;          // 変数 ID
        int         format;       // 変数値のﾌｫｰﾏｯﾄ( DSHDR2 で定める ITEM のﾌｫｰﾏｯﾄ
        int         asize;        // 変数値の配列ｻｲｽﾞ
        void        *value;       // 変数値が格納されているﾒﾓﾘﾎﾟｲﾝﾀ
} TV_VALUE;

typedef struct{
        int         count;        // 変数 ID の数
        TV_VALUE    **vv_list;    // 変数 ID と変数値格納構造体ﾎﾟｲﾝﾀﾘｽﾄへのﾎﾟｲﾝﾀ
(count 分)
} TV_VALUE_LIST;
```

（2）S2F29 装置定数名一覧要求の応答情報を格納するために以下の構造体を使用します。

```
typedef struct{
        TECID       ecid;         // 装置定数 ID
        char        *name;        // 名前
        int         format;       // 値のﾌｫｰﾏｯﾄ
        int         asize;        // 値の配列ｻｲｽﾞ
        void        *ecmin;       // 値の最小値
        void        *ecmax;       // 値の最大値
        void        *ecdef;       // 値のﾃﾞﾌｫﾙﾄ値(=Nominal,初期設定値)
        char        *units;       // 値の単位名
} TEC_NAME;

typedef struct{
        int         count;        // 装置定数 ID の数
        TEC_NAME    **name_list;  // 定数名情報が格納されている構造体ﾎﾟｲﾝﾀﾘｽﾄへのﾎﾟｲﾝﾀ
} TEC_NAME_LIST;
```

（3）S1F11 装置状態変数名一覧要求の応答情報を格納するために以下の構造体を使用します。

```
typedef struct{
        TSVID       svid;         // 装置状態変数 ID
        char        *name;        // 名前
        char        *units;       // 値の単位名
} TSV_NAME;

typedef struct{
        int         count;        // 装置状態変数 ID の数
        TSV_NAME    **name_list;  // 状態変数名情報が格納されている構造体ﾎﾟｲﾝﾀﾘｽﾄへのﾎﾟｲﾝﾀ
} TSV_NAME_LIST;
```

３．３．２　装置変数情報アクセス関数

以下説明する装置変数情報アクセス関数は、EC(装置定数)、SV(装置状態変数)ならびにDVVAL(装置データ変数)の３つの変数が対象になります。

３．３．３以降にEC，SV，DVVALそれぞれ個別対象のAPI関数の説明を行います。

３．３．２．１　GemSetVVal装置変数()　-　装置変数値設定関数

（1）呼出書式

[c,C++]

```
API int APIX GemSetVVal(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TVID   vid,                     // 変数ID
        void   *val,                    // 設定値ﾎﾟｲﾝﾀ
        int    asize                    // ﾃﾞｰﾀの配列ｻｲｽﾞ
);
```

[.NET VB]

```
Function GemSetVVal (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByVal arraysize As Int32) As Int32
```

[.NET C#]

```
int GemSetVVal(
        int eqid,
        uint vid,
        byte[] val,
        int arraysize );
```

（2）引数

eqid

GEM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。

vid

装置変数IDです。

装置管理情報定義ファイルに登録されている装置変数のIDでなければなりません。

val

設定したい値の格納ポインタです。vidのフォーマットに合わせた値を設定してください。

asize

設定値領域の配列サイズです。

フォーマットA,J の場合は、文字列長になります。

（3）戻り値

戻り値	意味
0	正常に設定された。
(-1)	vidの値が正しくなかった。(登録されていなかった)

（4）説明

vid で指定された V(装置変数)に新しい値を設定します。戻り値が設定結果になります。
正常に設定できた場合は、関数の戻り値は 0 になります。

３．３．２．２　GemGetVVal() - 装置変数値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetVVal(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ格納用
);
```

[.NET VB]

```
Function GemGetVVal (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetVVal(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

vid

装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。vid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。
フォーマット A,J の場合は、文字列長になり、文字列の最後に NULL 文字(=0)が付加されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置変数)の現在値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。

３．３．２．３　GemGetVName() - 装置変数名取得関数
GemGetVNameA()

（1）呼出書式

[c,C++]

```
API int APIX GemGetVName(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        char   *name,                   // 取得した名前の格納領域ﾎﾟｲﾝﾀ
        int    *asize                   // 名前の文字列長格納用
);

API int APIX GemGetVNameA(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 VID
        char   *name                    // 取得した名前の格納領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetVName (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As String,
        ByRef arraysize As Int32) As Int32

    Function GemGetVName (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As String) As Int32
```

[.NET C#]

```
int GemGetVName(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );

    int GemGetVName(
        int eqid,
        uint vid,
        byte[] val);
```

（2）引数

eqid

GEM 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。

vid

装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

name

取得した名前（文字列）を格納する領域のポインタです。名前格納に充分な領域を準備してください。

asize

取得された名前の文字列長(ﾊﾞｲﾄ)です。

（3）戻り値

戻り値	意味
0	正常に取得できた。 - GemGetVName()の場合
> 0	正常に取得できた。 - GemGetVNameA()の場合は名前のﾊﾞｲﾄ長が返る
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置変数 EC, SV or DV)の名前を取得します。

正常に取得できた場合は、GemGetVName()の戻り値は 0 になり、取得された名前の配列サイズは asize で指定された領域に返却されます。

GemGetVNameA()は取得できた名前の長さが返却されます。

引数 asize が無い場合は、名前のバイト長が返却されます。

3．3．2．4 GemGetVUnits() - 装置変数単位名取得関数
GemGetVUnitsA()

（1）呼出書式

[c,C++]

```
API int APIX GemGetVUnits(
      int   eqid,                  // 通信対象装置ID(0,1,2,...)
      TVID  vid,                   // VID
      char  *units,                // 取得した単位名の格納領域ﾎﾟｲﾝﾀ
      int   *asize                 // 単位名の文字列長格納用
);

API int APIX GemGetVUnitsA(
      int   eqid,                  // 通信対象装置ID(0,1,2,...)
      TVID  vid,                   // VID
      char  *units                 // 取得した単位名の格納領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetVUnits (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As String,
        ByRef arraysize As Int32) As Int32

    Function GemGetVUnits (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As String) As Int32
```

[.NET C#]

```
int GemGetVUnits(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );

    int GemGetVUnits(
        int eqid,
        uint vid,
        byte[] val);
```

（2）引数

eqid

GEM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。

vid

装置変数ID(EC, SV, DVVAL)です。
装置管理情報定義ファイルに登録されている変数のIDでなければなりません。

units

取得した単位名（文字列）を格納する領域のポインタです。名前格納に充分な領域を準備してください。

asize

取得された単位名の文字列長です。もし、単位名が無い場合には =0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。 - GemGetVUnits()の場合
>0	正常に取得できた。 - GemGetVUnitsA()の場合は名前のﾊﾞｲﾄ長が返る
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置変数)の値の単位名を取得します。

正常に取得できた場合は、GemGetVUnits()の戻り値は 0 になり、取得された単位名の配列サイズは asize で指定された領域に返却されます。

GemGetVUnitsA()は取得できた単位名の長さが返却されます。
引数 asize が無い場合は、単位名のバイト長が返却されます。

３．３．２．５　GemGetVFormat() - 装置変数フォーマットコード取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetVFormat(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        int    *fmt                     // 取得ﾌｫｰﾏｯﾄ値格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetVFormat (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByRef fmt As Int32) As Int32
```

[.NET C#]

```
int GemGetVFormat(
        int eqid,
        uint vid,
        ref int fmt );
```

（2）引数

eqid

GEM 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。

vid

装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

fmt

取得したフォーマットの値を格納する領域のポインタです。
返却されるフォーマット値は、DSHDR2 ドライバーが定義するアイテムフォーマットです。

（3）戻り値

戻り値	意味
＞ 0	正常に取得できた。戻り値はﾌｫｰﾏｯﾄの単位ﾊﾞｲﾄ長です。
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置変数)のフォーマット値を取得します。
フォーマット値は、DSHDR2 ドライバーで定義するアイテムフォーマットです。
（dsh.h ファイルでマクロ定義されています。）
正常に取得できた場合は、関数の戻り値は アイテムフォーマットの単位配列サイズ になります。

３．３．２．６　GemGetVArraySize() - 装置変数値の配列サイズ取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetVArraySize(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        int    *asize                   // 取得変数配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetVArraySize (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByRef val As Int32) As Int32
```

[.NET C#]

```
int GemGetVArraySize(
        int eqid,
        uint vid,
        ref int val );
```

（2）引数

eqid

GEM 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。

vid

装置変数 ID です。

装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

asize

取得した配列サイズ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)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置変数)の配列サイズを取得します。

例えば、指定された変数のフォーマットが、FORMAT A[6] の場合は、配列サイズとして=6 が返却されます。

ただし、可変配列文字列変数については最小配列サイズが返却されます。

FORMAT： A[4..16] の場合は =4 が返却されます。

３．３．２．７　GemGetVMin() - 装置変数最小(Min)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetVMin(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ格納用
);
```

[.NET VB]

```
Function GemGetVMin (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetVSizeMinMax(
        int eqid,
        uint vid,
        ref int min,
        ref int max );
```

（2）引数

eqid

GEM 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。

vid

装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。vid のフォーマットと配列サイズ分の値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズを格納します。フォーマット A,J の場合は、文字列長(byte)になります。
最小値が定義されていない場合は、=0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置変数)の最小(Min)値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。ただし、指定された V が最小値を持たない場合は、asize に = 0 が返却されます。

３．３．２．８　GemGetVMax() - 装置変数最大(Max)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetVMax(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ 格納用
);
```

[.NET VB]

```
Function GemGetVMax (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetVMax(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

vid

装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。vid のフォーマットと配列サイズ分の値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。フォーマット A,J の場合は、文字列長になります。
最大値が定義されていない場合は、=0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置変数)の最大(Max)値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。ただし、指定された V が最大値を持たない場合は、asize に = 0 が返却されます。

３．３．２．９　GemGetVNominal() - 装置変数初期(Nominal)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetVNominal(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ 格納用
);
```

[.NET VB]

```
Function GemGetVNominal (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetVNominal(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

vid

装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。vid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。フォーマット A,J の場合は、文字列長になります。
初期値が定義されていない場合は、=0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置変数)の初期値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。ただし、指定された V が初期値を持たない場合は、asize に = 0 が返却されます。

３．３．２．10 GemCheckVVal() - 装置変数値Min,Maxチェック関数

（1）呼出書式

[c,C++]

```
API int APIX GemCheckVVal(
      int   eqid,                  // 通信対象装置ID(0,1,2,...)
      TVID  vid,                   // 変数ID
      void  *val,                  // ﾁｪｯｸ値格納ﾎﾟｲﾝﾀ
      int   asize                  // ﾃﾞｰﾀの配列ｻｲｽﾞ
);
```

[.NET VB]

```
Function GemCheckVVal (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByVal arraysize As Int32) As Int32
```

[.NET C#]

```
int GemCheckVVal(
        int eqid,
        uint vid,
        byte[] val,
        int arraysize );
```

（2）引数

eqid

GEM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。

vid

変数IDです。装置管理情報定義ファイルに登録されている装置変数のIDでなければなりません。

val

チェックするVの値が格納されている領域のポインタです。

asize

チェック値領域の配列サイズです。

（3）戻り値

戻り値	意味
0	正常な値である。
(-1)	vidの値が正しくなかった。
(-2)	値が最小値より小さかった。
(-3)	値が最大値より大きかった。

（4）説明

vidで指定されたV(装置変数)としてのval領域に格納されているデータ値をチェックします。

チェック方法は、valに格納されている値が、当該vidのVのために指定された最小値、最大値の範囲内であるかどうかの判断です。

最小値、最大値がもし指定されていない場合は、指定されていないものとの比較判断は行いません。(正常であるとみなします。)

3．3．2．11 GemGetVList() - 全登録装置変数 ID 取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetVList(
        int         eqid,                   // 通信対象装置 ID(0,1,2,...)
        TBIN_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 GemGetVList (
        ByVal eqid As Int32,
        ByRef plist As IntPtr) As Int32
```

[.NET C#]

```
int GemGetVList(
        int eqid,
        IntPtr plist );
```

（2）引数

eqid

GEM 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 VID が格納されている TBIN_DLIST 構造体のポインタを格納する領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

（4）説明

システムに登録されている全 VID とその名前を TBIN_DLIST 構造体に取出すための関数です。
取出す名前は、装置管理情報定義ファイルで装置変数 定義時に与えられた名前です。
取得した情報の処理が終了した後、DshFreeTBIN_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。

TBIN_DLIST 構造体は次のとおりです。

```
typedef struct{
        int         count;              // 取得できた ID 数
        ULONG       *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
        }TBIN_DLIST;
```

３．３．２．12　GemGetVSizeMinMax() – 装置変数値の最小、最大配列サイズ取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetVSizeMinMax(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // 変数 ID
        int    *min,                    // 最小配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
        int    *max                     // 最大配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetVSizeMinMax (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByRef min As Int32,
        ByRef max As Int32) As Int32
```

[.NET C#]

```
int GemGetVSizeMinMax(
        int eqid,
        uint vid,
        ref int min,
        ref int max );
```

（2）引数

eqid

GEM 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。

vid

装置変数 ID です。
装置管理情報定義ファイルに登録されているデータ変数の ID でなければなりません。

min

取得した最小配列サイズを格納する領域のポインタです。

max

取得した最大配列サイズ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)	vid の値が正しくなかった。

（4）説明

vid で指定された VVAL(装置変数)の最小と最大配列サイズを取得します。
変数の配列サイズが固定の場合は、min, max に同じ値が返却されます。

可変配列文字列変数については最小、最大配列サイズがそれぞれ min,max に返却されます。
FORMAT： A[4..16] の場合は min =4 , max = 16 となります。

３．３．３　EC 装置定数情報アクセスとメッセージ送信関数

３．３．３．１　GemSetEcVal()　- EC 値設定関数

（1）呼出書式

[c,C++]

```
API int APIX GemSetEcVal(
      int   eqid,                    // 通信対象装置 ID(0,1,2,...)
      TECID ecid,                    // ECID
      void  *val,                    // 設定値ﾎﾟｲﾝﾀ
      int   asize                    // ﾃﾞｰﾀの配列ｻｲｽﾞ
);
```

[.NET VB]

```
Function GemSetEcVal (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As IntPtr,
        ByVal arraysize As Int32) As Int32

    Function GemSetEcVal (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByRef val As Int32,
        ByVal arraysize As Int32) As Int32

    Function GemSetEcVal (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As String,
        ByVal arraysize As Int32) As Int32

    Function GemGetEcVal (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemSetEcVal(
        int eqid,
        uint ecid,
        byte[] val,
        int arraysize );
```

（2）引数

eqid

GEM 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。

ecid

装置定数 ID です。
装置管理情報定義ファイルに登録されている定数の ID でなければなりません。

val

設定したい値が格納されている領域のポインタです。ecid のフォーマットに合わせた値を設定してください。

asize

設定値領域の配列サイズです。
フォーマット A,J の場合は、文字列長になり、文字列の最後に NULL 文字(=0)が付加されます。

（3）戻り値

戻り値	意味
0	正常に設定された。
(-1)	ecid の値が正しくなかった。（登録されていなかった）

（4）説明

ecid で指定された EC(装置定数)に新しい値を設定します。戻り値が設定結果になります。
正常に設定できた場合は、関数の戻り値は 0 になります。

３．３．３．２　GemGetEcVal() - EC 値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetEcVal(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TECID ecid,                     // ECID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ格納用
);
```

[.NET VB]

```
Function GemGetEcVal (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetEcVal(
        int eqid,
        uint ecid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

ecid

装置定数 ID です。
装置管理情報定義ファイルに登録されている定数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。ecid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	ecid の値が正しくなかった。

（4）説明

ecid で指定された EC(装置定数)の現在値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。

３．３．３．３ GemGetEcName() - EC名取得関数
GemGetEcNameA()

（1）呼出書式

[c,C++]

```
API int APIX GemGetEcName(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TECID ecid,                     // ECID
        char  *name,                    // 取得した名前の格納領域ﾎﾟｲﾝﾀ
        int    *asize                   // 名前の文字列長格納用
);

API int APIX GemGetEcNameA(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TECID ecid,                     // ECID
        char  *name                     // 取得した名前の格納領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetEcName (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As String,
        ByRef arraysize As Int32) As Int32

    Function GemGetEcName (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As String) As Int32
```

[.NET C#]

```
int GemGetEcName(
        int eqid,
        uint ecid,
        byte[] val,
        ref int arraysize );

    int GemGetEcName(
        int eqid,
        uint ecid,
        byte[] val);
```

（2）引数

eqid

GEM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。

ecid

装置定数IDです。
装置管理情報定義ファイルに登録されている定数のIDでなければなりません。

name

取得した名前（文字列）を格納する領域のポインタです。名前格納に充分な領域を準備してください。

asize

取得された名前の文字列長です。

（3）戻り値

戻り値	意味
0	正常に取得できた。 - GemGetEcName()の場合
> 0	正常に取得できた。 - GemGetEcNameA()の場合は名前のﾊﾞｲﾄ長が返る
(-1)	ecid の値が正しくなかった。

（4）説明

ecid で指定された EC(装置定数)の名前を取得します。

正常に取得できた場合は、GemGetEcName()の戻り値は 0 になり、取得された名前の配列サイズは asize で指定された領域に返却されます。

GemGetEcNameA()は取得できた名前の長さが返却されます。
引数 asize が無い場合は、名前のバイト長が返却されます。

３．３．３．４　GemGetEcUnits() - EC 単位名取得関数
GemGetEcUnitsA()

（1）呼出書式

[c,C++]

```
API int APIX GemGetEcUnits(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TECID ecid,                     // ECID
        char  *units,                   // 取得した単位名の格納領域ﾎﾟｲﾝﾀ
        int    *asize                   // 単位名の文字列長格納用
);

API int APIX GemGetEcUnitsA(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TECID ecid,                     // ECID
        char  *units                    // 取得した単位名の格納領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetEcUnits (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As String,
        ByRef arraysize As Int32) As Int32

Function GemGetEcUnits (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As String) As Int32
```

[.NET C#]

```
int GemGetEcUnits(
        int eqid,
        uint ecid,
        byte[] val,
        ref int arraysize );

    int GemGetEcUnits(
        int eqid,
        uint ecid,
        byte[] val);
```

（2）引数

eqid

GEM 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。

ecid

装置定数 ID です。
装置管理情報定義ファイルに登録されている定数の ID でなければなりません。

units

取得した単位名（文字列）を格納する領域のポインタです。名前格納に充分な領域を準備してください。

asize

取得された単位名の文字列長です。もし、単位名が無い場合には =0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。 - GemGetEcUnits()の場合
>0	正常に取得できた。 - GemGetEcUnitsA()の場合は名前のﾊﾞｲﾄ長が返る
(-1)	vid の値が正しくなかった。

（4）説明

ecid で指定された EC(装置定数)の値の単位名を取得します。

正常に取得できた場合は、GemGetEcUnits()の戻り値は 0 になり、取得された単位名の配列サイズは asize で指定された領域に返却されます。

GemGetEcUnitsA()は取得できた単位名の長さが返却されます。
引数 asize が無い場合は、単位名のバイト長が返却されます。

３．３．３．５　GemGetEcFormat() - EC フォーマットコード取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetEcFormat(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TECID ecid,                     // ECID
        int    *fmt                     // 取得ﾌｫｰﾏｯﾄ値格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetEcFormat (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByRef fmt As Int32) As Int32
```

[.NET C#]

```
int GemGetEcFormat(
        int eqid,
        uint ecid,
        ref int fmt );
```

（2）引数

eqid

GEM 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。

ecid

装置定数 ID です。
装置管理情報定義ファイルに登録されている定数の ID でなければなりません。

fmt

取得したフォーマットの値を格納する領域のポインタです。
返却されるフォーマット値は、DSHDR2 ドライバーが定義するアイテムフォーマットです。

（3）戻り値

戻り値	意味
> 0	正常に取得できた。戻り値はﾌｫｰﾏｯﾄの単位ﾊﾞｲﾄ長です。
(-1)	ecid の値が正しくなかった。

（4）説明

ecid で指定された EC(装置定数)のフォーマット値を取得します。
フォーマット値は、DSHDR2 ドライバーで定義するアイテムフォーマットです。
（dsh.h ファイルでマクロ定義されています。）
正常に取得できた場合は、関数の戻り値は アイテムフォーマットの単位配列サイズ になります。

３．３．３．６　GemGetEcArraySize() - EC 値の配列サイズ取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetEcArraySize(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TECID ecid,                     // ECID
        int    *asize                   // 取得配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetEcArraySize (
          ByVal eqid As Int32,
          ByVal ecid As Int32,
          ByRef val As Int32) As Int32
```

[.NET C#]

```
int GemGetEcArraySize(
          int eqid,
          uint ecid,
          ref int val );
```

（2）引数

eqid

GEM 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。

ecid

装置定数 ID です。

装置管理情報定義ファイルに登録されている定数の ID でなければなりません。

asize

取得した配列サイズ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)	ecid の値が正しくなかった。

（4）説明

ecid で指定された EC(装置定数)の配列サイズを取得します。

例えば、指定された EC が、EC_mdln A が FORMAT: A[6] の場合は、配列サイズとして=6 が返却されます。

ただし、可変配列文字列変数については最小配列サイズが返却されます。

FORMAT： A[4..16] の場合は =4 が返却されます。

３．３．３．７　GemGetEcMin() - EC最小(Min)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetEcMin(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TECID ecid,                     // ECID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ格納用
);
```

[.NET VB]

```
Function GemGetEcMin (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetEcMin(
        int eqid,
        uint ecid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM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。

ecid

装置定数IDです。
装置管理情報定義ファイルに登録されている定数のIDでなければなりません。

val

取得した値を格納する領域のポインタです。ecidのフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズはasizeに格納されます。

asize

取得された値の配列サイズ格納用です。フォーマットA,Jの場合は、文字列長になります。
最小値が定義されていない場合は、=0が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	ecidの値が正しくなかった。

（4）説明

ecidで指定されたEC(装置定数)の最小(Min)値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asizeで指定された領域に返却されます。ただし、指定されたECが最小値を持たない場合は、asizeに = 0が返却されます。

３．３．３．８　GemGetEcMax() - EC 最大(Max)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetEcMax(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TECID ecid,                     // ECID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ格納用
);
```

[.NET VB]

```
Function GemGetEcMax (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetEcMax(
        int eqid,
        uint ecid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

ecid

装置定数 ID です。
装置管理情報定義ファイルに登録されている定数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。ecid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。フォーマット A,J の場合は、文字列長になります。
最大値が定義されていない場合は、=0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	ecid の値が正しくなかった。

（4）説明

ecid で指定された EC(装置定数)の最大(Max)値を取得します。正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。
ただし、指定された EC が最大値を持たない場合は、asize に = 0 が返却されます。

３．３．３．９　GemGetEcNominal() - EC 初期(Nominal)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetEcNominal(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TECID ecid,                     // ECID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ 格納用
);
```

[.NET VB]

```
Function GemGetEcNominal (
        ByVal eqid As Int32,
        ByVal ecid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetEcNominal(
        int eqid,
        uint ecid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

ecid

装置定数 ID です。装置管理情報定義ファイルに登録されている定数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。ecid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。フォーマット A,J の場合は、文字列長になります。
初期値が定義されていない場合は、=0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	ecid の値が正しくなかった。

（4）説明

ecid で指定された EC(装置定数)の初期値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。ただし、指定された EC が初期値を持たない場合は、asize に = 0 が返却されます。

３．３．３．10　GemCheckEcVal() - EC 値 MinMax チェック関数

（1）呼出書式

[c,C++]

```
API int APIX GemCheckEcVal(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TECID ecid,                     // ECID
        void   *val,                    // ﾁｪｯｸ値格納ﾎﾟｲﾝﾀ
        int    asize                    // ﾃﾞｰﾀの配列ｻｲｽﾞ
);
```

[.NET VB]

```
Function GemCheckEcVal (
          ByVal eqid As Int32,
          ByVal ecid As Int32,
          ByVal val As IntPtr,
          ByVal arraysize As Int32) As Int32
```

[.NET C#]

```
int GemCheckEcVal(
          int eqid,
          uint ecid,
          byte[] val,
          int arraysize );
```

（2）引数

eqid

GEM 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。

ecid

装置定数 ID です。装置管理情報定義ファイルに登録されている定数の ID でなければなりません。

val

チェックする EC の値が格納されている領域のポインタです。

asize

チェック値領域の配列サイズです。

（3）戻り値

戻り値	意味
0	正常な値である。
(-1)	ecid の値が正しくなかった。
(-2)	値が最小値より小さかった。
(-3)	値が最大値より大きかった。

（4）説明

ecid で指定された EC(装置定数)としての val 領域に格納されているデータ値をチェックします。
チェック方法は、val に格納されている値が、当該 ecid の EC のために指定された最小値、最大値の範囲内であるかどうかの判断です。最小値、最大値がもし指定されていない場合は、指定されていないものとの比較判断は行いません。（正常であるとみなします。）

３．３．３．11　GemGetEcList() - 全登録 ECID 取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetEcList(
      int    eqid,                    // 通信対象装置 ID(0,1,2,...)
      TBIN_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 GemGetEcList (
        ByVal eqid As Int32,
        ByRef plist As IntPtr) As Int32
```

[.NET C#]

```
int GemGetEcList(
        int eqid,
        IntPtr plist );
```

（2）引数

eqid

GEM 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 ECID が格納されている TBIN_DLIST 構造体のポインタを格納する領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

（4）説明

システムに登録されている全 ECID とその名前を TBIN_DLIST 構造体に取出すための関数です。
取出す名前は、装置管理情報定義ファイルで EC 定義時に与えられた名前です。
取得した情報の処理が終了した後、DshFreeTBIN_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。

TBIN_DLIST 構造体は次のとおりです。

```
typedef struct{
        int         count;              // 取得できた ID 数
        ULONG       *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
        }TBIN_DLIST;
```

３．３．３．12 GemGetEcSizeMinMax() - EC 値の最小、最大配列サイズ取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetEcArraySize(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TECID ecid,                     // ECID
        int    *min,                    // 最小配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
        int    *max                     // 最大配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function EngGetEcSizeMinMax (
          ByVal eqid As Int32,
          ByVal ecid As Int32,
          ByRef min As Int32,
          ByRef max As Int32) As Int32
```

[.NET C#]

```
int EngGetEcizeMinMax(
          int eqid,
          uint ecid,
          ref int min,
          ref int max );
```

（2）引数

eqid

GEM 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。

ecid

装置定数 ID です。装置管理情報定義ファイルに登録されている定数の ID でなければなりません。

min

取得した最小配列サイズを格納する領域のポインタです。

max

取得した最大配列サイズ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)	ecid の値が正しくなかった。

（4）説明

ecid で指定された EC(装置定数)の最小と最大配列サイズを取得します。
変数の配列サイズが固定の場合は、min, max に同じ値が返却されます。

可変配列文字列変数については最小、最大配列サイズがそれぞれ min,max に返却されます。
FORMAT： A[4..16] の場合は min =4 , max = 16 となります。

３．３．３．13 GemSendS2F13() - 装置定数要求メッセージ送信

（1）呼出書式

[c,C++]

```
API int APIX GemSendS2F13(
        int  eqid,                          // 通信対象装置 ID(0,1,2,...)
        TVID *ecid_list,                    // 要求する ECID のﾘｽﾄ領域のﾎﾟｲﾝﾀ
        int  count,                         // ecid_list に含まれる ECID の数
        TV_VALUE_LIST *val_list,            // S2F14 で返された定数値情報格納用ﾘｽﾄﾎﾟｲﾝﾀ
        int (WINAPI *s2f13Callback)(),      // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                         // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS2F13 (
          ByVal eqid As Int32,
          ByRef ecid_list As Int32,
          ByVal count As Int32,
          ByRef val_list As dsh_info.TV_VALUE_LIST,
          ByVal callback As vcallback.callback_S2F13,
          ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS2F13(
          int eqid,
          ref uint ecid_list,
          int count,
          ref TV_VALUE_LIST val_list,
          CallbackS2F13 callback,
          uint upara );
```

（2）引数

eqid

GEM 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。

ecid_list

装置に要求したい装置定数 ID が格納されているリストのポインタです。

count

ecid_list 内に含まれる装置定数 ID の数です。
count=0 の場合は、全装置定数が対象になります。

val_list

ecid_list 内に指定された装置定数 ID の変数値情報を格納するためのリストへのポインタです。
変数値情報にはデータのフォーマット、配列サイズも与えられます。

s2f13Callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
このコールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバ

ックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。 val_list に受信した定数値情報が格納されています。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。（応答ﾒｯｾｰｼﾞが得られなかった）

（4）説明

装置に対し、ecid_list 内に指定された count 分の装置定数 ID が有する定数値を S2F13 メッセージを使って要求します。
正常に応答メッセージを受信できた場合は、そのメッセージをデコードし、val_list で指定された構造体リストに格納します。
また、count＝0 の場合は、予め GemSetEcidList()関数を使って、装置から S2F14 内に並べ与えられる ECID を DSHGEM-LIB に設定しておく必要があります。

送信要求から S2F13 応答メッセージ受信までの制御は引数の S2F13 コールバック関数指定の有無によって次のようになります。

s2f13Callback の指定	制御の流れ
なし (=0)	S2F13 送信後、応答ﾒｯｾｰｼﾞ S2F14 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、val_list に S2F14 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。
あり	送信要求後、S2F13 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で引数 val_list に S2F14 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

DSHGEM-LIB は S2F14 で得られた情報の装置管理情報としての更新は行いません。必要に応じてユーザ側プログラムで更新のための処理をしてください。

正常に応答メッセージを受信した場合、val_list 構造体内に定数値情報が渡されますが、ユーザ側で情報の処理を終えた後、その構造体に使用されているメモリを解放してください。

```
DshFreeTVAL_LIST(val_list);
```

（5）コールバック関数

[c,C++]

```
API int APIX s2f13Callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TVAL_LIST *val_list,            // EC 値が格納されている構造体のﾎﾟｲﾝﾀ
        ULONG upara                     // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S2F13(ByVal eqid As Integer, ByVal end_status As Integer, ByRef val_list As
dsh_info.TV_VALUE_LIST, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS2F13(int eqid, int end_status, ref TV_VALUE_LIST vlist, uint upara);
```

end_status には以下の値が設定されます。

結果	意味
0	正常に送受信できた。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

val_list は end_status=0 の時にだけ有効です。

３．３．３．14 GemSendS2F15() - 装置定数変更メッセージ送信

（1）呼出書式

[c,C++]

```
API int APIX GemSendS2F15(
        int   eqid,                         // 通信対象装置 ID(0,1,2,...)
        TV_VALUE_LIST *val_list,            // s1f16 で返された定数値設定情報格納用ﾘｽﾄﾎﾟｲﾝﾀ
        int   *eac,                         // 装置からの応答 ack が格納される領域のﾎﾟｲﾝﾀ
        int (WINAPI *s2f15Callback)(),      // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                         // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS2F15 (
        ByVal eqid As Int32,
        ByRef ecid_list As dsh_info.TV_VALUE_LIST,
        ByRef eac As Int32,
        ByVal callback As vcallback.callback_S2F15,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS2F15(
        int eqid,
        ref TV_VALUE_LIST ecid_list,
        ref int eac,
        CallbackAck callback,
        uint upara );
```

（2）引数

eqid

GEM 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。

val_list

val_list 内に定数の数、定数 ID そして変更したい値などが格納されている EC 情報リストのポインタです。

eac

受信した S2F16 に含まれる ack の値を格納するためのポインタです。

s2f15Callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。ユーザは任意の関数名を指定できます。
このコールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。 val_list に受信した定数値設定情報が格納されています。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。(応答ﾒｯｾｰｼﾞが得られなかった)

（4）説明

装置に対し、val_list 内に指定された count 分の定数 ID、設定したい定数値を S2F15 メッセージを使って値の変更要求をします。

val_list に定数情報を設定する際、以下のライブラリ関数を使用することができます。
①DshInitTV_VALUE_LIST()
②DshAddTV_VALUE_LIST()

装置に受け入れられたかどうかを eac に返却します。eac の値が=0 で正常に変更されたことを意味します。

送信要求から S2F15 の応答メッセージ受信までの制御は引数の S2F15 コールバック関数指定の有無によって次のようになります。

s2f15Callback の指定	制御の流れ
なし (=0)	S2F15 送信後、応答ﾒｯｾｰｼﾞ S2F16 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、val_list に S2F16 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。
あり	送信要求後、S2F15 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で引数 val_list に S2F16 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

正常に応答メッセージを受信した場合、eac に ack の値が渡されます。

（5）コールバック関数

[c,C++]

```
API int APIX s2f15Callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        int    *eac,                    // S2F16 の Ack が格納されているﾎﾟｲﾝﾀ
        ULONG upara                     // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S2F15(ByVal eqid As Integer, ByVal end_status As Integer, ByRef eac As Integer,
ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackAck(int eqid, int end_status, int *eac, uint upara);
```

end_status には以下の値が設定されます。

結果	意味
0	正常に送受信できた。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

end_status = 0 のときのみ、eac の内容が有効です。

３．３．３．15 GemSendS2F29() - 装置定数名一覧要求メッセージ送信

（1）呼出書式

[c,C++]

```
API int APIX GemSendS2F29(
        int  eqid,                          // 通信対象装置 ID(0,1,2,...)
        TVID *ecid_list,                    // 要求する ECID のﾘｽﾄ領域のﾎﾟｲﾝﾀ
        int  count,                         // ecid_list に含まれる ECID の数
        TEC_NAME_LIST *name_list,           // s2f30 で返された定数名一覧情報格納用ﾘｽﾄﾎﾟｲﾝﾀ
        int (WINAPI *s2f29Callback)(),      // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                         // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS2F29 (
        ByVal eqid As Int32,
        ByRef ecid_list As Int32,
        ByVal count As Int32,
        ByRef name_list As dsh_info.TEC_NAME_LIST,
        ByVal callback As vcallback.callback_S2F29,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS2F29(
        int eqid,
        ref uint ecid_list,
        int count,
        ref TEC_NAME_LIST name_list,
        CallbackS2F29 callback,
        uint upara );
```

（2）引数

eqid

GEM 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。

ecid_list

装置に要求したい定数名一覧の定数 ID が格納されているリストのポインタです。

count

ecid_list 内に含まれる装置定数 ID の数です。
count=0 の場合は、全定数が対象になります。

name_list

ecid_list 内に指定された定数 ID の変数名一覧情報を格納するためのリストへのポインタです。
定数名一覧情報には変数名、フォーマット、最小、最大、ﾃﾞﾌｫﾙﾄ値ならびに値の単位名が与えられます。

s2f29Callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
このコールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。 name_list に受信した定数名一覧情報が格納されています。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。（応答ﾒｯｾｰｼﾞが得られなかった）

（4）説明

装置に対し、ecid_list 内に指定された count 分の定数 ID の定数名一覧を S2F29 メッセージを使って要求します。

正常に応答メッセージを受信できた場合は、そのメッセージをデコードし、name_list で指定された構造体リストに格納します。

また、count＝0 の場合は、予め GemSetEcidList()関数を使って、装置から S2F30 内に並べ与えられる ECID を DSHGEM-LIB に設定しておく必要があります。

送信要求から S2F29 応答メッセージ受信までの制御は引数の S2F29 コールバック関数指定の有無によって次のようになります。

s2f29Callback の指定	制御の流れ
なし (=0)	S2F29 送信後、応答ﾒｯｾｰｼﾞ S2F30 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、name_list に S2F30 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。
あり	送信要求後、S2F29 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で引数 name_list に S2F30 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

正常に応答メッセージを受信した場合、name_list 構造体内に定数名一覧情報が渡されますが、ユーザ側で情報の処理を終えた後、その構造体に使用されているメモリを解放してください。

```
DshFreeTEC_NAME_LIST(name_list);
```

（5）コールバック関数

[c,C++]

```
API int APIX s2f29Callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TEC_NAME_LIST *name_list,       // EC 名一覧リストが格納されている構造体のﾎﾟｲﾝﾀ
        ULONG upara                     // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

Function callback_S2F29(ByVal eqid As Integer, ByVal end_status As Integer, ByRef name_list As dsh_info.TEC_NAME_LIST, ByVal upara As Integer) As Integer

[.NET C#]

int CallbackS2F29(int eqid, int end_status, ref TEC_NAME_LIST name_list, uint upara);

end_status には以下の値が設定されます。

結果	意味
0	正常に送受信できた。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

end_status = 0 のときのみ name_list の内容が有効です。

３．３．４　SV 装置状態変数情報アクセスとメッセージ送信関数

３．３．４．１　GemSetSvVal()　- SV 値設定関数

（1）呼出書式

[c,C++]

```
API int APIX GemSetSvVal(
      int   eqid,                  // 通信対象装置 ID(0,1,2,...)
      TSVID svid,                  // SVID
      void  *val,                  // 設定値ﾎﾟｲﾝﾀ
      int   asize                  // ﾃﾞｰﾀの配列ｻｲｽﾞ
);
```

[.NET VB]

```
Function GemSetSvVal (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As IntPtr,
        ByVal arraysize As Int32) As Int32

    Function GemSetSvVal (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As String,
        ByVal arraysize As Int32) As Int32

    Function GemSetSvVal (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByRef val As Integer,
        ByVal arraysize As Int32) As Int32

    Function GemGetSvVal (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemSetSvVal(
        int eqid,
        uint svid,
        byte[] val,
        int arraysize );
```

（2）引数

eqid

GEM 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。

svid

装置状態変数 ID です。

装置管理情報定義ファイルに登録されている状態変数の ID でなければなりません。

val

設定したい値の格納ポインタです。svid のフォーマットに合わせた値を設定してください。

asize

設定値領域の配列サイズです。

フォーマット A,J の場合は、文字列長になります。

（3）戻り値

戻り値	意味
0	正常に設定された。
(-1)	svid の値が正しくなかった。（登録されていなかった）

（4）説明

svid で指定された SV(装置状態変数)に新しい値を設定します。戻り値が設定結果になります。

正常に設定できた場合は、関数の戻り値は 0 になります。

３．３．４．２　GemGetSvVal() - SV 値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetSvVal(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TSVID svid,                     // SVID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ 格納用
);
```

[.NET VB]

```
Function GemGetSvVal (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetSvVal(
        int eqid,
        uint svid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

svid

装置状態変数 ID です。
装置管理情報定義ファイルに登録されている状態変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。svid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。
フォーマット A,J の場合は、文字列長になり、最後に NULL 文字(=0) が付加されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	svid の値が正しくなかった。

（4）説明

svid で指定された SV(装置状態変数)の現在値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。

３．３．４．３　GemGetSvName() - SV名取得関数
GemGetSvNameA()

（1）呼出書式

[c,C++]

```
API int APIX GemGetSvName(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TSVID svid,                     // SVID
        char   *name,                   // 取得した名前の格納領域ﾎﾟｲﾝﾀ
        int    *asize                   // 名前の文字列長格納用
);

API int APIX GemGetSvNameA(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TECID svid,                     // SVID
        char   *name                    // 取得した名前の格納領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetSvName (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As String,
        ByRef arraysize As Int32) As Int32

    Function GemGetSvName (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As String) As Int32
```

[.NET C#]

```
int GemGetSvName(
        int eqid,
        uint svid,
        byte[] val,
        ref int arraysize );

    int GemGetSvName(
        int eqid,
        uint svid,
        byte[] val );
```

（2）引数

eqid

GEM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。

svid

装置状態変数IDです。
装置管理情報定義ファイルに登録されている状態変数のIDでなければなりません。

name

取得した名前（文字列）を格納する領域のポインタです。名前格納に充分な領域を準備してください。

asize

取得された名前の文字列長です。

（3）戻り値

戻り値	意味
0	正常に取得できた。 - GemGetSvName()の場合
> 0	正常に取得できた。 - GemGetSvnameA()の場合は名前のﾊﾞｲﾄ長が返る
(-1)	ecid の値が正しくなかった。

（4）説明

svid で指定された SV(装置状態変数)の名前を取得します。

正常に取得できた場合は、GemGetSvName()の戻り値は 0 になり、取得された名前の配列サイズは asize で指定された領域に返却されます。

GemGetSvNameA()は取得できた名前の長さが返却されます。
引数 asize が無い場合は、名前のバイト長が返却されます。

３．３．４．４　GemGetSvUnits() - SV 単位名取得関数
GemGetSvUnitsA()

（1）呼出書式

[c,C++]

```
API int APIX GemGetSvUnits(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TSVID svid,                     // SVID
        char  *units,                   // 取得した単位名の格納領域ﾎﾟｲﾝﾀ
        int   *asize                    // 単位名の文字列長格納用
);

API int APIX GemGetSvUnitsA(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TSVID svid,                     // SVID
        char  *units                    // 取得した単位名の格納領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetSvUnits (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As String,
        ByRef arraysize As Int32) As Int32

    Function GemGetSvUnits (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As String) As Int32
```

[.NET C#]

```
int GemGetSvUnits(
        int eqid,
        uint svid,
        byte[] val,
        ref int arraysize );

    int GemGetSvUnits(
        int eqid,
        uint svid,
        byte[] val);
```

（2）引数

eqid

GEM 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。

svid

装置状態変数 ID です。
装置管理情報定義ファイルに登録されている状態変数の ID でなければなりません。

units

取得した単位名（文字列）を格納する領域のポインタです。名前格納に充分な領域を準備してください。

asize

取得された単位名の文字列長です。もし、単位名が無い場合には =0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。 - GemGetSvUnits()の場合
>0	正常に取得できた。 - GemGetSvUnitsA()の場合は名前のﾊﾞｲﾄ長が返る
(-1)	vid の値が正しくなかった。

（4）説明

svid で指定された SV(装置状態変数)の値の単位名を取得します。

正常に取得できた場合は、GemGetSvUnits()の戻り値は 0 になり、取得された単位名の配列サイズは asize で指定された領域に返却されます。

GemGetSvUnitsA()は取得できた単位名の長さが返却されます。
引数 asize が無い場合は、単位名のバイト長が返却されます。

３．３．４．５　GemGetSvFormat() - SV フォーマットコード取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetSvFormat(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TSVID svid,                     // SVID
        int    *fmt                     // 取得ﾌｫｰﾏｯﾄ値格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetSvFormat (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByRef fmt As Int32) As Int32
```

[.NET C#]

```
int GemGetSvFormat(
        int eqid,
        uint svid,
        ref int fmt );
```

（2）引数

eqid

GEM 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。

svid

装置状態変数 ID です。
装置管理情報定義ファイルに登録されている状態変数の ID でなければなりません。

fmt

取得したフォーマットの値を格納する領域のポインタです。
返却されるフォーマット値は、DSHDR2 ドライバーが定義するアイテムフォーマットです。

（3）戻り値

戻り値	意味
> 0	正常に取得できた。戻り値はﾌｫｰﾏｯﾄの単位ﾊﾞｲﾄ長です。
(-1)	svid の値が正しくなかった。

（4）説明

svid で指定された SV(装置状態変数)のフォーマット値を取得します。
フォーマット値は、DSHDR2 ドライバーで定義するアイテムフォーマットです。
（dsh.h ファイルでマクロ定義されています。）
正常に取得できた場合は、関数の戻り値は アイテムフォーマットの単位配列サイズ になります。

３．３．４．６　GemGetSvArraySize() - SV 値の配列サイズ取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetSvArraySize(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TSVID svid,                     // SVID
        int    *asize                   // 取得配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetSvArraySize (
          ByVal eqid As Int32,
          ByVal svid As Int32,
          ByRef val As Int32) As Int32
```

[.NET C#]

```
int GemGetSvArraySize(
          int eqid,
          uint svid,
          ref int val );
```

（2）引数

eqid

GEM 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。

svid

装置状態変数 ID です。

装置管理情報定義ファイルに登録されている状態変数の ID でなければなりません。

asize

取得した配列サイズ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)	svid の値が正しくなかった。

（4）説明

svid で指定された SV(装置状態変数)の配列サイズを取得します。

例えば、指定された SV が、SV_mdln が FORMAT A[6] の場合は、配列サイズとして=6 が返却されます。

ただし、可変配列文字列変数については最小配列サイズが返却されます。

FORMAT： A[4..16] の場合は =4 が返却されます。

３．３．４．７　GemGetSvMin() - SV最小(Min)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetSvMin(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TSVID svid,                     // SVID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ格納用
);
```

[.NET VB]

```
Function GemGetSvMin (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetSvMin(
        int eqid,
        uint svid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM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。

svid

装置状態変数IDです。装置管理情報定義ファイルに登録されている状態変数のIDでなければなりません。

val

取得した値を格納する領域のポインタです。svidのフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズはasizeに格納されます。

asize

取得された値の配列サイズ格納用です。フォーマットA,Jの場合は、文字列長になります。
最小値が定義されていない場合は、=0が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	svidの値が正しくなかった。

（4）説明

svidで指定されたSV(装置状態変数)の最小(Min)値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asizeで指定された領域に返却されます。ただし、指定されたSVが最小値を持たない場合は、asizeに = 0が返却されます。

３．３．４．８　GemGetSvMax() - SV 最大(Max)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetSvMax(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TSVID svid,                     // SVID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ格納用
);
```

[.NET VB]

```
Function GemGetSvMax (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetSvMax(
        int eqid,
        uint svid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

svid

装置状態変数 ID です。装置管理情報定義ファイルに登録されている状態変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。svid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。フォーマット A,J の場合は、文字列長になります。
最大値が定義されていない場合は、=0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	svid の値が正しくなかった。

（4）説明

svid で指定された SV(装置状態変数)の最大(Max)値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。ただし、指定された SV が最大値を持たない場合は、asize に = 0 が返却されます。

３．３．４．９　GemGetSvNominal() - SV 初期(Nominal)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetSvNominal(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TSVID svid,                     // SVID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ 格納用
);
```

[.NET VB]

```
Function GemGetSvNominal (
        ByVal eqid As Int32,
        ByVal svid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetSvNominal(
        int eqid,
        uint svid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

svid

装置状態変数 ID です。装置管理情報定義ファイルに登録されている状態変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。svid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。フォーマット A,J の場合は、文字列長になります。
初期値が定義されていない場合は、=0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	svid の値が正しくなかった。

（4）説明

svid で指定された SV(装置状態変数)の初期値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。ただし、指定された SV が初期値を持たない場合は、asize に = 0 が返却されます。

３．３．４．10　GemCheckSvVal() - SV値MinMaxチェック関数

（1）呼出書式

[c,C++]

```
API int APIX GemCheckSvVal(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TSVID svid,                     // SVID
        void   *val,                    // ﾁｪｯｸ値格納ﾎﾟｲﾝﾀ
        int    asize                    // ﾃﾞｰﾀの配列ｻｲｽﾞ格納用
);
```

[.NET VB]

```
Function GemCheckSvVal (
          ByVal eqid As Int32,
          ByVal svid As Int32,
          ByVal val As IntPtr,
          ByVal arraysize As Int32) As Int32
```

[.NET C#]

```
int GemCheckSvVal(
          int eqid,
          uint svid,
          byte[] val,
          int arraysize );
```

（2）引数

eqid

GEM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。

svid

装置状態変数IDです。

装置管理情報定義ファイルに登録されている状態変数のIDでなければなりません。

val

チェックするSVの値が格納されている領域のポインタです。

asize

チェック値領域の配列サイズです。

（3）戻り値

戻り値	意味
0	正常な値である。
(-1)	svidの値が正しくなかった。
(-2)	値が最小値より小さかった。
(-3)	値が最大値より大きかった。

（4）説明

svidで指定されたSV(装置状態変数)としてのval領域に格納されているデータ値をチェックします。

チェック方法は、valに格納されている値が、当該svidのSVのために指定された最小値、最大値の範囲内であるかどうかの判断です。最小値、最大値がもし指定されていない場合は、指定されていないものとの比較判断は行いません。（正常であるとみなします。）

３．３．４．11　GemGetSvList() - 全登録 SVID 取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetSvList(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TBIN_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 GemGetSvList (
        ByVal eqid As Int32,
        ByRef list As IntPtr) As Int32
```

[.NET C#]

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

（2）引数

eqid

GEM 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 SVID が格納されている TBIN_DLIST 構造体のポインタを格納する領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

（4）説明

システムに登録されている全 SVID とその名前を TBIN_DLIST 構造体に取出すための関数です。
取出す名前は、装置管理情報定義ファイルで SV 定義時に与えられた名前です。
取得した情報の処理が終了した後、DshFreeTBIN_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。

TBIN_DLIST 構造体は次のとおりです。

```
typedef struct{
        int         count;              // 取得できた ID 数
        ULONG       *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
        }TBIN_DLIST;
```

３．３．４．12　GemGetSvSizeMinMax() - SV 値の最小、最大配列サイズ取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetSvSizeMinMax(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TSVID svid,                     // SVID
        int    *min,                    // 最小配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
        int    *max                     // 最大配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetSvSizeMinMax (
          ByVal eqid As Int32,
          ByVal svid As Int32,
          ByRef min As Int32,
          ByRef max As Int32) As Int32
```

[.NET C#]

```
int GemGetSvSizeMinMax(
          int eqid,
          uint svid,
          ref int min,
          ref int max );
```

（2）引数

eqid

GEM 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。

svid

装置状態変数 ID です。

装置管理情報定義ファイルに登録されている状態変数の ID でなければなりません。

min

取得した最小配列サイズを格納する領域のポインタです。

max

取得した最大配列サイズ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)	svid の値が正しくなかった。

（4）説明

svid で指定された SV(装置状態変数)の最小と最大配列サイズを取得します。

変数の配列サイズが固定の場合は、min, max に同じ値が返却されます。

可変配列文字列変数については最小、最大配列サイズがそれぞれ min,max に返却されます。

FORMAT： A[4..16] の場合は min =4 , max = 16 となります。

３．３．４．13　GemSendS1F3() - 装置状態変数要求メッセージ送信

（1）呼出書式

[c,C++]

```
API int APIX GemSendS1F3(
        int  eqid,                          // 通信対象装置 ID(0,1,2,...)
        TVID *svid_list,                    // 要求する SVID のﾘｽﾄ領域のﾎﾟｲﾝﾀ
        int  count,                         // svid_list に含まれる SVID の数
        TV_VALUE_LIST *val_list,            // S1F4 で返された状態変数値情報格納用ﾘｽﾄﾎﾟｲﾝﾀ
        int (WINAPI *s1f3Callback)(),       // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                         // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS1F3 (
        ByVal eqid As Int32,
        ByRef svid_list As Int32,
        ByVal count As Int32,
        ByRef val_list As dsh_info.TV_VALUE_LIST,
        ByVal callback As vcallback.callback_S1F3,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS1F3(
        int eqid,
        ref uint svid_list,
        int count,
        ref TV_VALUE_LIST val_list,
        CallbackS1F3 callback,
        uint upara );
```

（2）引数

eqid

GEM 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。

svid_list

装置に要求したい装置状態変数 ID が格納されているリストのポインタです。

count

svid_list 内に含まれる装置状態変数 ID の数です。
count=0 の場合は、全装置状態変数が対象になります。

val_list

svid_list 内に指定された装置状態変数 ID の変数値情報を格納するためのリストへのポインタです。
変数値情報にはデータのフォーマット、配列サイズも与えられます。

s1f3Callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
このコールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバ

ックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。 val_list に受信した状態変数値情報が格納されています。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。（応答ﾒｯｾｰｼﾞが得られなかった）

（4）説明

装置に対し、svid_list 内に指定された count 分の装置状態変数 ID が有する状態変数値を S1F3 メッセージを使って要求します。
正常に応答メッセージを受信できた場合は、そのメッセージをデコードし、val_list で指定された構造体リストに格納します。
また、count＝0 の場合は、予め GemSetSvidList()関数を使って、装置から S1F4 内に並べ与えられる SVID を DSHGEM-LIB に設定しておく必要があります。

送信要求から S1F3 応答メッセージ受信までの制御は引数の S1F3 コールバック関数指定の有無によって次のようになります。

s1f3Callback の指定	制御の流れ
なし (=0)	S1F3 送信後、応答ﾒｯｾｰｼﾞ S1F4 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、val_list に S1F4 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。
あり	送信要求後、S1F3 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で引数 val_list に S1F4 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

DSHGEM-LIB は S1F4 で得られた情報の装置管理情報としての更新は行いません。必要に応じてユーザ側プログラムで更新のための処理をしてください。

正常に応答メッセージを受信した場合、val_list 構造体内に状態変数値情報が渡されますが、ユーザ側で情報の処理を終えた後、その構造体に使用されているメモリを解放してください。

```
DshFreeTVAL_LIST(val_list);
```

（5）コールバック関数

[c,C++]

```
API int APIX s1f3Callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TVAL_LIST *val_list,            // SV 値が格納されている構造体のﾎﾟｲﾝﾀ
        ULONG upara                     // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function callback_S1F3(ByVal eqid As Integer, ByVal end_status As Integer, ByRef val_list As
dsh_info.TV_VALUE_LIST, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS1F3(int eqid, int end_status, ref TV_VALUE_LIST vlist, uint upara);
```

end_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

end_status = 0 のときのみ vlist の内容が有効です。

３．３．４．14 GemSendS1F11() - 装置状態変数名一覧要求メッセージ送信

（1）呼出書式

[c,C++]

```
API int APIX GemSendS1F11(
        int  eqid,                          // 通信対象装置 ID(0,1,2,...)
        TVID *svid_list,                    // 要求する SVID のﾘｽﾄ領域のﾎﾟｲﾝﾀ
        int  count,                         // svid_list に含まれる SVID の数
        TSV_NAME_LIST *name_list,           // s1f12 で返された状態変数名一覧情報格納用ﾘｽﾄﾎﾟｲﾝﾀ
        int (WINAPI *s1f11Callback)(),      // 実行終了時のｺｰﾙﾊﾞｯｸ関数
        ULONG upara                         // callback 時のﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

```
Function GemSendS1F11 (
        ByVal eqid As Int32,
        ByRef svid_list As Int32,
        ByVal count As Int32,
        ByRef name_list As dsh_info.TSV_NAME_LIST,
        ByVal callback As vcallback.callback_S1F11,
        ByVal upara As Int32) As Int32
```

[.NET C#]

```
int GemSendS1F11(
        int eqid,
        ref uint svid_list,
        int count,
        ref TSV_NAME_LIST name_list,
        CallbackS1F11 callback,
        uint upara );
```

（2）引数

eqid

GEM 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。

svid_list

装置に要求したい状態変数名一覧の状態変数 ID が格納されているリストのポインタです。

count

svid_list 内に含まれる装置状態変数 ID の数です。
count=0 の場合は、全状態変数が対象になります。

name_list

svid_list 内に指定された状態変数 ID の変数名一覧情報を格納するためのリストへのポインタです。
変数名一覧情報には変数名と値の単位名が与えられます。

s1f11Callback

DSHGEM-LIB によるメッセージ送信処理が終了したときに呼出されるコールバック関数を指定します。
ユーザは任意の関数名を指定できます。
このコールバックの指定が=0 の場合はブロックモードになります。

upara

コールバックされたときに引数で渡してもらうためのパラメータです。関数実行終了時にコールバ

ックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定します。

（3）戻り値

戻り値	意味
0	(1) ﾌﾞﾛｯｸﾓｰﾄﾞ： 正常に送信できた。 name_list に受信した状態変数名一覧情報が格納されています。 (2) 非ﾌﾞﾛｯｸﾓｰﾄﾞ：要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3ﾀｲﾑｱｳﾄを検出した。（応答ﾒｯｾｰｼﾞが得られなかった）

（4）説明

装置に対し、svid_list 内に指定された count 分の状態変数 ID の状態変数名一覧を S1F11 メッセージを使って要求します。

正常に応答メッセージを受信できた場合は、そのメッセージをデコードし、name_list で指定された構造体リストに格納します。

また、count＝0 の場合は、予め GemSetSvidList()関数を使って、装置から S1F12 内に並べ与えられる SVID を DSHGEM-LIB に設定しておく必要があります。

送信要求から S1F11 応答メッセージ受信までの制御は引数の S1F11 コールバック関数指定の有無によって次のようになります。

s1f11Callback の指定	制御の流れ
なし (=0)	S1F11 送信後、応答ﾒｯｾｰｼﾞ S1F12 が受信されるか、またはｴﾗｰを検出するまで制御が要求元に戻ってきません。 正常終了であれば、name_list に S1F12 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。
あり	送信要求後、S1F11 の送信前に制御が戻されます。実行結果はｺｰﾙﾊﾞｯｸ関数で与えられます。 ｺｰﾙﾊﾞｯｸ関数の end_status が=0 ならば正常で引数 name_list に S1F12 ﾒｯｾｰｼﾞ内の情報がﾃﾞｺｰﾄﾞされて渡されます。 ｴﾗｰが検出された場合、(-1)が end_status にｾｯﾄされます。

正常に応答メッセージを受信した場合、name_list 構造体内に状態変数名一覧情報が渡されますが、ユーザ側で情報の処理を終えた後、その構造体に使用されているメモリを解放してください。

```
DshFreeTSV_NAME_LIST(name_list);
```

（5）コールバック関数

[c,C++]

```
API int APIX s1f11Callback(
        int    eqid,                    // 装置 ID
        int    end_status,              // 実行結果
        TSV_NAME_LIST *name_list,       // SV 名一覧が格納されている構造体のﾎﾟｲﾝﾀ
        ULONG upara                     // 呼出時に指定したﾊﾟﾗﾒｰﾀ
);
```

[.NET VB]

Function callback_S1F11(ByVal eqid As Integer, ByVal end_status As Integer, ByRef name_list As dsh_info.TSV_NAME_LIST, ByVal upara As Integer) As Integer

[.NET C#]

int CallbackS1F11(int eqid, int end_status, ref TSV_NAME_LIST namelist, uint upara);

end_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。
(-1)	送信ｴﾗｰを検出した。
(-14)	T3 ﾀｲﾑｱｳﾄを検出した。

end_status = 0 のときのみ name_list の内容が有効です。

3．3．5　DVVAL データ変数情報アクセスとメッセージ送信関数

3．3．5．1　GemSetDvVal()　- DVVAL 値設定関数

（1）呼出書式

[c,C++]

```
API int APIX GemSetDvVal(
      int   eqid,                    // 通信対象装置 ID(0,1,2,...)
      TVID  vid,                     // VID
      void  *val,                    // 設定値ﾎﾟｲﾝﾀ
      int   asize                    // ﾃﾞｰﾀの配列ｻｲｽﾞ
);
```

[.NET VB]

```
Function GemSetDvVal (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByVal arraysize As Int32) As Int32

    Function GemSetDvVal (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByRef val As Int32,
        ByVal arraysize As Int32) As Int32

    Function GemSetDvVal (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As String,
        ByVal arraysize As Int32) As Int32
```

[.NET C#]

```
int GemSetDvVal(
        int eqid,
        uint vid,
        byte[] val,
        int arraysize );
```

（2）引数

eqid

GEM 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。

vid

装置データ変数 ID です。
装置管理情報定義ファイルに登録されているデータ変数の ID でなければなりません。

val

設定したい値の格納ポインタです。vid のフォーマットに合わせた値を設定してください。

asize

設定値の配列サイズ格納です。
フォーマットA,J の場合は、文字列長になります。

（3）戻り値

戻り値	意味
0	正常に設定された。
(-1)	vid の値が正しくなかった。（登録されていなかった）

（4）説明

vid で指定されたV(装置データ変数)に新しい値を設定します。戻り値が設定結果になります。
正常に設定できた場合は、関数の戻り値は 0 になります。

３．３．５．２　GemGetDvVal() - DVVAL 値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetDvVal(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // VID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀのﾊﾞｲﾄ長格納用
);
```

[.NET VB]

```
Function GemGetDvVal (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetDvVal(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

vid

装置データ変数 ID です。
装置管理情報定義ファイルに登録されているデータ変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。vid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。
フォーマット A,J の場合は、文字列長になり、文字列の最後に NULL 文字(=0)が付加されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置データ変数)の現在値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。

３．３．５．３ GemGetDvName() - DVVAL名取得関数
GemGetDvNameA()

（1）呼出書式

[c,C++]

```
API int APIX GemGetDvName(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TVID   vid,                     // VID
        char   *name,                   // 取得した名前の格納領域ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ格納用
);

API int APIX GemGetDvNameA(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TVID vid,                       // VID
        char   *name                    // 取得した名前の格納領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetDvName (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As String,
        ByRef arraysize As Int32) As Int32

    Function GemGetDvName (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As String) As Int32
```

[.NET C#]

```
int GemGetDvName(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );

    int GemGetDvName(
        int eqid,
        uint vid,
        byte[] val);
```

（2）引数

eqid

GEM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。

vid

装置データ変数IDです。
装置管理情報定義ファイルに登録されているデータ変数のIDでなければなりません。

name

取得した名前（文字列）を格納する領域のポインタです。名前格納に充分な領域を準備してください。

asize

取得された名前の文字列長です。

（3）戻り値

戻り値	意味
0	正常に取得できた。 - GemGetDvName()の場合
> 0	正常に取得できた。 - GemGetDvNameA()の場合は名前のﾊﾞｲﾄ長が返る
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された DV(装置データ変数)の名前を取得します。

正常に取得できた場合は、GemGetDvName()の戻り値は 0 になり、取得された名前の配列サイズは asize で指定された領域に返却されます。

GemGetDvNameA()は取得できた名前の長さが返却されます。
引数 asize が無い場合は、名前のバイト長が返却されます。

３．３．５．４　GemGetDvUnits() - DVVAL単位名取得関数
GemGetDvUnitsA()

（1）呼出書式

[c,C++]

```
API int APIX GemGetDvUnits(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TDVID dvid,                     // DVID
        char  *units,                   // 取得した単位名の格納領域ﾎﾟｲﾝﾀ
        int    *asize                   // 単位名の文字列長格納用
);

API int APIX GemGetDvUnitsA(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TDVID dvid,                     // DVID
        char  *units                    // 取得した単位名の格納領域ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetDvUnits (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As String,
        ByRef arraysize As Int32) As Int32

    Function GemGetDvUnits (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As String) As Int32
```

[.NET C#]

```
int GemGetDvUnits(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );

    int GemGetDvUnits(
        int eqid,
        uint vid,
        byte[] val );
```

（2）引数

eqid

GEM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。

dvid

装置データ変数IDです。
装置管理情報定義ファイルに登録されているデータ変数のIDでなければなりません。

units

取得した単位名（文字列）を格納する領域のポインタです。名前格納に充分な領域を準備してください。

asize

取得された単位名の文字列長です。もし、単位名が無い場合には =0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。 - GemGetDvUnits()の場合
>0	正常に取得できた。 - GemGetDvUnitsA()の場合は名前のﾊﾞｲﾄ長が返る
(-1)	vid の値が正しくなかった。

（4）説明

dvid で指定された DV(装置データ変数)の値の単位名を取得します。

正常に取得できた場合は、GemGetDvUnits()の戻り値は 0 になり、取得された単位名の配列サイズは asize で指定された領域に返却されます。

GemGetDvUnitsA()は取得できた単位名の長さが返却されます。
引数 asize が無い場合は、単位名のバイト長が返却されます。

３．３．５．５　GemGetDvFormat() - DVVAL フォーマットコード取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetDvFormat(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // VID
        int    *fmt                     // 取得ﾌｫｰﾏｯﾄ値格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetDvFormat (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByRef fmt As Int32) As Int32
```

[.NET C#]

```
int GemGetDvFormat(
        int eqid,
        uint vid,
        ref int fmt );
```

（2）引数

eqid

GEM 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。

vid

装置データ変数 ID です。
装置管理情報定義ファイルに登録されているデータ変数の ID でなければなりません。

fmt

取得したフォーマットの値を格納する領域のポインタです。
返却されるフォーマット値は、DSHDR2 ドライバーが定義するアイテムフォーマットです。

（3）戻り値

戻り値	意味
> 0	正常に取得できた。戻り値はﾌｫｰﾏｯﾄの単位ﾊﾞｲﾄ長です。
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置データ変数)のフォーマット値を取得します。
フォーマット値は、DSHDR2 ドライバーで定義するアイテムフォーマットです。
（dsh.h ファイルでマクロ定義されています。）
正常に取得できた場合は、関数の戻り値は アイテムフォーマットの単位配列サイズ になります。

３．３．５．６　GemGetDvArraySize() - DVVAL 値の配列サイズ取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetDvArraySize(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // VID
        int    *asize                   // 取得配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetDvArraySize (
          ByVal eqid As Int32,
          ByVal vid As Int32,
          ByRef val As Int32) As Int32
```

[.NET C#]

```
int GemGetDvArraySize(
          int eqid,
          uint vid,
          ref int val );
```

（2）引数

eqid

GEM 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。

vid

装置データ変数 ID です。

装置管理情報定義ファイルに登録されているデータ変数の ID でなければなりません。

asize

取得した配列サイズ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)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置データ変数)の配列サイズを取得します。

例えば、指定された V が、DV_temp_FORMAT: A[6] の場合は、配列サイズとして=6 が返却されます。

ただし、可変配列文字列変数については最小配列サイズが返却されます。

FORMAT： A[4..16] の場合は =4 が返却されます。

３．３．５．７　GemGetDvMin() - DVVAL 最小(Min)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetDvMin(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // VID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ 格納用
);
```

[.NET VB]

```
Function GemGetDvMin (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetDvMin(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

vid

装置データ変数 ID です。
装置管理情報定義ファイルに登録されているデータ変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。vid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。フォーマット A,J の場合は、文字列長になります。
最小値が定義されていない場合は、=0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置データ変数)の最小(Min)値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。ただし、指定された V が最小値を持たない場合は、asize に = 0 が返却されます。

３．３．５．８　GemGetDvMax() - DVVAL 最大(Max)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetDvMax(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // VID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ格納用
);
```

[.NET VB]

```
Function GemGetDvMax (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetDvMax(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

vid

装置データ変数 ID です。
装置管理情報定義ファイルに登録されているデータ変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。vid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。フォーマット A,J の場合は、文字列長になります。
最大値が定義されていない場合は、=0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置データ変数)の最大(Max)値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。ただし、指定された V が最大値を持たない場合は、asize に = 0 が返却されます。

３．３．５．９　GemGetDvNominal() - DVVAL 初期(Nominal)値取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetDvNominal(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TVID   vid,                     // VID
        void   *val,                    // 取得値格納ﾎﾟｲﾝﾀ
        int    *asize                   // ﾃﾞｰﾀの配列ｻｲｽﾞ 格納用
);
```

[.NET VB]

```
Function GemGetDvNominal (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByRef arraysize As Int32) As Int32
```

[.NET C#]

```
int GemGetDvNominal(
        int eqid,
        uint vid,
        byte[] val,
        ref int arraysize );
```

（2）引数

eqid

GEM 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。

vid

装置データ変数 ID です。
装置管理情報定義ファイルに登録されているデータ変数の ID でなければなりません。

val

取得した値を格納する領域のポインタです。vid のフォーマットと配列サイズの値を格納するために充分な領域を準備してください。実際に取得した値の配列サイズは asize に格納されます。

asize

取得された値の配列サイズ格納用です。フォーマット A,J の場合は、文字列長になります。
初期値が定義されていない場合は、=0 が格納されます。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	vid の値が正しくなかった。

（4）説明

vid で指定された V(装置データ変数)の初期値を取得します。
正常に取得できた場合は、関数の戻り値は 0 になります。取得データの配列サイズは asize で指定された領域に返却されます。ただし、指定された V が初期値を持たない場合は、asize に = 0 が返却されます。

３．３．５．10 GemCheckDvVal() - DVVAL値Min,Maxチェック関数

（1）呼出書式

[c,C++]

```
API int APIX GemCheckDvVal(
        int    eqid,                    // 通信対象装置ID(0,1,2,...)
        TVID   vid,                     // VID
        void   *val,                    // ﾁｪｯｸ値格納ﾎﾟｲﾝﾀ
        int    asize                    // ﾃﾞｰﾀの配列ｻｲｽﾞ
);
```

[.NET VB]

```
Function GemCheckDvVal (
        ByVal eqid As Int32,
        ByVal vid As Int32,
        ByVal val As IntPtr,
        ByVal arraysize As Int32) As Int32
```

[.NET C#]

```
int GemCheckDvVal(
        int eqid,
        uint vid,
        byte[] val,
        int arraysize );
```

（2）引数

eqid

GEM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。

vid

装置データ変数IDです。

装置管理情報定義ファイルに登録されているデータ変数のIDでなければなりません。

val

チェックするVの値が格納されている領域のポインタです。

asize

チェック値領域の配列サイズです。

（3）戻り値

戻り値	意味
0	正常な値である。
(-1)	vidの値が正しくなかった。
(-2)	値が最小値より小さかった。
(-3)	値が最大値より大きかった。

（4）説明

vidで指定されたV(装置データ変数)としてのval領域に格納されているデータ値をチェックします。

チェック方法は、valに格納されている値が、当該vidのVのために指定された最小値、最大値の範囲内であるかどうかの判断です。最小値、最大値がもし指定されていない場合は、指定されていないものとの比較判断は行いません。(正常であるとみなします。)

３．３．５．11　GemGetDvList() - 全登録 DVVALID 取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetDvList(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TBIN_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 GemGetDvList (
        ByVal eqid As Int32,
        ByRef list As IntPtr) As Int32
```

[.NET C#]

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

（2）引数

eqid

GEM 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 DVID が格納されている TBIN_DLIST 構造体のポインタを格納する領域のポインタです。

（3）戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

（4）説明

システムに登録されている全 DVID とその名前を TBIN_DLIST 構造体に取出すための関数です。
取出す名前は、装置管理情報定義ファイルで DVVAL 定義時に与えられた名前です。
取得した情報の処理が終了した後、DshFreeTBIN_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。

TBIN_DLIST 構造体は次のとおりです。

```
typedef struct{
        int         count;              // 取得できた ID 数
        ULONG       *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
        }TBIN_DLIST;
```

３．３．５．12 GemGetDvSizeMinMax() - DVVAL 値の最小、最大配列サイズ取得関数

（1）呼出書式

[c,C++]

```
API int APIX GemGetDvSizeMinMax(
        int    eqid,                    // 通信対象装置 ID(0,1,2,...)
        TDVID dvid,                     // DVID
        int    *min,                    // 最小配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
        int    *max                     // 最大配列ｻｲｽﾞ格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function GemGetDvSizeMinMax (
          ByVal eqid As Int32,
          ByVal vid As Int32,
          ByRef min As Int32,
          ByRef max As Int32) As Int32
```

[.NET C#]

```
int GemGetDvSizeMinMax(
          int eqid,
          uint vid,
          ref int min,
          ref int max );
```

（2）引数

eqid

GEM 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。

dvid

装置データ変数 ID です。
装置管理情報定義ファイルに登録されているデータ変数の ID でなければなりません。

min

取得した最小配列サイズを格納する領域のポインタです。

max

取得した最大配列サイズ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)	dvid の値が正しくなかった。

（4）説明

dvid で指定された DVVAL(装置データ変数)の最小と最大配列サイズを取得します。
変数の配列サイズが固定の場合は、min, max に同じ値が返却されます。

可変配列文字列変数については最小、最大配列サイズがそれぞれ min,max に返却されます。
FORMAT： A[4..16] の場合は min =4 , max = 16 となります。

３．３．６　変数情報操作関連ライブラリ関数

３．３．６．１　DshInitTV_VALUE_LIST () - 変数情報構造体の初期設定

（1）呼出書式

[c,C++]

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

[.NET VB]

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

[.NET C#]

```
Sub DshInitTV_VALUE_LIST (
        ByRef list As dsh_info.TV_VALUE_LIST,
        ByVal count As Int32)
```

（2）引数

list

TV_VALUE_LIST 構造体のポインタです。

count

list に格納する変数情報の数です。(VID の数)

（3）戻り値

なし。

（4）説明

GemSendS2F15()関数を使って装置変数の変更のために S2F15 メッセージを送信する際、GemSendS2F15()関数の引数として TV_VALUE_LIST 構造体情報を付けます。その TV_VALUE_LIST 内に変更したい変数の数、変数 ID ならびに変数値を設定します。

DshInitTV_VALUE_LIST()関数は TV_VALUE_LIST を初期化するための関数です。

TV_VALUE_COUNT に count 分の変数情報を格納するための初期設定処理を行います。

３．３．６．２　DshAddTV_VALUE_LIST () - 装置変数情報の追加

（1）呼出書式

[c,C++]

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

[.NET VB]

```
Function DshAddTV_VALUE_LIST (
        ByRef list As dsh_info.TV_VALUE_LIST,
        ByVal vid As Int32,
        ByVal fmt As Int32,
        ByVal asize As Int32,
        ByVal value As Int32) As Int32
```

[.NET C#]

```
int DshAddTV_VALUE_LIST(
        ref TV_VALUE_LIST list,
        uint vid,
        int fmt,
        int asize,
        byte[] value );
```

（2）引数

list

TV_VALUE_LIST 構造体のポインタです。

vid

list 内に追加する変数 ID です。

fmt

変数のデータフォーマットです。

asize

変数データの配列サイズです。

value

変数値が格納されている領域のポインタです。

（3）戻り値

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

（4）説明

先に説明したDshInitTV_VALUE_LIST()関数で初期設定されたlist内に1個の変数データ情報を追加します。
追加する情報は引数に指定されている ID とデータ情報です。
追加は、本関数が呼び出される順番に先頭から行われます。

DshInitTV_VALUE_LIST()関数で設定した count 分だけの変数情報を加えることができます。
count 分を超える数の変数情報を追加しようとした場合、戻り値として(-1)が返却されます。

３．３．６．３　DshFreeTV_VALUE_LIST() - 変数情報リスト構造体メモリの開放

（1）呼出書式

[c,C++]

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

[.NET VB]

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

[.NET C#]

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

（2）引数

list

メモリを解放したい変数情報リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TV_VALUE_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TV_VALUE_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。
GemSendS1F3(), GemSendS2F13() API 関数の実行で得られた変数情報リスト構造体に使用されているメモリを開放します。
また、GemSendS2F15() API 関数で引数として使用した変数情報リスト構造体のメモリの開放にも使用します。

３．３．６．４　DshFreeTEC_NAME_LIST() - 装置定数名一覧リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTEC_NAME_LIST(
      TEC_NAME_LIST *list                  // ﾒﾓﾘを開放したい装置定数名一覧ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

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

[.NET C#]

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

（2）引数

list

メモリを解放したい装置定数名一覧リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TEC_NAME_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TEC_NAME_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。
GemSendS2F29() API 関数の実行で得られた装置定数名一覧リスト構造体に使用されているメモリを開放します。

３．３．６．５　DshFreeTSV_NAME_LIST() - 装置状態変数名一覧リスト構造体メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTSV_NAME_LIST(
      TSV_NAME_LIST *list          // ﾒﾓﾘを開放したい装置状態変数名一覧ﾘｽﾄ構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

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

[.NET C#]

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

（2）引数

list

メモリを解放したい装置状態変数名一覧リスト構造体のポインタです。

（3）戻り値

なし。

（4）説明

TSV_NAME_LIST 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TSV_NAME_LIST の内容を 全て 0 で初期設定します。
list が NULL ならば、何も処理しません。
GemSendS1F11() API 関数の実行で得られた装置状態変数名一覧リスト構造体に使用されているメモリを開放します。

３．３．６．６　DshEncodeVidList() - VID ﾘｽﾄを DSHMSG のテキストへエンコード

（1）呼出書式

[c,C++]

```
API int  APIX DshEncodeVidList(
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        int    s,                       // Stream (Sx)
        int    f,                       // Function(Fy)
        BYTE   *buffer,                 // VIDLIST を格納するﾊﾞｯﾌｧﾎﾟｲﾝﾀ
        int    bufflen,                 // buffer のﾊﾞｲﾄｻｲｽﾞ
        TVID   *vid_list,               // ｴﾝｺｰﾄﾞしたい変数 ID ﾘｽﾄのﾎﾟｲﾝﾀ
        int    count                    // vid_list に設定されている変数 ID 数
);
```

[.NET VB]

```
Function DshEncodeVidList (
        ByRef smsg As dshdr2.DSHMSG,
        ByVal s As Int32,
        ByVal f As Int32,
        ByRef buff As Byte,
        ByVal bufflen As Int32,
        ByRef list As Int32,
        ByVal count As Int32) As Int32
```

[.NET C#]

```
int DshEncodeVidList(
        ref DSHMSG smsg,
        int s,
        int f,
        byte[] buff,
        int bufflen,
        ref uint list,
        int count );
```

（2）引数

smsg

エンコードしたメッセージを格納するメッセージ情報構造体のポインタです。

s

smsg に設定する Stream コード Sx です。

f

smsg に設定する Function コード Fy です。

buffer

エンコードした SECS-II メッセージテキストを格納するためのバッファポインタです。

bufflen

buffer のバイトサイズです。

vid_list

エンコードしたい変数 ID ﾘｽﾄが格納されているメモリのポインタです。

count

変数 ID ﾘｽﾄ vid_list 内に設定されている変数 ID 数です。

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	smsg を正しくｴﾝｺｰﾄﾞできなかった。 ｴﾝｺｰﾄﾞ中に buffer ｵｰﾊﾞｰﾌﾛｰを検出した。

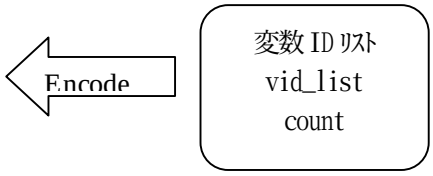
（4）説明

vid_list に格納されている変数を、SECS メッセージ情報構造体 smsg と buffer で指定されたテキストバッファにエンコードします。S1F3，S1F13，S2F13，S2F29，S2F47 メッセージのエンコードに利用します。

buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。

smsg，buffer

L,n
　vid1
　vid2
　.
　.



SECS メッセージ情報構造体 smsg 内の stream、function、buffer メンバーと length の設定も行います。

３．３．６．７　DshEncodeVarValMsg () - 変数値ﾘｽﾄを DSHMSG のテキストへエンコード

（1）呼出書式

[c,C++]

```
API int  APIX DshEncodeVarValMsg (
        DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        int     s,                      // Stream (Sx)
        int     f,                      // Function(Fy)
        BYTE    *buffer,                // VIDLIST を格納するﾊﾞｯﾌｧﾎﾟｲﾝﾀ
        int     bufflen,                // buffer のﾊﾞｲﾄｻｲｽﾞ
        TV_VALUE_LIST *list             // ｴﾝｺｰﾄﾞ対象変数値ﾘｽﾄ情報が格納されている構造体ﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshEncodeVarValMsg (
        ByRef smsg As dshdr2.DSHMSG,
        ByVal s As Int32,
        ByVal f As Int32,
        ByRef buff As Byte,
        ByVal bufflen As Int32,
        ByRef list As dsh_info.TV_VALUE_LIST) As Int32
```

[.NET C#]

```
int DshEncodeVarValMsg(
        ref DSHMSG smsg,
        int s,
        int f,
        byte[] buff,
        int bufflen,
        ref TV_VALUE_LIST list );
```

（2）引数

smsg

エンコードしたメッセージを格納するメッセージ情報構造体のポインタです。

s

smsg に設定する Stream コード Sx です。

f

smsg に設定する Function コード Fy です。

buffer

エンコードした SECS-II メッセージテキストを格納するためのバッファポインタです。

bufflen

buffer のバイトサイズです。

list

エンコードしたい変数 ID ﾘｽﾄが格納されているメモリのポインタです。

count

変数 ID ﾘｽﾄ vid_list 内に設定されている変数 ID 数です。

（3）戻り値

戻り値	意味
0	正常にｴﾝｺｰﾄﾞできた。
(-1)	smsg を正しくｴﾝｺｰﾄﾞできなかった。 ｴﾝｺｰﾄﾞ中に buffer ｵｰﾊﾞｰﾌﾛｰを検出した。

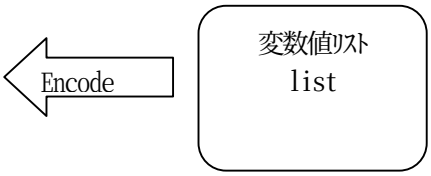
（4）説明

vid_list に格納されている変数を、SECS メッセージ情報構造体 smsg と buffer で指定されたテキストバッファにエンコードします。S2F15（装置定数の変更）メッセージのエンコードに利用します。

buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。

smsg, buffer

```
L,n
  L2
    ecid1
    ecval1
  .
  .
```



SECS メッセージ情報構造体 smsg 内の stream、function、buffer メンバーと length の設定も行います。

３．３．６．８　DshDecodeVarValMsg() - 変数値メッセージを TV_VAL_LIST 構造体へデコード

（1）呼出書式

[c,C++]

```
API int  APIX DshDecodeVarValMsg (
        DSHMSG *smsg,                   // ﾒｯｾｰｼﾞが格納されている SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
        TV_VALUE_LIST  *list            // ﾃﾞｺｰﾄﾞした変数値格納用構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshDecodeVarValMsg (
          ByRef smsg As dshdr2.DSHMSG,
          ByRef list As dsh_info.TV_VALUE_LIST) As Int32
```

[.NET C#]

```
int DshDecodeVarValMsg(
          ref DSHMSG smsg,
          ref TV_VALUE_LIST list );
```

（2）引数

smsg

変数値メッセージが格納されているメッセージ情報構造体のポインタです。

list

デコードした変数値リストの結果を格納するための構造体のポインタです。

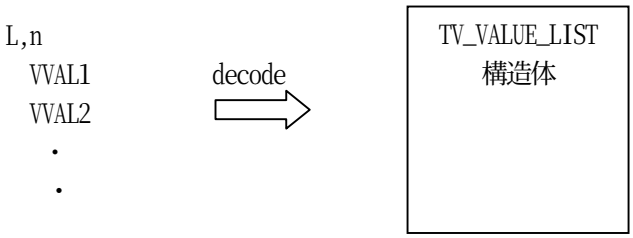
（3）戻り値

戻り値	意味
0	正常にﾃﾞｺｰﾄﾞできた。
(-1)	smsg を正しくﾃﾞｺｰﾄﾞできなかった。

（4）説明

変数値を含むメッセージに含まれる変数値情報を list で指定される TV_VALUE_LIST 構造体にデコードし格納します。S1F4, S2F14 メッセージのデコードに使用します。

smsg (S1F4 or S2F14)



なお、TV_VALUE_LIST 内の vid メンバーには、=0 が設定格納されます。
得られた list 情報の処理が終った後は、DshFreeTV_VALUE_LIST()関数で list 内部に割付使用されたメモリを開放することができます。

3．3．6．9　DshGetTV_VALUE_LIST_vid() - 装置変数情報の取得

（1）呼出書式

[c,C++]

```
API TVID APIX DshGetTV_VALUE_LIST_vid (
      TV_VALUE_LIST *list,          // TV_VALUE_LIST 変数値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
      int           order,          // 取得する変数の順位(0,1,2...)
      int           *fmt,           // 変数ﾃﾞｰﾀのﾌｫｰﾏｯﾄ格納番地
      int           *asize          // 変数ﾃﾞｰﾀの配列ｻｲｽﾞ 格納番地
);
```

[.NET VB]

```
Function DshGetTV_VALUE_LIST_vid (
        ByRef list As dsh_info.TV_VALUE_LIST,
        ByVal order As Int32,
        ByRef fmt As Int32,
        ByRef asize As Int32) As Int32
```

[.NET C#]

```
uint DshGetTV_VALUE_LIST_vid (
        ref TV_VALUE_LIST list,
        int order,
        ref int fmt,
        ref int asize);
```

（2）引数

list

TV_VALUE_LIST 構造体のポインタです。

order

list 内に存在する変数の順位です。list[order] の変数対象です。

fmt

取得した変数のデータフォーマットを格納する領域です。

asize

取得した変数データの配列サイズを格納する領域です。

（3）戻り値

戻り値	意味
>=0	正常に取得できたときの VID です。
(-1)	order で指定された位置に情報が無かった。

（4）説明

TV_VALUE_LIST 構造体である list 内に order で指定された配列順位の変数データ情報の VID，データフォーマット，データサイズを取得します。
TV_VALUE_LIST には、1 個以上の変数情報の格納に使用されます。

info 構造体内の order >= list->count の場合は、VID として返却値(-1)を返却します。
データ形式は、DSHDR2HSMS 通信ドライバーで定義される形式で、ICODE_A，ICODE_B などで定義される SECS-II の仕様が定めるデータアイテムコードです。

３．３．６．10　DshPutTV_VALUE_LIST_vid() - 装置変数情報へのVIDの設定

（1）呼出書式

[c,C++]

```
API int APIX DshPutTV_VALUE_LIST_vid (
        TV_VALUE_LIST *list,            // TV_VALUE_LIST 変数値情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int           order,            // 設定変数の順位(0,1,2...)
        TVID          vid       // 設定する VID
);
```

[.NET VB]

```
Function DshAddTV_VALUE_LIST (
          ByRef list As dsh_info.TV_VALUE_LIST,
          ByVal order As Int32,
          ByVal vid As Int32) As Int32
```

[.NET C#]

```
uint DshAddTV_VALUE_LIST(
          ref TV_VALUE_LIST list,
          int order,
          uint vid);
```

（2）引数

list

TV_VALUE_LIST 構造体のポインタです。

order

list 内に存在する変数の順位です。list[order] の変数対象です。

vid

order 番目の変数情報に設定する変数 ID です。

（3）戻り値

戻り値	意味
0	正常に設定できた。
(-1)	order で指定された位置に情報が無かった。

（4）説明

TV_VALUE_LIST 構造体である list 内の order で指定された配列順位の変数データ情報の中の vid メンバーに引数で与えられた変数 ID を設定します。
info 構造体内の order >= list->count の場合は、エラー (-1)を返却します。
正常に設定できた場合は、=0 を返却します。

用途は、S2F13 装置定数(EC)要求の応答 S2F14 メッセージをデコードし、情報を TV_VAL_LIST 内に取得格納しますが、S2F14 メッセージには、変数 ID が含まれていません。すなわち、TV_VALUE_LIST を構成する TV_VALUE 構造体の vid メンバーが不定になっています。それで、アプリケーションプログラムが各変数情報と変数 ID の対応が取れ、処理しやすいように、TV_VALUE 構造体に変数 ID を設定するために使用するものです。
装置状態変数(SV)要求の S1F3，S1F4 のケースでも同様の目的で使用することができます。