SavoyGem ActiveX コントロール
ユーザーガイド

Jazz Soft, Inc.

1　改訂履歴

バージョン	日付	氏名	説明
1.00	2009 年 7 月 31 日	Hikaru Okada	新規に作成。
1.00a	2009 年 8 月 22 日	Hikaru Okada	マニュアルのページ数が大きくなったため分割。
1.00b	2009 年 8 月 30 日	Hikaru Okada	一部の誤記を訂正。
1.00c	2009 年 10 月 12 日	Hikaru Okada	DataBakCount を追加。
1.00d	2009 年 12 月 24 日	Hikaru Okada	ControlStateSwitch の記述を訂正。
1.00e	1020 年 12 月 14 日	Hikaru Okada	ALIDEnable, ALIDSet, CEIDEnable プロパティ、RegisterCEID メソッドを追加。

2 目次

3 SavoyGem

SavoyGem コントロールは SEMI E30 (GEM)の機能を作成するための開発支援製品です。装置側ソフトに対ホスト向け GEM 通信機能を実装する場合、膨大な予備知識や作業が必要となりますが、SavoyGem コントロールはそれを大幅に軽減してくれます。

プロパティ

名前	説明
ALIDCount	登録されている ALID の数を取得します。
Appearance	外観を決定する値を取得または設定します。
BorderStyle	境界線スタイルを取得または設定します。
CEIDCount	登録されている CEID の数を取得します。
CommunicationState	GEM 通信状態を取得または設定します。
ControlState	GEM コントロール状態を取得または設定します。
ControlStateSwitch	コントロール状態スイッチを設定します。
DataFileName	データファイル名を取得または設定します。
DeviceID	デバイス ID を取得または設定します。
DiscardDuplicatedBlock	二重ブロックを破棄するかどうかを取得または設定します。
Function	SECS-II のファンクション番号を取得または設定します。
Host	ホストか装置かの役割を取得または設定します。
IniFileName	設定を読み書きするための INI ファイル名を取得または設定します。
IPAddress	HSMS 接続のパッシブエンティティの IP アドレスを取得または設定します。
Log	ログの有効・無効を取得または設定します。
LogBakCount	ログのバックアップファイルの数を取得または設定します。
LogCommunication	通信部分のログが有効かどうかを取得または設定します。
LogFileName	ログファイル名を取得または設定します。
LogicalConnection	論理接続を取得または設定します。
LogSize	ログファイルのサイズを取得または設定します。
Msg	メッセージデータを取得または設定します。
MyPortNumber	HSMS 接続のローカルポート番号を取得または設定します。
Node	メッセージ内容を操作するためのノードを取得または設定します。
NodeCount	サブアイテムの個数を取得します。
NodeType	ノードタイプを取得します。
NodeValue	ノードの値を取得します。
NodeValueHex	ノードの値を 16 進法表現で取得します。
OfflineRequest	オフライン要求を取得または設定します。
OnlineRequest	オンライン要求を取得または設定します。
Reply	ワークスペースのモードを取得または設定します。
PhysicalConnection	物理接続を取得または設定します。
PortNumber	HSMS のポート番号を取得または設定します。
PType	プレゼンテーションタイプを取得または設定します。
Server	エンティティのタイプを取得または設定します。
SessionID	セッション ID を取得または設定します。
SML	メッセージを SML 文字列として取得または設定します。
Stream	ストリーム番号を取得または設定します。
SType	セッションタイプを取得または設定します。
SystemBytes	システムバイトを取得または設定します。
T1	T1 タイムアウトを取得または設定します。
T2	T2 タイムアウトを取得または設定します。
T3	T3 タイムアウトを取得または設定します。
T4	T4 タイムアウトを取得または設定します。
T5	T5 タイムアウトを取得または設定します。
T6	T6 タイムアウトを取得または設定します。
T7	T7 タイムアウトを取得または設定します。
T8	T8 タイムアウトを取得または設定します。
Verification	メッセージ構造の検証結果を取得または設定します。

VIDCount	VID の個数を取得します。
ViewStyle	表示形式を取得または設定します。
Wbit	ウェイトビットを取得または設定します。
WorkSpace	ワークスペースを取得または設定します。

配列型プロパティ

名前	説明
ALCD	ALID のアラームコードを取得または設定します。
ALTX	ALID のアラームテキストを取得または設定します。
CEIDDescription	CEID の説明を取得または設定します。
VIDDefault	VID のデフォルト値を取得または設定します。
VIDDescription	VID の説明を取得または設定します。
VIDMax	VID の最大値を取得または設定します。
VIDMin	VID の最小値を取得または設定します。
VIDNodeType	VID のノードタイプを取得または設定します。
VIDRawValue	VID の現在の値を取得または設定します。
VIDType	VID の変数型を取得または設定します。
VIDUnit	VID の単位を取得または設定します。
VIDValue	VID の現在の値を型に合わせた表現で取得または設定します。

メソッド

名前	説明
AboutBox	バージョン情報を表示します。
DefProc	デフォルトの処理を呼び出します。
InvokeAlarm	アラームイベントを発生させます。
InvokeEvent	イベント発生させます。
IsValidVID	指定された VID が有効かどうかを検査します。
LoadData	データファイルを読み込みます。
LoadIniFile	設定内容を INI ファイルから読み出します。
RegisterALID	ALID を登録します。
RegisterVID	VID を登録します。
SaveData	データファイルにデータを保存します。
Send	指定されたメッセージを送信します。
Setup	セットアップ画面を表示します。
ToALID	インデックスを ALID に変換します。
ToCEID	インデックスを CEID に変換します。
ToVID	インデックスを VID に変換します。
UnregisterALID	ALID を登録解除します。
UnregisterVID	VID を登録解除します。
WriteLogFile	指定された文字列をログファイルに書き込みます。

イベント

名前	説明
CommunicationStateChanged	通信状態が変化したときに通知されます。
Connected	HSMS 接続が成立したときに通知されます。
ConnectionStateChanged	接続状態が変化したときに通知されます。
ControlStateChanged	コントロール状態が変化したときに通知されます。
Disconnected	HSMS 接続が切断したときに通知されます。
Problem	エラーが発生したときに通知されます。
Received	メッセージを受信したときに通知されます。
Sent	メッセージが送信されたときに通知されます。
VIDChanged	VID が変更されたときに通知されます。

3.1 プロパティ

3.1.1 ALIDCount

登録されている ALID の数を取得します。0 の場合は一つも登録されていないことを意味します。

構文

Visual Basic 6.0

```
ALIDCount As Long
```

Visual C++ 6.0

```
long GetALIDCount()
```

使用例

Visual Basic 6.0

```
Dim lALIDCount As Long
lALIDCount = .ALIDCount
```

Visual C++ 6.0

```
long lALIDCount = m_ctrl.GetALIDCount();
```

特記事項

読み出し専用プロパティ。

ALIDCount プロパティは登録されている ALID の数を返すので、インデックスとして使用可能な値は 0 から(ALIDCount - 1)までとなります。これを ToALID メソッドを使って ALID に変換します。

参照

3.1.2 Appearance

SavoyGem コントロールの外観を決定する値を取得または設定します。

値	説明
0	フラット
1	凹んだ枠線

構文

Visual Basic 6.0

```
Appearance As Integer
```

Visual C++ 6.0

```
short GetAppearance()
void SetAppearance(short)
```

使用例

Visual Basic 6.0

```
.Appearance = 0     ' flat
.Appearance = 1     ' sunken
```

Visual C++ 6.0

```
m_ctrl.SetAppearance(0);     // flat
m_ctrl.SetAppearance(1);     // sunken
```

特記事項

永続化プロパティ。

参照

3.1.3 BorderStyle

SavoyGem コントロールの境界線スタイルを取得または設定します。

値	説明
0	境界線なし
1	境界線あり

構文

Visual Basic 6.0

```
BorderStyle As Integer
```

Visual C++ 6.0

```
short GetBorderStyle()
void SetBorderStyle(short)
```

使用例

Visual Basic 6.0

```
.BorderStyle = 0     ' no border
.BorderStyle = 1     ' with border
```

Visual C++ 6.0

```
m_ctrl.SetBorderStyle(0);     // no border
m_ctrl.SetBorderStyle(1);     // with border
```

特記事項

永続化プロパティ。

参照

3.1.4 CEIDCount

登録されている CEID の数を取得します。0 の場合は一つも登録されていないことを意味します。

構文

Visual Basic 6.0

```
CEIDCount As Long
```

Visual C++ 6.0

```
long GetCEIDCount()
```

使用例

Visual Basic 6.0

```
Dim lCEIDCount As Long
lCEIDCount = .CEIDCount
```

Visual C++ 6.0

```
long lCEIDCount = m_ctrl.GetCEIDCount();
```

特記事項

読み出し専用プロパティ。

CEIDCount プロパティは登録されている CEID の数を返すので、インデックスとして使用可能な値は 0 から(CEIDCount - 1)までとなります。これを ToCEID メソッドを使って CEID に変換します。

参照

3.1.5 CommunicationState

GEM 通信状態を取得または設定します。通信状態は以下のいずれかです。

値	説明
0	通信が無効
1	通信が中断
2	通信している

構文

Visual Basic 6.0

```
CommunicationState As Integer
```

Visual C++ 6.0

```
short GetCommunicationState()
void SetCommunicationState(short)
```

使用例

Visual Basic 6.0

```
.Server = False
.IPAddress = “127.0.0.1”
.PortNumber = 5001
.MyPortNumber = 0
.CommunicationState = 1
```

Visual C++ 6.0

```
.m_ctrl.SetServer(false);
.m_ctrl.SetIPAddress(“127.0.0.1”);
.m_ctrl.SetPortNumber(5001);
.m_ctrl.SetMyPortNumber(0);
.m_ctrl.SetCommunicationState(1);
```

特記事項

参照

3.1.6 ControlState

GEM コントロール状態を取得または設定します。コントロール状態は以下のいずれかです。

値	説明
0	装置オフライン
1	オンライン試行
2	ホストオフライン
3	オンラインローカル
4	オンラインリモート

構文

Visual Basic 6.0

```
ControlState As Integer
```

Visual C++ 6.0

```
short GetControlState()
void SetControlState(short)
```

使用例

Visual Basic 6.0

```
Dim nControlState As Integer
nControlState = .ControlState
```

Visual C++ 6.0

```
short sControlState = m_ctrl.GetControlState();
```

特記事項

ある特定のコントロール状態の遷移は許可されない場合があります。このことからオンライン／オフラインの切り替えには ControlStateSwitch プロパティを使用してください。

参照

3.1.7 ControlStateSwitch

コントロール状態スイッチを設定します。

値	説明
True	オフラインからオンラインへの切り替えを試みる
False	オンラインからオフラインへの切り替え

ある特定のコントロール状態の遷移は許可されない場合があります。このことからオンライン／オフラインの切り替えには ControlStateSwitch プロパティを使用してください。

構文

Visual Basic 6.0

```
ControlStateSwitch As Boolean
```

Visual C++ 6.0

```
void SetControlStateSwitch(BOOL)
```

使用例

Visual Basic 6.0

```
.ControlStateSwitch = True
```

Visual C++ 6.0

```
m_ctrl.SetControlStateSwitch(true);
```

特記事項

書き込み専用プロパティ。

実際に状態遷移した場合は ControlStateChanged イベントが発生します。

参照

3.1.8 DataBakCount

SavoyGem データファイルのバックアップファイルの数を取得または設定します。データファイルが更新される際には、SavoyGem コントロールはファイル名をリネームし、新たに空のデータファイルを作成します。もしバックアップファイルの数が DataBakCount プロパティに達した場合、SavoyGem コントロールは古いバックアップファイルから消去します。

構文

Visual Basic 6.0

```
DataBakCount As Integer
```

Visual C++ 6.0

```
short GetDataBakCount()
void SetDataBakCount(short)
```

使用例

Visual Basic 6.0

```
.DataBakCount = 10
```

Visual C++ 6.0

```
m_ctrl.SetDataBakCount(10);
```

特記事項

永続化プロパティ。

参照

3.1.9 DataFileName

SavoyGem データファイル名を取得または設定します。

構文

Visual Basic 6.0

```
DataFileName As String
```

Visual C++ 6.0

```
CString GetDataFileName()
void SetDataFileName(LPCTSTR)
```

使用例

Visual Basic 6.0

```
.DataFileName = ".¥SavoyGem.bop"
.LoadData
```

Visual C++ 6.0

```
m_ctrl.SetDataFileName(".¥SavoyGem.bop");
m_ctrl.LoadData();
```

特記事項

永続化プロパティ。

参照

LoadData メソッド

3.1.10 DeviceID

デバイス ID を取得または設定します。デバイス ID は SECS-II ヘッダの先頭 2 ビット目から 15 ビットです。

HSMS データメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	W ストリーム番号
4	ファンクション番号
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

HSMS コントロールメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	
4	
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

構文

Visual Basic 6.0

```
DeviceID As Long
```

Visual C++ 6.0

```
long GetDeviceID()
void SetDeviceID(long)
```

使用例

Visual Basic 6.0

```
.DeviceID = 1
```

Visual C++ 6.0

```
m_ctrl.SetDeviceID(1);
```

特記事項

永続化プロパティ。

DeviceID は SessionID の最上位ビットを除いた 15 ビットです。

参照

3.1.11 DiscardDuplicatedBlock

SavoyGem コントロールが二重ブロックを破棄するかどうかを取得または設定します。もしこのプロパティが true のときに SavoyGemコントロールが直前のメッセージと同一ヘッダのメッセージを受信した場合、SavoyGemコントロールはそのようなメッセージを二重ブロックとみなして無視します。

値	説明
True	二重ブロックを破棄する。
False	二重ブロックを破棄しない。

構文

Visual Basic 6.0

```
DiscardDuplicatedBlock As Boolean
```

Visual C++ 6.0

```
BOOL GetDiscardDuplicatedBlock()
void SetDiscardDuplicatedBlock(BOOL)
```

使用例

Visual Basic 6.0

```
.DiscardDuplicatedBlock = True
```

Visual C++ 6.0

```
m_ctrl.SetDiscardDuplicatedBlock(true);
```

特記事項

永続化プロパティ。

参照

3.1.12 Function

SECS-II ヘッダのファンクション番号を取得または設定します。

HSMS データメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	W ストリーム番号
4	ファンクション番号
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

構文

Visual Basic 6.0

```
Function As Integer
```

Visual C++ 6.0

```
short GetFunction()
void SetFunction(short)
```

使用例

Visual Basic 6.0

```
If .Stream = 1 AND .Function = 13 Then
    ‘ S1F13
    ...
```

Visual C++ 6.0

```
If(m_ctrl.GetStream()==1 && m_ctrl.GetFunction()==13)
{
    // S1F13
    ...
```

特記事項

参照

3.1.13 Host

SavoyGem コントロールの役割を取得または設定します。SavoyGem コントロールは装置として機能するため、このプロパティは常に False であるべきです。

値	説明
False	装置
True	ホスト

構文

Visual Basic 6.0

```
Host As Boolean
```

Visual C++ 6.0

```
BOOL GetHost()
void SetHost(BOOL)
```

使用例

Visual Basic 6.0

```
.Host = False
```

Visual C++ 6.0

```
m_ctrl.SetHost(false);
```

特記事項

永続化プロパティ。

参照

3.1.14 IniFileName

設定を読み書きするための INI ファイル名を取得または設定します。もし INI ファイル名がフルパス名かフォルダの相対参照を含む場合は、INI ファイルは指定された場所に作成されます。そうでない場合は、Windows のシステムフォルダに INI ファイルが作成されます。この理由からフォルダ名と共に使用するのが推奨されます。もしカレントディレクトリが INI ファイルの場所であれば、「./」を先頭に付けます。

「/」または「¥」をフォルダ名の区切り文字として使用できます。

構文

Visual Basic 6.0

```
IniFileName As String
```

Visual C++ 6.0

```
CString GetIniFileName()
void SetIniFileName(LPCTSTR)
```

使用例

Visual Basic 6.0

```
.IniFileName = “./SavoyGem.ini”
.LoadIniFile
```

Visual C++ 6.0

```
m_ctrl.SetIniFileName(“./SavoyGem.ini”);
m_ctrl.LoadIniFile();
```

特記事項

永続化プロパティ。

SavoyGem コントロールによっていくつかの INI セクションが使われます。アプリケーションで同じ INI ファイルを参照する場合は、名前の重複に注意が必要です。

参照

3.1.15 IPAddress

HSMS 接続のパッシブエンティティのコンピュータの IP アドレスを取得または設定します。Server プロパティが true のときは、接続を待ち受けていて IP アドレスは不要なので、このプロパティは無視されます。

ローカルコンピュータ（自分自身のコンピュータ）に接続する場合は 127.0.0.1 を使用してください。IP アドレスの代わりにコンピュータ名を使用することもできます。

構文

Visual Basic 6.0

```
IPAddress As String
```

Visual C++ 6.0

```
CString GetIPAddress()
void SetIPAddress(LPCTSTR)
```

使用例

Visual Basic 6.0

```
.Server = False
.IPAddress = “127.0.0.1”
.PortNumber = 5001
.MyPortNumber = 0
.CommunicationState = 1
```

Visual C++ 6.0

```
.m_ctrl.SetServer(false);
.m_ctrl.SetIPAddress(“127.0.0.1”);
.m_ctrl.SetPortNumber(5001);
.m_ctrl.SetMyPortNumber(0);
.m_ctrl.SetCommunicationState(1);
```

特記事項

永続化プロパティ。

参照

3.1.16 Log

ログの有効・無効を取得または設定します。もし Log プロパティが True の場合、処理情報はログファイルに書き込まれます。もし Log プロパティが False の場合はログファイルには記録されません。

値	説明
True	ログファイルに記録する
False	ログファイルに記録しない

構文

Visual Basic 6.0

```
Log As Boolean
```

Visual C++ 6.0

```
BOOL GetLog()
void SetLog(BOOL)
```

使用例

Visual Basic 6.0

```
.LogFileName = "./SavoyGem"
.LogSize = 1024
.LogBakCount = 10
.LogCommunication = False
.Log = True
```

Visual C++ 6.0

```
m_ctrl.SetLogFileName("./SavoyGem");
m_ctrl.SetLogSize(1024);
m_ctrl.SetLogBakCount(10);
m_ctrl.SetLogCommunication(false);
m_ctrl.SetLog(true);
```

特記事項

永続化プロパティ。

参照

3.1.17 LogBakCount

ログのバックアップファイルの数を取得または設定します。もしログファイルの実際のファイルサイズが LogSize プロパティを超えた場合、SavoyGem コントロールはファイル名をリネームし、新たに空のログファイルを作成します。もしバックアップファイルの数が LogBakCount プロパティに達した場合、SavoyGem コントロールは古いバックアップファイルから消去します。

構文

Visual Basic 6.0

```
LogBakCount As Integer
```

Visual C++ 6.0

```
short GetLogBakCount()
void SetLogBakCount(short)
```

使用例

Visual Basic 6.0

```
.LogFileName = "./SavoyGem"
.LogSize = 1024
.LogBakCount = 10
.LogCommunication = False
.Log = True
```

Visual C++ 6.0

```
m_ctrl.SetLogFileName("./SavoyGem");
m_ctrl.SetLogSize(1024);
m_ctrl.SetLogBakCount(10);
m_ctrl.SetLogCommunication(false);
m_ctrl.SetLog(true);
```

特記事項

永続化プロパティ。

参照

3.1.18 LogCommunication

通信部分のログが有効かどうかを取得または設定します。

構文

Visual Basic 6.0

```
LogCommunication As Boolean
```

Visual C++ 6.0

```
BOOL GetLogCommunication()
void SetLogCommunication(BOOL)
```

使用例

Visual Basic 6.0

```
.LogFileName = "./SavoyGem"
.LogSize = 1024
.LogBakCount = 10
.LogCommunication = False
.Log = True
```

Visual C++ 6.0

```
m_ctrl.SetLogFileName("./SavoyGem");
m_ctrl.SetLogSize(1024);
m_ctrl.SetLogBakCount(10);
m_ctrl.SetLogCommunication(false);
m_ctrl.SetLog(true);
```

特記事項

永続化プロパティ。

参照

3.1.19 LogFileName

ログファイル名を取得または設定します。

構文

Visual Basic 6.0

```
LogFileName As String
```

Visual C++ 6.0

```
CString GetLogFileName()
void SetLogFileName(LPCTSTR)
```

使用例

Visual Basic 6.0

```
.LogFileName = "./SavoyGem"
.LogSize = 1024
.LogBakCount = 10
.LogCommunication = False
.Log = True
```

Visual C++ 6.0

```
m_ctrl.SetLogFileName("./SavoyGem");
m_ctrl.SetLogSize(1024);
m_ctrl.SetLogBakCount(10);
m_ctrl.SetLogCommunication(false);
m_ctrl.SetLog(true);
```

特記事項

永続化プロパティ。

ログファイル名を指定するときは、拡張子を付加しないでください。自動的に拡張子「.log」が付加されます。

参照

3.1.20 LogicalConnection

論理接続を取得または設定します。論理接続は下記のいずれかとなります。

値	説明
0	汎用モデル。
1	GEM モデル。

構文

Visual Basic 6.0

```
LogicalConnection As Integer
```

Visual C++ 6.0

```
short GetLogicalConnection()
void SetLogicalConnection(short)
```

使用例

Visual Basic 6.0

```
.LogicalConnection = 1
```

Visual C++ 6.0

```
m_ctrl.SetLogicalConnection(1);
```

特記事項

永続化プロパティ。

SavoyGem は GEM 対応の製品ですので、LogicalConnection プロパティの値は常に 1 にします。

参照

3.1.21 LogSize

ログファイルのサイズを取得または設定します。もしログファイルの実際のファイルサイズが LogSize プロパティを超えた場合、SavoyGem コントロールはファイル名をリネームし、新たに空のログファイルを作成します。もしバックアップファイルの数が LogBakCount プロパティに達した場合、SavoyGem コントロールは古いバックアップファイルから消去します。

構文

Visual Basic 6.0

```
LogSize As Long
```

Visual C++ 6.0

```
long GetLogSize()
void SetLogSize(long)
```

使用例

Visual Basic 6.0

```
.LogFileName = "./SavoyGem"
.LogSize = 1024
.LogBakCount = 10
.LogCommunication = False
.Log = True
```

Visual C++ 6.0

```
m_ctrl.SetLogFileName("./SavoyGem");
m_ctrl.SetLogSize(1024);
m_ctrl.SetLogBakCount(10);
m_ctrl.SetLogCommunication(false);
m_ctrl.SetLog(true);
```

特記事項

永続化プロパティ。

参照

3.1.22 Msg

メッセージデータを取得または設定します。メッセージデータ形式は 16 進法の ASCII 文字列です。

構文

Visual Basic 6.0

```
Msg As String
```

Visual C++ 6.0

```
CString GetMsg()
void SetMsg(LPCTSTR)
```

使用例

Visual Basic 6.0

```
.WorkSpace = 0
.Reply = False
.SML = “s1f13w{<a’Savoy’><a’1.00’>}”
Dim strMsg As String
strMsg = .Msg
```

Visual C++ 6.0

```
m_ctrl.SetWorkSpace(0);
m_ctrl.SetReply(false);
m_ctrl.SetSml(“s1f13w{<a’Savoy’><a’1.00’>}”);
CString strMsg = m_ctrl.GetMsg();
```

特記事項

参照

3.1.23 MyPortNumber

HSMS 接続のローカルポート番号を取得または設定します。もし SavoyGem コントロールがアクティブエンティティとして動作している場合、このプロパティは 0 であるべきです。さもなければ接続は TCP/IP レベルでのタイムアウトが発生するまで再接続ができなくなります。

SavoyGem コントロールがパッシブエンティティとして動作している場合、クライアントに対して公開するポート番号となります。

構文

Visual Basic 6.0

```
MyPortNumber As Long
```

Visual C++ 6.0

```
long GetMyPortNumber()
void SetMyPortNumber(long)
```

使用例

Visual Basic 6.0

```
.Server = False
.IPAddress = “127.0.0.1”
.PortNumber = 5001
.MyPortNumber = 0
.CommunicationState = 1
```

Visual C++ 6.0

```
.m_ctrl.SetServer(false);
.m_ctrl.SetIPAddress(“127.0.0.1”);
.m_ctrl.SetPortNumber(5001);
.m_ctrl.SetMyPortNumber(0);
.m_ctrl.SetCommunicationState(1);
```

特記事項

永続化プロパティ。

参照

3.1.24 Node

メッセージ内容を操作するためのノードを取得または設定します。ノードは“/”（スラッシュ）とノード番号と“[”“]”（カギカッコ）で構成されます。ノード番号は 1 から始まる数字です。ノードが空文字列の場合はルートが指定されたとみなされます。

構文

Visual Basic 6.0

```
Node As String
```

Visual C++ 6.0

```
CString GetNode()
void SetNode(LPCTSTR)
```

使用例

次のような SML 構造のメッセージを作ってみます。

```
s1f13w
{
    <a'Savoy'>
    <a'1'>
}
```

既にSavoySecsIIコントロールにメッセージ構造が入っているかもしれません。まず最初に全体を書き換える必要があるため、ルートノードを指定します。

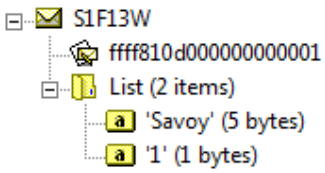
Visual Basic 6.0

```
.Node = ""
.SML = "s1f13w{<a'Savoy'><a'1'>}"
```

Visual C++ 6.0

```
m_ctrl.SetNode("");
m_ctrl.SetSml("s1f13w{<a'Savoy'><a'1'>}");
```

このコードを実行すると以下のような構造のメッセージが作成されます。



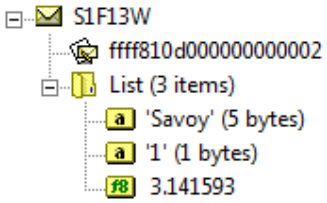
この構造に 3 番目のノードを追加したい場合は、Node に 3 を指定します。

Visual Basic 6.0

```
.Node = "3"
.SML = "<f8 3.1415926535>"
```

Visual C++ 6.0

```
m_ctrl.SetNode("3");
m_ctrl.SetSml("<f8 3.1415926535>");
```



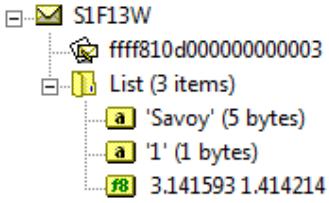
3 番目のノードを配列にしたい場合は、同じ型で数値を指定します。

Visual Basic 6.0

```
.Node = "3"
.SML = "<f8 141421356>"
```

Visual C++ 6.0

```
m_ctrl.SetNode("3");
m_ctrl.SetSml("<f8 141421356>");
```



ここで 3 番目のノードの値を NodeValue プロパティを使って読み出すと、配列の要素はスペース文字で分離され、”3.141593 1.414214”という文字列が返ります。配列の個々の要素を取り出したい場合は、以下のように [ ] を使ってインデックス指定します。インデックスは C/C++/Java/C#言語のように、0 から始まります。

Visual Basic 6.0

```
.Node = "3[0]"
.Node = "3[1]"
```

Visual C++ 6.0

```
m_ctrl.SetNode("3[0]");
m_ctrl.SetNode("3[1]");
```

上記”3[0]”では、”3.141593”という文字列が返り、”3[1]”では”1.414214”という文字列が返ります。

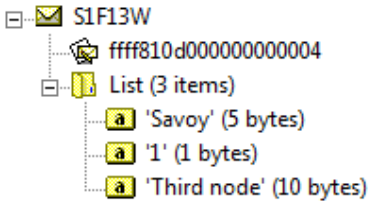
もし型の異なるノードに変更したい場合は、違う型で指定します。

Visual Basic 6.0

```
.Node = "3"
.SML = "<a’Third node’>"
```

Visual C++ 6.0

```
m_ctrl.SetNode("3");
m_ctrl.SetSml("<a'Third node'>");
```



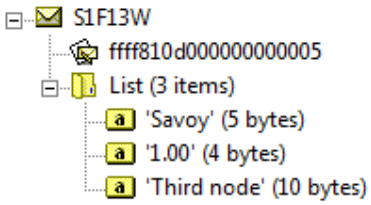
文字列を連結したい場合は、同じ型で文字列を指定します。

Visual Basic 6.0

```
.Node = "2"
.SML = "<a'.00'>"
```

Visual C++ 6.0

```
m_ctrl.SetNode("2");
m_ctrl.SetSml("<a'.00'>");
```



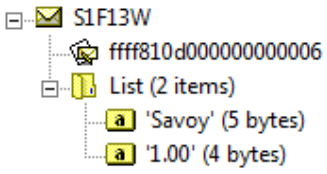
SML に空文字を指定すると、そのノードを消去します。

Visual Basic 6.0

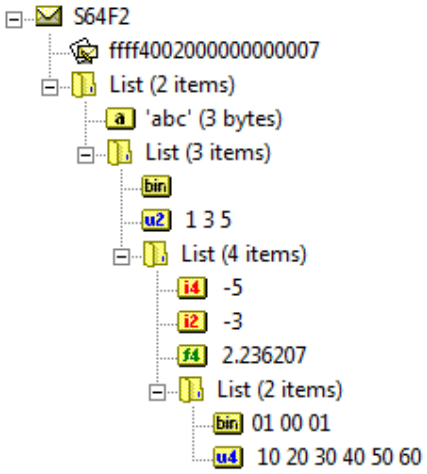
```
.Node = "3"
.SML = ""
```

Visual C++ 6.0

```
m_ctrl.SetNode("3");
m_ctrl.SetSml("");
```



複雑な構造のメッセージでも Node プロパティを使えば、ピンポイントで値を読み出すことができます。



このメッセージの最後に u4 型の 6 個の配列があります。この 4 番目の値を読み出してみましょう。それにはノードの位置を特定する必要があります。最初から見ていくと、リストの 2 番目、その中の 3 番目、その中の 4 番目、その中の 2 番目、配列の 4 番目だということが分かります。

Visual Basic 6.0

```
.Node = "2/3/4/2[3]"
```

Visual C++ 6.0

```
m_ctrl.SetNode("2/3/4/2[3]");
```

この例ではノードに”2/3/4/2[3]”を指定しています。NodeValue プロパティを読み出すと”40”が戻ります。

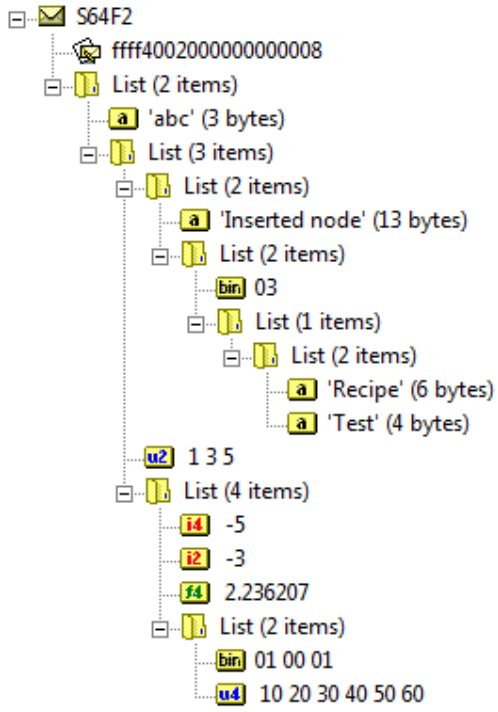
複雑な構造の SML をノードに指定することもできます。

Visual Basic 6.0

```
.Node = "2/1"
.SML = " {<a'Inserted node'>{<b 3>{{<a'Recipe'><a'Test'>}}}}"
```

Visual C++ 6.0

```
m_ctrl.SetNode("2/1");
m_ctrl.SetSml("{<a'Inserted node'>{<b 3>{{<a'Recipe'><a'Test'>}}}}");
```



特記事項

ノードは Windows のフォルダ構造に例えることができます（説明の中で『ノード』と記述されている部分を『フォルダ』と置き換えると理解しやすいと思います）。

ノードを作成する場合にはまずノード指定してそれに対して追加命令を発行します。メッセージボディ全体を書き換える場合は、ルートノードを指定します。

参照

3.1.25 NodeCount

サブアイテムの個数を取得します。もしノードプロパティがリスト型の場合、このプロパティはサブノードの数を意味します。それ以外は配列の個数です。

構文

Visual Basic 6.0

```
NodeCount As Long
```

Visual C++ 6.0

```
long GetNodeCount()
```

使用例

Visual Basic 6.0

```
.Node = ""
.SML = “{{<b 1>}}”
.Node = "99"
Text1.Text = "NodeCount = " + Format$(.NodeCount)
```

Visual C++ 6.0

```
m_ctrl.SetNode("");
m_ctrl.SetSml(“{{<b 1>}}”);
m_ctrl.SetNode("99");
m_text1.Format("NodeCount = %d",m_ctrl.GetNodeCount());
```

特記事項

読み出し専用プロパティ。

参照

3.1.26 NodeType

ノードタイプを取得します。

値	列挙型	説明
1	SecsTypeList	リスト
2	SecsTypeBinary	バイナリ
3	SecsTypeBoolean	ブーリアン
4	SecsTypeAscii	ASCII 文字列
5	SecsTypeJis	JIS 8 文字列
6	SecsTypeLong8	8 バイト符号付き整数
7	SecsTypeChar	1 バイト符号付き整数
8	SecsTypeShort	2 バイト符号付き整数
9	SecsTypeLong	4 バイト符号付き整数
10	SecsTypeDouble	8 バイト浮動小数点数
11	SecsTypeFloat	4 バイト浮動小数点数
12	SecsTypeDWord8	8 バイト符号なし整数
13	SecsTypeByte	1 バイト符号なし整数
14	SecsTypeWord	2 バイト符号なし整数
15	SecsTypeDWord	4 バイト符号なし整数
16	SecsTypeAscii2	2 バイト ASCII 文字列

構文

Visual Basic 6.0

```
NodeType As Integer
```

Visual C++ 6.0

```
short GetNodeType()
```

使用例

Visual Basic 6.0

```
.Node = "1/2"
Text1.Text = "NodeType = " + Format$(.NodeType)
```

Visual C++ 6.0

```
m_ctrl.SetNode("1/2");
m_text1.Format("NodeType = %d",m_ctrl.GetNodeType());
```

特記事項

読み出し専用プロパティ。

参照

3.1.27 NodeValue

ノードの値を取得します。もしノードが数値型の場合、値は 10 進法文字列に変換されます。

構文

Visual Basic 6.0

```
NodeValue As String
```

Visual C++ 6.0

```
CString GetNodeValue()
```

使用例

Visual Basic 6.0

```
If CInt(.NodeValue) = 201 Then
   Text1.Text = "CEID is 201"
End If
```

Visual C++ 6.0

```
if(::atoi(m_ctrl.GetNodeValue())==201)
   m_text1 = "CEID is 201";
```

特記事項

読み出し専用プロパティ。

参照

3.1.28 NodeValueHex

ノードの値を 16 進法表現で取得します。

構文

Visual Basic 6.0

```
NodeValueHex As String
```

Visual C++ 6.0

```
CString GetNodeValueHex()
```

使用例

Visual Basic 6.0

```
If .NodeValueHex = "ff" Then
   Text1.Text = "Value is 0xff"
End If
```

Visual C++ 6.0

```
if(m_ctrl.GetNodeValueHex()=="ff")
   m_text1="Value is 0xff";
```

特記事項

読み出し専用プロパティ。

参照

3.1.29 OfflineRequest

S1F15 オフライン要求(ROFL)を受け入れるかどうかを取得または設定します。OfflineRequest プロパティが True の時はオフライン要求を受け入れ、S1F16 オフライン要求確認(OFLA)で OFLACK = 0 を返します。False の時は要求を拒否し、OFLACK = 1 を返します。

値	説明
False	S1F15 オフライン要求(ROFL)を受け入れない
True	S1F15 オフライン要求(ROFL)を受け入れる

構文

Visual Basic 6.0

```
OfflineRequest As Boolean
```

Visual C++ 6.0

```
BOOL GetOfflineRequest()
void SetOfflineRequest(BOOL)
```

使用例

Visual Basic 6.0

```
.OfflineRequest = True
```

Visual C++ 6.0

```
m_ctrl.SetOfflineRequest(true);
```

特記事項

参照

3.1.30 OnlineRequest

S1F17 オンライン要求(RONL)を受け入れるかどうかを取得または設定します。OnlineRequest プロパティが True の時はオンライン要求を受け入れ、S1F18 オンライン要求確認(ONLA)で ONLACK = 0 を返します。False の時は要求を拒否し、ONLACK = 1 を返します。

値	説明
False	S1F17 オンライン要求(RONL)を受け入れない
True	S1F17 オンライン要求(RONL)を受け入れる

構文

Visual Basic 6.0

```
OnlineRequest As Boolean
```

Visual C++ 6.0

```
BOOL GetOnlineRequest()
void SetOnlineRequest(BOOL)
```

使用例

Visual Basic 6.0

```
.OnlineRequest = True
```

Visual C++ 6.0

```
m_ctrl.SetOnlineRequest(true);
```

特記事項

参照

3.1.31 Reply

ワークスペースのモードを取得または設定します。もしこのプロパティが True の場合、バッファ 1 が選択されます。もしこのプロパティが False の場合、バッファ 0 が選択されます。

値	説明
False	バッファ 0 を選択する
True	バッファ 1 を選択する

構文

Visual Basic 6.0

```
Reply As Boolean
```

Visual C++ 6.0

```
BOOL GetReply()
void SetReply(BOOL)
```

使用例

Visual Basic 6.0

```
.WorkSpace = 0
.Reply = False
.SML = “s1f13w{<a’Savoy’><a’1.00’>}”
Dim strMsg As String
strMsg = .Msg
```

Visual C++ 6.0

```
m_ctrl.SetWorkSpace(0);
m_ctrl.SetReply(false);
m_ctrl.SetSml(“s1f13w{<a’Savoy’><a’1.00’>}”);
CString strMsg = m_ctrl.GetMsg();
```

特記事項

参照

WorkSpace プロパティ

3.1.32 PhysicalConnection

物理接続が有効かどうかを取得または設定します。このプロパティが True の時は物理接続モデルが有効となり、False の時は無効となります。

値	説明
False	物理接続モデルが無効
True	物理接続モデルが有効

構文

Visual Basic 6.0

```
PhysicalConnection As Boolean
```

Visual C++ 6.0

```
BOOL GetPhysicalConnection()
void SetPhysicalConnection(BOOL)
```

使用例

Visual Basic 6.0

Visual C++ 6.0

特記事項

このプロパティを True にすることで通信ポートがオープンされ、通信が可能になります。ただし実際に通信ポートが開かれるのは、プロパティを True にセットした瞬間ではなく、少しだけ後です。このため以下のようなプログラムを書いても、ポートがオープンできたかどうかを知ることはできません。

Visual Basic 6.0

```
.PhysicalConnection = True
If .PhysicalConnection Then
    ...
```

Visual C++ 6.0

```
m_ctrl.SetPhysicalConnection(true);
If(m_ctrl.GetPhysicalConnection())
{
    ...
```

参照

3.1.33 PortNumber

HSMS のポート番号を取得または設定します。

構文

Visual Basic 6.0

```
PortNumber As Long
```

Visual C++ 6.0

```
long GetPortNumber()
void SetPortNumber(long)
```

使用例

Visual Basic 6.0

```
.Server = False
.IPAddress = “127.0.0.1”
.PortNumber = 5001
.MyPortNumber = 0
.CommunicationState = 1
```

Visual C++ 6.0

```
.m_ctrl.SetServer(false);
.m_ctrl.SetIPAddress(“127.0.0.1”);
.m_ctrl.SetPortNumber(5001);
.m_ctrl.SetMyPortNumber(0);
.m_ctrl.SetCommunicationState(1);
```

特記事項

永続化プロパティ。

パッシブエンティティの場合は相手に接続しにいくわけではないので、指定する必要はありません（無視されます）。アクティブエンティティの場合は接続先のサーバポート番号を指定します。

参照

3.1.34 PType

SECS-II ヘッダのプレゼンテーションタイプを取得または設定します。

HSMS データメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	W ストリーム番号
4	ファンクション番号
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

HSMS コントロールメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	
4	
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

構文

Visual Basic 6.0

```
PType As Integer
```

Visual C++ 6.0

```
short GetPType()
void SetPType(short)
```

使用例

Visual Basic 6.0

```
If .PType <> 0 Then
   MsgBox "Invalid P-type!"
End If
```

Visual C++ 6.0

```
if(m_ctrl.GetPType()!=0)
   MessageBox("Invalid P-type!");
```

特記事項

現在は SECS-II のみが規定されているため、PType プロパティは常に 0 です。

参照

3.1.35 Server

エンティティのタイプを取得または設定します。もしこのプロパティが True の場合、パッシブエンティティを意味します。もしこのプロパティが False の場合、アクティブエンティティを意味します。

値	説明
False	アクティブエンティティ（クライアント）
True	パッシブエンティティ（サーバ）

構文

Visual Basic 6.0

```
Server As Boolean
```

Visual C++ 6.0

```
BOOL GetServer()
void SetServer(BOOL)
```

使用例

Visual Basic 6.0

```
.Server = False
.IPAddress = “127.0.0.1”
.PortNumber = 5001
.MyPortNumber = 0
.CommunicationState = 1
```

Visual C++ 6.0

```
.m_ctrl.SetServer(false);
.m_ctrl.SetIPAddress(“127.0.0.1”);
.m_ctrl.SetPortNumber(5001);
.m_ctrl.SetMyPortNumber(0);
.m_ctrl.SetCommunicationState(1);
```

特記事項

永続化プロパティ。

参照

3.1.36 SessionID

HSMS のセッション ID を取得または設定します。セッション ID は SECS-II ヘッダの先頭 16 ビットです。

HSMS データメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	W ストリーム番号
4	ファンクション番号
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

HSMS コントロールメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	
4	
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

構文

Visual Basic 6.0

```
SessionID As Long
```

Visual C++ 6.0

```
long GetSessionID()
void SetSessionID(long)
```

使用例

Visual Basic 6.0

```
.SessionID = &HFFFF
```

Visual C++ 6.0

```
m_ctrl.SetSessionID(0xffff);
```

特記事項

DeviceID は SessionID の最上位ビットを除いた 15 ビットです。

参照

3.1.37 SML

SECS-II メッセージを SML 文字列として取得または設定します。

構文

Visual Basic 6.0

```
SML As String
```

Visual C++ 6.0

```
CString GetSml()
void SetSml(LPCTSTR)
```

使用例

Visual Basic 6.0

```
.WorkSpace = 0
.Reply = False
.SML = “s1f13w{<a’Savoy’><a’1.00’>}”
Dim strMsg As String
strMsg = .Msg
```

Visual C++ 6.0

```
m_ctrl.SetWorkSpace(0);
m_ctrl.SetReply(false);
m_ctrl.SetSml(“s1f13w{<a’Savoy’><a’1.00’>}”);
CString strMsg = m_ctrl.GetMsg();
```

特記事項

SML プロパティにセットする文字列の構文は以下のようになっています。

■一般的な注意

ホワイトスペース（スペース、タブ、改行、復改コード）は区切り文字としての意味しかありません。このため適度にタブや改行コードを挿入することで見易くすることができます。ただしコメント中および文字列中は文字として扱われます。

アスタリスク（*）から行末まではコメントとなります。ただし文字列中のアスタリスクは除きます。

整数は 0～9 までの文字とマイナス（-）から構成されます。16 進数で記述したい場合は’0x’を先頭に付加します。この場合は a～f と A～F までの文字も使用できます。小数は’0.9’を’.9’というように’0’を省略して記述することもできます。指数表現も可能です。また予約語として true（=1）と false（=0）を使うこともできます。

文字列はシングルクォーテーション（’）で囲まれた範囲です。文字列中には改行コードとシングルクォーテーション自身を含めることはできません。このためどうしてもこれらの文字を入れたい場合は、0x0a などのように16進数表現を併用します。

説明文中の青色太字部分はその文字を記述することを表します。基本的にこれらの文字は大文字でも小文字でも構いません。斜体字はそれぞれの説明を参照してください。また[ ]で囲まれた部分は省略することができます。

■構文

[sxx**f**yy[w]] Body

項目	説明
xx	ストリーム番号。文字’s’、’f’との間にはスペースを入れません。
yy	ファンクション番号。文字’f’、’w’との間にはスペースを入れません。
w	ウェイトビット。指定する場合は’w’と記述します。省略可能です。
Body	メッセージのボディ。

ストリーム、ファンクション、ウェイトビットはひとかたまりで認識するため、これらの間にスペースや改行コードを入れないようにします。
またストリーム、ファンクションを全て省略してメッセージボディのみを記述することもできます。

■メッセージボディ

メッセージのボディは階層構造になっています。

リスト

{[l [NumOfItem]] Body**}**
<[l [NumOfItem]] Body**>**

項目	説明
NumOfItem	リストの数です。SECSIM との互換性のためだけに用意されています。この数字は無視されます。
Body	メッセージのボディ。他のアイテムを並べることができます。

アスキー文字列

<a [Strings]**>**

項目	説明
Strings	文字列です。

長い文字列は分割して記述することもできます。また直接文字コードを記述することもできます。例えば

```
<a ‘ABC’ ‘DEF’ ‘012’ 0x33 ‘4’ 53 54 ‘789’>
```

これは

```
<a ‘ABCDEF0123456789’>
```

と同じです。

2 バイト文字列

```
<a2 [Strings]>
```

項目	説明
Strings	2 バイト文字列です。現在のバージョンでは DBCS にのみ対応しています。

JIS8 文字列

```
<j [Strings]>
```

アスキー型と同じように扱われます。

項目	説明
Strings	文字列です。

長い文字列は分割して記述することもできます。また直接文字コードを記述することもできます。例えば

```
<j ‘ABC’ ‘DEF’ ‘012’ 0x33 ‘4’ 53 54 ‘789’>
```

これは

```
<j ‘ABCDEF0123456789’>
```

と同じです。

整数

```
<i1 [Numbers]>
<i2 [Numbers]>
<i4 [Numbers]>
<i8 [Numbers]>
<u1 [Numbers]>
<u2 [Numbers]>
<u4 [Numbers]>
<u8 [Numbers]>
```

項目	説明
Numbers	数値です。それぞれ以下の意味となります。

型	説明
i1	符号付き 8 ビット整数
i2	符号付き 16 ビット整数
i4	符号付き 32 ビット整数
i8	符号付き 64 ビット整数
u1	符号なし 8 ビット整数
u2	符号なし 16 ビット整数
u4	符号なし 32 ビット整数
u8	符号なし 64 ビット整数

いくつかの数字を並べて記述することもできます。この場合は配列となります。例えば

```
<i1 1 0x02 3>
```

のように記述することができます。

現在のバージョンでは i8 と u8 に巨大な値を入れることはできません。

浮動小数点数

<f4 [*FNumbers*]**>**
<f8 [*FNumbers*]**>**

項目	説明
FNumbers	浮動小数点数です。それぞれ以下の意味となります。

型	説明
f4	32 ビット浮動小数点数
f8	64 ビット浮動小数点数

例えば

```
<f4 0 1.0 3.14>
```

のように記述します。

バイナリ

<b [*Numbers*]**>**

項目	説明
Numbers	数値です。

例えば

```
<b 0xff 0x3e 255 0>
```

のように記述します。

ブーリアン

<bool [*Numbers*]**>**
<boolean [*Numbers*]**>**

項目	説明
Numbers	数値です。

例えば

```
<bool true false 1 0>
```

のように記述します。

参照

3.1.38 Stream

SECS-II ヘッダのストリーム番号を取得または設定します。

HSMS データメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	W ストリーム番号
4	ファンクション番号
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

構文

Visual Basic 6.0

```
Stream As Integer
```

Visual C++ 6.0

```
short GetStream()
void SetStream(short)
```

使用例

Visual Basic 6.0

```
If .Stream = 1 AND .Function = 13 Then
    ‘ S1F13
    ...
```

Visual C++ 6.0

```
If(m_ctrl.GetStream()==1 && m_ctrl.GetFunction()==13)
{
    // S1F13
    ...
```

特記事項

参照

3.1.39 SType

SECS-II ヘッダのセッションタイプを取得または設定します。

値	説明
0	データメッセージ
1	Select.Req
2	Select.Rsp
3	Deselect.Req
4	Deselect.Rsp
5	LinkTest.Req
6	LinkTest.Rsp
7	Reject.Req
8	（未使用）
9	Separate.Req
10	（未使用）
11-127	
128-255	

HSMS データメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	W ストリーム番号
4	ファンクション番号
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

HSMS コントロールメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	
4	
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

構文

Visual Basic 6.0

```
SType As Integer
```

Visual C++ 6.0

```
short GetSType()
void SetSType(short)
```

使用例

Visual Basic 6.0

Visual C++ 6.0

特記事項

参照

3.1.40 SystemBytes

SECS-II ヘッダのシステムバイトを取得または設定します。

HSMS データメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	W ストリーム番号
4	ファンクション番号
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

HSMS コントロールメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	
4	
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

構文

Visual Basic 6.0

```
SystemBytes As Long
```

Visual C++ 6.0

```
long GetSystemBytes()
void SetSystemBytes(long)
```

使用例

Visual Basic 6.0

```
Dim lSystemBytes As Long
lSystemBytes =    .SystemBytes
```

Visual C++ 6.0

```
long lSystemBytes = m_ctrl.GetSystemBytes();
```

特記事項

参照

3.1.41 T1

SECS-I の T1 タイムアウトを取得または設定します。単位はミリ秒。

構文

Visual Basic 6.0

```
T1 As Long
```

Visual C++ 6.0

```
long GetT1()
void SetT1(long)
```

使用例

Visual Basic 6.0

```
Dim lT1 As Long
lT1 = .T1
```

Visual C++ 6.0

```
long lT1 = m_ctrl.GetT1();
```

特記事項

永続化プロパティ。

このプロパティは現時点では未使用です。

参照

3.1.42 T2

SECS-I の T2 タイムアウトを取得または設定します。単位はミリ秒。

構文

Visual Basic 6.0

```
T2 As Long
```

Visual C++ 6.0

```
long GetT2()
void SetT2(long)
```

使用例

Visual Basic 6.0

```
Dim IT2 As Long
IT2 = .T2
```

Visual C++ 6.0

```
long IT2 = m_ctrl.GetT2();
```

特記事項

永続化プロパティ。

このプロパティは現時点では未使用です。

参照

3.1.43 T3

T3 タイムアウトを取得または設定します。単位は秒。

構文

Visual Basic 6.0

```
T3 As Long
```

Visual C++ 6.0

```
long GetT3()
void SetT3(long)
```

使用例

Visual Basic 6.0

```
Dim IT3 As Long
IT3 = .T3
```

Visual C++ 6.0

```
long IT3 = m_ctrl.GetT3();
```

特記事項

永続化プロパティ。

トランザクションタイムアウトとも言います。一次メッセージを送信してから二次メッセージが返信されるまでの時間です。この時間以上経過すると T3 タイムアウトとなり、装置は S9F9 トランザクションタイマタイムアウト(TIN)を送信します。

参照

3.1.44 T4

SECS-I の T4 タイムアウトを取得または設定します。単位は秒。

構文

Visual Basic 6.0

```
T4 As Long
```

Visual C++ 6.0

```
long GetT4()
void SetT4(long)
```

使用例

Visual Basic 6.0

```
Dim lT4 As Long
lT4 = .T4
```

Visual C++ 6.0

```
long lT4 = m_ctrl.GetT4();
```

特記事項

永続化プロパティ。

このプロパティは現時点では未使用です。

参照

3.1.45 T5

HSMS の T5 タイムアウトを取得または設定します。単位は秒。

構文

Visual Basic 6.0

```
T5 As Long
```

Visual C++ 6.0

```
long GetT5()
void SetT5(long)
```

使用例

Visual Basic 6.0

```
Dim lT5 As Long
lT5 = .T5
```

Visual C++ 6.0

```
long lT5 = m_ctrl.GetT5();
```

特記事項

永続化プロパティ。

接続セパレーションタイムアウトとも言います。T5 プロパティはアクティブエンティティの場合のみ関係します。接続に失敗した場合や切断した場合に、再接続するにはこの時間以上は待たなければなりません。

参照

3.1.46 T6

HSMS の T6 タイムアウトを取得または設定します。単位は秒。

構文

Visual Basic 6.0

```
T6 As Long
```

Visual C++ 6.0

```
long GetT6()
void SetT6(long)
```

使用例

Visual Basic 6.0

```
Dim lT6 As Long
lT6 = .T6
```

Visual C++ 6.0

```
long lT6 = m_ctrl.GetT6();
```

特記事項

永続化プロパティ。

コントロールトランザクションタイムアウトとも言います。コントロールメッセージ（S タイプが 0 以外のメッセージ）で、リクエストを送信してから、レスポンスを受信するまでの時間です。T3 タイムアウトはデータメッセージに対してのトランザクションを監視しますが、そのコントロールメッセージ版だと言えます。

具体的には、以下の時間がタイムアウトになったタイミングで発生します。

S タイプ	説明
1	Select.req を送信してから Select.rsp を受信するまでの時間
3	Deselect.req を送信してから Deselect.rsp を受信するまでの時間
5	Linktest.req を送信してから Linktest.rsp を受信するまでの時間

参照

3.1.47 T7

HSMS の T7 タイムアウトを取得または設定します。単位は秒。

構文

Visual Basic 6.0

```
T7 As Long
```

Visual C++ 6.0

```
long GetT7()
void SetT7(long)
```

使用例

Visual Basic 6.0

```
Dim lT7 As Long
lT7 = .T7
```

Visual C++ 6.0

```
long lT7 = m_ctrl.GetT7();
```

特記事項

永続化プロパティ。

NOT SELECTED タイムアウトとも言います。HSMS で TCP/IP レベルでは接続はしたものの、Selected 状態に移行しないで放置しておくと、設定した時間が経過した後に切断されます。また Deselect.req で Selected 状態から Not Selected 状態に移行しても、再び Selected 状態に移行しないで放置しておくと、やはり切断されます。

参照

3.1.48 T8

HSMS の T8 タイムアウトを取得または設定します。単位は秒。

構文

Visual Basic 6.0

```
T8 As Long
```

Visual C++ 6.0

```
long GetT8()
void SetT8(long)
```

使用例

Visual Basic 6.0

```
Dim lT8 As Long
lT8 = .T8
```

Visual C++ 6.0

```
long lT8 = m_ctrl.GetT8();
```

特記事項

永続化プロパティ。

ネットワークキャラクタ間タイムアウトとも言います。HSMS 接続が切れていなくても、1つのメッセージ受信の途中でしばらくの間データが途切れると、メッセージの続きなのかどうかの判断ができません。T8 タイマー以上経過すると、「通信の失敗」とみなされ、結果的に切断されることになります。

SECS-I の T1 タイムアウトに似ています。

参照

3.1.49 Verification

メッセージ構造の検証結果を取得または設定します。以下のいずれかとなります。

値	列挙型	説明
0	VerificationCorrect	問題なし。
1	VerificationUserDefined	ユーザ定義メッセージ。
2	VerificationIncorrect	メッセージ内容に問題あり。
3	VerificationIncorrectAndReply	メッセージ内容に問題があり、返信する必要がある。
4	VerificationNoWBit	ウェイトビットが必要なメッセージなのに、実際にはない。
5	VerificationWBit	ウェイトビットが不要なメッセージなのに、実際にはある。
6	VerificationWrongDirection	メッセージの方向が間違っている。
7	VerificationUnrecognizedStream	認識できないストリーム。
8	VerificationUnrecognizedFunction	認識できないファンクション。

構文

Visual Basic 6.0

```
Verification As Integer
```

Visual C++ 6.0

```
short GetVerification()
void SetVerification(short)
```

使用例

Visual Basic 6.0

```
```

Visual C++ 6.0

```
```

特記事項

メッセージを受信すると SavoyGem コントロール内部でメッセージ構造が検証されます。その結果が Verification にセットされ、Received イベントが発生します。

参照

3.1.50 VIDCount

登録されている VID の個数を取得します。0 の場合は一つも登録されていないことを意味します。

構文

Visual Basic 6.0

```
VIDCount As Long
```

Visual C++ 6.0

```
long GetVIDCount()
```

使用例

Visual Basic 6.0

```
Dim lVIDCount As Long
lVIDCount = .VIDCount
```

Visual C++ 6.0

```
long lVIDCount = m_ctrl.GetVIDCount();
```

特記事項

読み出し専用プロパティ。

VIDCount プロパティは登録されている VID の数を返すので、インデックスとして使用可能な値は 0 から(VIDCount - 1)までとなります。これを ToVID メソッドを使って VID に変換します。

参照

3.1.51 ViewStyle

SavoyGem コントロールの表示形式を取得または設定します。以下のうちいずれかとなります。

スタイル	値	説明
RedrawNone	0	何も表示しない
RedrawHsms	1	HSMS の物理接続状態だけを表示
RedrawGem	2	GEM の論理接続状態だけを表示
RedrawNormal	3	全て表示

構文

Visual Basic 6.0

```
ViewStyle As Integer
```

Visual C++ 6.0

```
short GetViewStyle()
void SetViewStyle(short)
```

使用例

Visual Basic 6.0

```
.ViewStyle = 3
```

Visual C++ 6.0

```
m_ctrl.SetViewStyle(3);
```

特記事項

永続化プロパティ。

参照

3.1.52 Wbit

SECS-II ヘッダのウェイトビットを取得または設定します。

値	説明
False	返信なし
True	返信あり

HSMS データメッセージでは下記のヘッダ構造が使用されます。

Byte	説明
1	セッション ID
2	
3	W | ストリーム番号
4	ファンクション番号
5	P タイプ
6	S タイプ
7	システムバイト
8	
9	
10	

構文

Visual Basic 6.0

```
Wbit As Boolean
```

Visual C++ 6.0

```
BOOL GetWbit()
void SetWbit(BOOL)
```

使用例

Visual Basic 6.0

```
If .Wbit Then
    ‘ Reply
```

Visual C++ 6.0

```
If(m_ctrl.GetWbit())
{
    // Reply
```

特記事項

参照

3.1.53 WorkSpace

SECS-II メッセージのワークスペースを取得または設定します。ワークスペースとは操作対象となるメッセージの作業領域のことです。SavoyGem コントロールには 3 つのワークスペースが用意されています。それぞれに「一次メッセージ用」の Reply=True と「二次メッセージ用」の Reply=False がありますので、全部で 6 つのメッセージを扱うことができます。基本的にはワークスペース 0 だけで処理できるようになっています。

WorkSpace	Reply	説明
0	False	送受信メッセージ
0	True	受信メッセージの返信
1	False	送信済みメッセージ
1	True	自由に使用可能
2	False	自由に使用可能
2	True	自由に使用可能

メッセージを受信して Received イベントで通知される際には、WorkSpace0 の Reply=False に受信したメッセージが格納されます。受信したメッセージが二次メッセージだとしても、Reply=False に格納されているので注意が必要です。イベントが通知された瞬間は WorkSpace0 の Reply=False が選択されていますが、イベントハンドラから SavoyGem コントロールに処理が戻ると、以前に選択されていた WorkSpace と Reply に戻ります。

一次メッセージの場合は、この時点で「推奨される返信メッセージ」も Reply=True に格納されています。このまま返信する場合は、Reply を True に切り替えて Send を呼ぶだけですが、内容を編集することもできます。

一次メッセージを送信する場合は WorkSpace0 の Reply=False を使用します。送信が完了すると Sent イベントで通知されますが、その際には自動的に WorkSpace1 の Reply=False が選択されています。

構文

Visual Basic 6.0

```
WorkSpace As Integer
```

Visual C++ 6.0

```
short GetWorkSpace()
void SetWorkSpace(short)
```

使用例

Visual Basic 6.0

```
.WorkSpace = 0
.Reply = False
.SML = “s1f13w{<a’Savoy’><a’1.00’>}”
Dim strMsg As String
strMsg = .Msg
```

Visual C++ 6.0

```
m_ctrl.SetWorkSpace(0);
m_ctrl.SetReply(false);
m_ctrl.SetSml(“s1f13w{<a’Savoy’><a’1.00’>}”);
CString strMsg = m_ctrl.GetMsg();
```

特記事項

参照

3.2 配列型プロパティ

3.2.1 ALCD

ALID のアラームコードを取得または設定します。

ALID	説明
0	未使用
1	人間の安全に関わるもの
2	装置の安全に関わるもの
3	パラメータコントロールアラーム
4	パラメータコントロールエラー
5	回復不能エラー
6	装置状態の警告
7	注意フラグ
8	データの保証不可
>8	その他のカテゴリ
9-63	保留

構文

Visual Basic 6.0

```
Property ALCD(lALID As Long) As Integer
```

Visual C++ 6.0

```
short GetAlcd(long lALID)
void SetAlcd(long lALID, short nNewValue)
```

引数	説明
lALID	アラーム ID

使用例

ALID が登録されていない場合は値をセットすることはできません。登録されている ALID の一覧を得るには以下の方法を使います。

まず ALIDCount プロパティの値がいくつかを調べます。

Visual Basic 6.0

```
Dim lCount As Long
lCount = .ALIDCount
```

Visual C++ 6.0

```
long lCount = m_ctrl.GetALIDCount();
```

ALIDCount プロパティは登録されている ALID の数を返すので、インデックスとして使用可能な値は 0 から(ALIDCount - 1)までとなります。これを ToALID メソッドを使って ALID に変換します。

Visual Basic 6.0

```
Dim lALID As Long
lALID = .ToALID(0)
```

Visual C++ 6.0

```
long lALID = m_ctrl.ToALID(0);
```

ALID が得られたので ALCD、ALTX、ALIDEnalbe、ALIDSet にアクセスすることができます。

Visual Basic 6.0

```
Dim nALCD As Integer
nALCD = .ALCD(lALID)

Dim strALTX As String
strALTX = ALTX(lALID)

Dim bALIDEnable As Boolean
bALIDEnable = ALIDEnable(lALID)

Dim bALIDSet As Boolean
bALIDSet = ALIDSet(lALID)
```

Visual C++ 6.0

```
int nALCD = m_ctrl.GetAlcd(lALID);
CString strALTX = m_ctrl.GetAltx(lALID);
bool bALIDEnable = !!m_ctrl.GetALIDEnable(lALID);
bool bALIDSet = !!m_ctrl.GetALIDSet(lALID);
```

これを For 文で繰り返して列挙します。全ソースコードを示します。

Visual Basic 6.0

```
Dim lCount As Long
lCount = .ALIDCount

Dim lCnt As Long
For lCnt = 0 to lCount – 1
    Dim lALID As Long
    lALID = .ToALID(lCnt)

    Dim nALCD As Integer
    nALCD = .ALCD(lALID)

    Dim strALTX As String
    strALTX = .ALTX(lALID)

    Dim bALIDEnable As Boolean
    bALIDEnable = ALIDEnable(lALID)

    Dim bALIDSet As Boolean
    bALIDSet = ALIDSet(lALID)
Next lCnt
```

Visual C++ 6.0

```
long lCount = m_ctrl.GetALIDCount();
for(long lCnt=0;lCnt<lCount;lCnt++)
{
    long lALID = m_ctrl.ToALID(0);
    int nALCD = m_ctrl.GetAlcd(lALID);
    CString strALTX = m_ctrl.GetAltx(lALID);
    bool bALIDEnable = !!m_ctrl.GetALIDEnable(lALID);
    bool bALIDSet = !!m_ctrl.GetALIDSet(lALID);
```

```
}
```

特記事項

ALCD プロパティに値をセットしても、下位 7 ビットしか記録されません。ALCD の最上位ビットは、アラームの発生・解除の意味で使われますが、このビットは InvokeAlarm の引数で指定します。

参照

3.2.2 ALIDEnable

ALID の有効・無効を設定します。

構文

Visual Basic 6.0

```
ALIDEnable(lALID As Long) As Boolean
```

Visual C++ 6.0

```
BOOL GetALIDEnable(long lALID)
void SetALIDEnable(long lALID, BOOL bNewValue)
```

引数	説明
lALID	アラーム ID

使用例

ALID が登録されていない場合は値をセットすることはできません。登録されている ALID の一覧を得るには以下の方法を使います。

まず ALIDCount プロパティの値がいくつかを調べます。

Visual Basic 6.0

```
Dim lCount As Long
lCount = .ALIDCount
```

Visual C++ 6.0

```
long lCount = m_ctrl.GetALIDCount();
```

ALIDCount プロパティは登録されている ALID の数を返すので、インデックスとして使用可能な値は 0 から(ALIDCount - 1)までとなります。これを ToALID メソッドを使って ALID に変換します。

Visual Basic 6.0

```
Dim lALID As Long
lALID = .ToALID(0)
```

Visual C++ 6.0

```
long lALID = m_ctrl.ToALID(0);
```

ALID が得られたので ALCD、ALTX、ALIDEnalbe、ALIDSet にアクセスすることができます。

Visual Basic 6.0

```
Dim nALCD As Integer
nALCD = .ALCD(lALID)

Dim strALTX As String
strALTX = ALTX(lALID)
```

```
Dim bALIDEnable As Boolean
bALIDEnable = ALIDEnable(lALID)

Dim bALIDSet As Boolean
bALIDSet = ALIDSet(lALID)
```

Visual C++ 6.0

```
int nALCD = m_ctrl.GetAlcd(lALID);
CString strALTX = m_ctrl.GetAltx(lALID);
bool bALIDEnable = !!m_ctrl.GetALIDEnable(lALID);
bool bALIDSet = !!m_ctrl.GetALIDSet(lALID);
```

これを For 文で繰り返して列挙します。全ソースコードを示します。

Visual Basic 6.0

```
Dim lCount As Long
lCount = .ALIDCount

Dim lCnt As Long
For lCnt = 0 to lCount – 1
    Dim lALID As Long
    lALID = .ToALID(lCnt)

    Dim nALCD As Integer
    nALCD = .ALCD(lALID)

    Dim strALTX As String
    strALTX = .ALTX(lALID)

    Dim bALIDEnable As Boolean
    bALIDEnable = ALIDEnable(lALID)

    Dim bALIDSet As Boolean
    bALIDSet = ALIDSet(lALID)

Next lCnt
```

Visual C++ 6.0

```
long lCount = m_ctrl.GetALIDCount();
for(long lCnt=0;lCnt<lCount;lCnt++)
{
    long lALID = m_ctrl.ToALID(0);
    int nALCD = m_ctrl.GetAlcd(lALID);
    CString strALTX = m_ctrl.GetAltx(lALID);
    bool bALIDEnable = !!m_ctrl.GetALIDEnable(lALID);
    bool bALIDSet = !!m_ctrl.GetALIDSet(lALID);
}
```

特記事項

参照

3.2.3 ALIDSet

ALID のアラーム発生・解除を設定します。

構文

Visual Basic 6.0

```
ALIDSet(lALID As Long) As Boolean
```

Visual C++ 6.0

```
BOOL GetALIDSet(long lALID)
void SetALIDSet(long lALID, BOOL bNewValue)
```

引数	説明
lALID	アラーム ID

使用例

ALID が登録されていない場合は値をセットすることはできません。登録されている ALID の一覧を得るには以下の方法を使います。

まず ALIDCount プロパティの値がいくつかを調べます。

Visual Basic 6.0

```
Dim lCount As Long
lCount = .ALIDCount
```

Visual C++ 6.0

```
long lCount = m_ctrl.GetALIDCount();
```

ALIDCount プロパティは登録されている ALID の数を返すので、インデックスとして使用可能な値は 0 から(ALIDCount - 1)までとなります。これを ToALID メソッドを使って ALID に変換します。

Visual Basic 6.0

```
Dim lALID As Long
lALID = .ToALID(0)
```

Visual C++ 6.0

```
long lALID = m_ctrl.ToALID(0);
```

ALID が得られたので ALCD、ALTX、ALIDEnalbe、ALIDSet にアクセスすることができます。

Visual Basic 6.0

```
Dim nALCD As Integer
nALCD = .ALCD(lALID)

Dim strALTX As String
strALTX = ALTX(lALID)
```

```
Dim bALIDEnable As Boolean
bALIDEnable = ALIDEnable(lALID)

Dim bALIDSet As Boolean
bALIDSet = ALIDSet(lALID)
```

Visual C++ 6.0

```
int nALCD = m_ctrl.GetAlcd(lALID);
CString strALTX = m_ctrl.GetAltx(lALID);
bool bALIDEnable = !!m_ctrl.GetALIDEnable(lALID);
bool bALIDSet = !!m_ctrl.GetALIDSet(lALID);
```

これを For 文で繰り返して列挙します。全ソースコードを示します。

Visual Basic 6.0

```
Dim lCount As Long
lCount = .ALIDCount

Dim lCnt As Long
For lCnt = 0 to lCount – 1
    Dim lALID As Long
    lALID = .ToALID(lCnt)

    Dim nALCD As Integer
    nALCD = .ALCD(lALID)

    Dim strALTX As String
    strALTX = .ALTX(lALID)

    Dim bALIDEnable As Boolean
    bALIDEnable = ALIDEnable(lALID)

    Dim bALIDSet As Boolean
    bALIDSet = ALIDSet(lALID)

Next lCnt
```

Visual C++ 6.0

```
long lCount = m_ctrl.GetALIDCount();
for(long lCnt=0;lCnt<lCount;lCnt++)
{
    long lALID = m_ctrl.ToALID(0);
    int nALCD = m_ctrl.GetAlcd(lALID);
    CString strALTX = m_ctrl.GetAltx(lALID);
    bool bALIDEnable = !!m_ctrl.GetALIDEnable(lALID);
    bool bALIDSet = !!m_ctrl.GetALIDSet(lALID);
}
```

特記事項

参照

3.2.4 ALTX

ALID のアラームテキストを取得または設定します。

構文

Visual Basic 6.0

```
Property ALTX(lALID As Long) As String
```

Visual C++ 6.0

```
CString GetAltx(long lALID)
void SetAltx(long lALID, LPCTSTR lpszNewValue)
```

引数	説明
lALID	アラーム ID

使用例

ALID が登録されていない場合は値をセットすることはできません。登録されている ALID の一覧を得るには以下の方法を使います。

まず ALIDCount プロパティの値がいくつかを調べます。

Visual Basic 6.0

```
Dim lCount As Long
lCount = .ALIDCount
```

Visual C++ 6.0

```
long lCount = m_ctrl.GetALIDCount();
```

ALIDCount プロパティは登録されている ALID の数を返すので、インデックスとして使用可能な値は 0 から(ALIDCount - 1)までとなります。これを ToALID メソッドを使って ALID に変換します。

Visual Basic 6.0

```
Dim lALID As Long
lALID = .ToALID(0)
```

Visual C++ 6.0

```
long lALID = m_ctrl.ToALID(0);
```

ALID が得られたので ALCD、ALTX、ALIDEnalbe、ALIDSet にアクセスすることができます。

Visual Basic 6.0

```
Dim nALCD As Integer
nALCD = .ALCD(lALID)

Dim strALTX As String
strALTX = ALTX(lALID)
```

```
Dim bALIDEnable As Boolean
bALIDEnable = ALIDEnable(lALID)

Dim bALIDSet As Boolean
bALIDSet = ALIDSet(lALID)
```

Visual C++ 6.0

```
int nALCD = m_ctrl.GetAlcd(lALID);
CString strALTX = m_ctrl.GetAltx(lALID);
bool bALIDEnable = !!m_ctrl.GetALIDEnable(lALID);
bool bALIDSet = !!m_ctrl.GetALIDSet(lALID);
```

これを For 文で繰り返して列挙します。全ソースコードを示します。

Visual Basic 6.0

```
Dim lCount As Long
lCount = .ALIDCount

Dim lCnt As Long
For lCnt = 0 to lCount – 1
    Dim lALID As Long
    lALID = .ToALID(lCnt)

    Dim nALCD As Integer
    nALCD = .ALCD(lALID)

    Dim strALTX As String
    strALTX = .ALTX(lALID)

    Dim bALIDEnable As Boolean
    bALIDEnable = ALIDEnable(lALID)

    Dim bALIDSet As Boolean
    bALIDSet = ALIDSet(lALID)

Next lCnt
```

Visual C++ 6.0

```
long lCount = m_ctrl.GetALIDCount();
for(long lCnt=0;lCnt<lCount;lCnt++)
{
    long lALID = m_ctrl.ToALID(0);
    int nALCD = m_ctrl.GetAlcd(lALID);
    CString strALTX = m_ctrl.GetAltx(lALID);
    bool bALIDEnable = !!m_ctrl.GetALIDEnable(lALID);
    bool bALIDSet = !!m_ctrl.GetALIDSet(lALID);
}
```

特記事項

ALTX は SEMI の規格では最大 40 文字に制限されていますが、SavoyGem コントロールではこの制限はありませんので、自由に好きな長さのアラームテキストを送ることができます。

参照

3.2.5 CEIDDescription

CEID の説明を取得または設定します。

構文

Visual Basic 6.0

```
Property CEIDDescription(lCEID As Long) As String
```

Visual C++ 6.0

```
CString GetCEIDDescription(long lCEID)
void SetCEIDDescription(long lCEID, LPCTSTR lpszNewValue)
```

引数	説明
lCEID	イベント ID

使用例

Visual Basic 6.0

```
.CEIDDescription(3000) = “Ready to load”
```

Visual C++ 6.0

```
m_ctrl.SetCEIDDescription(3000,“Ready to load”);
```

特記事項

参照

3.2.6 CEIDEnable

CEID の有効・無効を取得または設定します。

構文

Visual Basic 6.0

```
Property CEIDEnable(lCEID As Long) As String
```

Visual C++ 6.0

```
CString GetCEIDEnable(long lCEID)
void SetCEIDEnable(long lCEID, LPCTSTR lpszNewValue)
```

引数	説明
lCEID	イベント ID

使用例

Visual Basic 6.0

```
.CEIDEnable(3000) = True
```

Visual C++ 6.0

```
m_ctrl.SetCEIDEnable(3000,true);
```

特記事項

参照

3.2.7 VIDDefault

VID のデフォルト値を取得または設定します。

構文

Visual Basic 6.0

```
Property VIDDefault(lVID As Long) As String
```

Visual C++ 6.0

```
CString GetVIDDefault(long lVID)
void SetVIDDefault(long lVID, LPCTSTR lpszNewValue)
```

引数	説明
lVID	変数 ID

使用例

Visual Basic 6.0

```
Dim lVID As Long
lVID = 1100

.VIDType(lVID) = 2
.VIDValue(lVID) = “7”
.VIDNodeType(lVID) = 15
.VIDDescription(lVID) = “Laser level”
.VIDMin(lVID) = “0”
.VIDMax(lVID) = “10”
.VIDDefault(lVID) = “5”
.VIDUnit(lVID) = “mW”

Dim strVID As String
strVID = .VIDRawValue(lVID)
```

Visual C++ 6.0

```
long lVID = 1100

m_ctrl.SetVIDType(lVID, 2);
m_ctrl.SetVIDValue(lVID, ”7”);
m_ctrl.SetVIDNodeType(lVID, 15);
m_ctrl.SetVIDDescription(lVID, “Laser level”);
m_ctrl.SetVIDMin(lVID, “0”);
m_ctrl.SetVIDMax(lVID, “10”);
m_ctrl.SetVIDDefault(lVID, “5”);
m_ctrl.SetVIDUnit(lVID, “mW”);

CString strVID = m_ctrl.GetVIDRawValue(lVID);
```

特記事項

S2F30 装置定数名リスト(ECN)の ECDEF です。

参照

3.2.8 VIDDescription

VID の説明を取得または設定します。

構文

Visual Basic 6.0

```
Property VID 説明(lVID As Long) As String
```

Visual C++ 6.0

```
CString GetVID 説明(long lVID)
void SetVID 説明(long lVID, LPCTSTR lpszNewValue)
```

引数	説明
lVID	変数 ID

使用例

Visual Basic 6.0

```
Dim lVID As Long
lVID = 1100

.VIDType(lVID) = 2
.VIDValue(lVID) = “7”
.VIDNodeType(lVID) = 15
.VIDDescription(lVID) = “Laser level”
.VIDMin(lVID) = “0”
.VIDMax(lVID) = “10”
.VIDDefault(lVID) = “5”
.VIDUnit(lVID) = “mW”

Dim strVID As String
strVID = .VIDRawValue(lVID)
```

Visual C++ 6.0

```
long lVID = 1100

m_ctrl.SetVIDType(lVID, 2);
m_ctrl.SetVIDValue(lVID, ”7”);
m_ctrl.SetVIDNodeType(lVID, 15);
m_ctrl.SetVIDDescription(lVID, “Laser level”);
m_ctrl.SetVIDMin(lVID, “0”);
m_ctrl.SetVIDMax(lVID, “10”);
m_ctrl.SetVIDDefault(lVID, “5”);
m_ctrl.SetVIDUnit(lVID, “mW”);

CString strVID = m_ctrl.GetVIDRawValue(lVID);
```

特記事項

このプロパティは VIDType によって扱いが若干異なります。SVID の場合は S1F12 状態変数名リスト応答(SVNRR)の SVNAME として扱われます。

ECID の場合は S2F30 装置定数名リスト(ECN)の ECNAME として扱われます。

DVID の場合には DVNAME として扱いたいところですが、あいにく GEM では S6F4 も S6F8 も定義されていないので、SavoyGem コントロールでも自動処理されることはありません。しかしユーザが VIDDescription プロパティを使用して S6F4 や S6F8 に対応させることは可能です。

参照

3.2.9 VIDMax

VID の最大値を取得または設定します。

構文

Visual Basic 6.0

```
Property VIDMax(lVID As Long) As String
```

Visual C++ 6.0

```
CString GetVIDMax(long lVID)
void SetVIDMax(long lVID, LPCTSTR lpszNewValue)
```

引数	説明
lVID	変数 ID

使用例

Visual Basic 6.0

```
Dim lVID As Long
lVID = 1100

.VIDType(lVID) = 2
.VIDValue(lVID) = “7”
.VIDNodeType(lVID) = 15
.VIDDescription(lVID) = “Laser level”
.VIDMin(lVID) = “0”
.VIDMax(lVID) = “10”
.VIDDefault(lVID) = “5”
.VIDUnit(lVID) = “mW”

Dim strVID As String
strVID = .VIDRawValue(lVID)
```

Visual C++ 6.0

```
long lVID = 1100

m_ctrl.SetVIDType(lVID, 2);
m_ctrl.SetVIDValue(lVID, ”7”);
m_ctrl.SetVIDNodeType(lVID, 15);
m_ctrl.SetVIDDescription(lVID, “Laser level”);
m_ctrl.SetVIDMin(lVID, “0”);
m_ctrl.SetVIDMax(lVID, “10”);
m_ctrl.SetVIDDefault(lVID, “5”);
m_ctrl.SetVIDUnit(lVID, “mW”);

CString strVID = m_ctrl.GetVIDRawValue(lVID);
```

特記事項

S2F30 装置定数名リスト(ECN)の ECMAX です。

参照

3.2.10 VIDMin

VID の最小値を取得または設定します。

構文

Visual Basic 6.0

```
Property VIDMin(lVID As Long) As String
```

Visual C++ 6.0

```
CString GetVIDMin(long lVID)
void SetVIDMin(long lVID, LPCTSTR lpszNewValue)
```

引数	説明
lVID	変数 ID

使用例

Visual Basic 6.0

```
Dim lVID As Long
lVID = 1100

.VIDType(lVID) = 2
.VIDValue(lVID) = “7”
.VIDNodeType(lVID) = 15
.VIDDescription(lVID) = “Laser level”
.VIDMin(lVID) = “0”
.VIDMax(lVID) = “10”
.VIDDefault(lVID) = “5”
.VIDUnit(lVID) = “mW”

Dim strVID As String
strVID = .VIDRawValue(lVID)
```

Visual C++ 6.0

```
long lVID = 1100

m_ctrl.SetVIDType(lVID, 2);
m_ctrl.SetVIDValue(lVID, ”7”);
m_ctrl.SetVIDNodeType(lVID, 15);
m_ctrl.SetVIDDescription(lVID, “Laser level”);
m_ctrl.SetVIDMin(lVID, “0”);
m_ctrl.SetVIDMax(lVID, “10”);
m_ctrl.SetVIDDefault(lVID, “5”);
m_ctrl.SetVIDUnit(lVID, “mW”);

CString strVID = m_ctrl.GetVIDRawValue(lVID);
```

特記事項

S2F30 装置定数名リスト(ECN)の ECMIN です。

参照

3.2.11 VIDNodeType

VID のノードタイプを取得または設定します。

値	列挙型	説明
1	SecsTypeList	リスト
2	SecsTypeBinary	バイナリ
3	SecsTypeBoolean	ブーリアン
4	SecsTypeAscii	ASCII 文字列
5	SecsTypeJis	JIS 8 文字列
6	SecsTypeLong8	8 バイト符号付き整数
7	SecsTypeChar	1 バイト符号付き整数
8	SecsTypeShort	2 バイト符号付き整数
9	SecsTypeLong	4 バイト符号付き整数
10	SecsTypeDouble	8 バイト浮動小数点数
11	SecsTypeFloat	4 バイト浮動小数点数
12	SecsTypeDWord8	8 バイト符号なし整数
13	SecsTypeByte	1 バイト符号なし整数
14	SecsTypeWord	2 バイト符号なし整数
15	SecsTypeDWord	4 バイト符号なし整数
16	SecsTypeAscii2	2 バイト ASCII 文字列

構文

Visual Basic 6.0

```
Property VIDNodeType(lVID As Long) As Integer
```

Visual C++ 6.0

```
short GetVIDNodeType(long lVID)
void SetVIDNodeType(long lVID, short nNewValue)
```

引数	説明
lVID	変数 ID

使用例

Visual Basic 6.0

```
Dim lVID As Long
lVID = 1100

.VIDType(lVID) = 2
.VIDValue(lVID) = “7”
.VIDNodeType(lVID) = 15
.VIDDescription(lVID) = “Laser level”
.VIDMin(lVID) = “0”
.VIDMax(lVID) = “10”
.VIDDefault(lVID) = “5”
.VIDUnit(lVID) = “mW”

Dim strVID As String
strVID = .VIDRawValue(lVID)
```

Visual C++ 6.0

```
long lVID = 1100
```

```
m_ctrl.SetVIDType(lVID, 2);
m_ctrl.SetVIDValue(lVID, ”7”);
m_ctrl.SetVIDNodeType(lVID, 15);
m_ctrl.SetVIDDescription(lVID, “Laser level”);
m_ctrl.SetVIDMin(lVID, “0”);
m_ctrl.SetVIDMax(lVID, “10”);
m_ctrl.SetVIDDefault(lVID, “5”);
m_ctrl.SetVIDUnit(lVID, “mW”);

CString strVID = m_ctrl.GetVIDRawValue(lVID);
```

特記事項

VIDNodeType プロパティを読み出したときに値が 0 の場合は、「無効な型」であることを意味します。

参照

3.2.12 VIDRawValue

VID の現在の設定値を SML 表現で取得または設定します。

構文

Visual Basic 6.0

```
Property VIDRawValue(lVID As Long) As String
```

Visual C++ 6.0

```
CString GetVIDRawValue(long lVID)
void SetVIDRawValue(long lVID, LPCTSTR lpszNewValue)
```

引数	説明
lVID	変数 ID

使用例

Visual Basic 6.0

```
Dim lVID As Long
lVID = 1100

.VIDType(lVID) = 2
.VIDValue(lVID) = “7”
.VIDNodeType(lVID) = 15
.VIDDescription(lVID) = “Laser level”
.VIDMin(lVID) = “0”
.VIDMax(lVID) = “10”
.VIDDefault(lVID) = “5”
.VIDUnit(lVID) = “mW”

Dim strVID As String
strVID = .VIDRawValue(lVID)
```

Visual C++ 6.0

```
long lVID = 1100

m_ctrl.SetVIDType(lVID, 2);
m_ctrl.SetVIDValue(lVID, ”7”);
m_ctrl.SetVIDNodeType(lVID, 15);
m_ctrl.SetVIDDescription(lVID, “Laser level”);
m_ctrl.SetVIDMin(lVID, “0”);
m_ctrl.SetVIDMax(lVID, “10”);
m_ctrl.SetVIDDefault(lVID, “5”);
m_ctrl.SetVIDUnit(lVID, “mW”);

CString strVID = m_ctrl.GetVIDRawValue(lVID);
```

特記事項

ここに SML 文字列をセットすると、S6F11 イベントレポート送信(ERS)でイベントを送信する際に、レポートに従ってメッセージが組み立てられます。このとき個々の VID の値はこの VIDRawValue プロパティの文字列が使われます。SML にはストリーム、ファンクションは記述できません。

SavoyGem コントロールでは文法エラーや VID の型をチェックしないので、正しい SML をセットする必要があります。もし VIDNodeType プロパティの型に従って値をセットしたい場合は、VIDValue プロパティを使用してください。

参照

3.2.13 VIDType

VID の変数型を取得または設定します。以下のうちいずれかとなります。

タイプ	説明
1	ECID
2	SVID
4	DVID

これ以外のタイプを指定することはできません。

構文

Visual Basic 6.0

```
Property VIDType(lVID As Long) As Integer
```

Visual C++ 6.0

```
short GetVIDType(long lVID)
void SetVIDType(long lVID, short nNewValue)
```

引数	説明
lVID	変数 ID

使用例

Visual Basic 6.0

```
Dim lVID As Long
lVID = 1100

.VIDType(lVID) = 2
.VIDValue(lVID) = “7”
.VIDNodeType(lVID) = 15
.VIDDescription(lVID) = “Laser level”
.VIDMin(lVID) = “0”
.VIDMax(lVID) = “10”
.VIDDefault(lVID) = “5”
.VIDUnit(lVID) = “mW”

Dim strVID As String
strVID = .VIDRawValue(lVID)
```

Visual C++ 6.0

```
long lVID = 1100

m_ctrl.SetVIDType(lVID, 2);
m_ctrl.SetVIDValue(lVID, ”7”);
m_ctrl.SetVIDNodeType(lVID, 15);
m_ctrl.SetVIDDescription(lVID, “Laser level”);
m_ctrl.SetVIDMin(lVID, “0”);
m_ctrl.SetVIDMax(lVID, “10”);
m_ctrl.SetVIDDefault(lVID, “5”);
m_ctrl.SetVIDUnit(lVID, “mW”);

CString strVID = m_ctrl.GetVIDRawValue(lVID);
```

特記事項

参照

3.2.14 VIDUnit

VID の単位を取得または設定します。

構文

Visual Basic 6.0

```
Property VIDUnit(lVID As Long) As String
```

Visual C++ 6.0

```
CString GetVIDUnit(long lVID)
void SetVIDUnit(long lVID, LPCTSTR lpszNewValue)
```

引数	説明
lVID	変数 ID

使用例

Visual Basic 6.0

```
Dim lVID As Long
lVID = 1100

.VIDType(lVID) = 2
.VIDValue(lVID) = “7”
.VIDNodeType(lVID) = 15
.VIDDescription(lVID) = “Laser level”
.VIDMin(lVID) = “0”
.VIDMax(lVID) = “10”
.VIDDefault(lVID) = “5”
.VIDUnit(lVID) = “mW”

Dim strVID As String
strVID = .VIDRawValue(lVID)
```

Visual C++ 6.0

```
long lVID = 1100

m_ctrl.SetVIDType(lVID, 2);
m_ctrl.SetVIDValue(lVID, ”7”);
m_ctrl.SetVIDNodeType(lVID, 15);
m_ctrl.SetVIDDescription(lVID, “Laser level”);
m_ctrl.SetVIDMin(lVID, “0”);
m_ctrl.SetVIDMax(lVID, “10”);
m_ctrl.SetVIDDefault(lVID, “5”);
m_ctrl.SetVIDUnit(lVID, “mW”);

CString strVID = m_ctrl.GetVIDRawValue(lVID);
```

特記事項

S2F30 装置定数名リスト(ECN)の UNITS です。

参照

3.2.15 VIDValue

VID の現在の値を型に合わせた表現で取得または設定します。

構文

Visual Basic 6.0

```
Property VIDValue(lVID As Long) As String
```

Visual C++ 6.0

```
CString GetVIDValue(long lVID)
void SetVIDValue(long lVID, LPCTSTR lpszNewValue)
```

引数	説明
lVID	変数 ID

使用例

Visual Basic 6.0

```
Dim lVID As Long
lVID = 1100

.VIDType(lVID) = 2
.VIDValue(lVID) = “7”
.VIDNodeType(lVID) = 15
.VIDDescription(lVID) = “Laser level”
.VIDMin(lVID) = “0”
.VIDMax(lVID) = “10”
.VIDDefault(lVID) = “5”
.VIDUnit(lVID) = “mW”

Dim strVID As String
strVID = .VIDRawValue(lVID)
```

Visual C++ 6.0

```
long lVID = 1100

m_ctrl.SetVIDType(lVID, 2);
m_ctrl.SetVIDValue(lVID, ”7”);
m_ctrl.SetVIDNodeType(lVID, 15);
m_ctrl.SetVIDDescription(lVID, “Laser level”);
m_ctrl.SetVIDMin(lVID, “0”);
m_ctrl.SetVIDMax(lVID, “10”);
m_ctrl.SetVIDDefault(lVID, “5”);
m_ctrl.SetVIDUnit(lVID, “mW”);

CString strVID = m_ctrl.GetVIDRawValue(lVID);
```

特記事項

VIDNodeType プロパティで指定された型に従って、VIDRawValue プロパティに SML 文字列が生成されます。

参照

3.3 メソッド

3.3.1 AboutBox

バージョン情報を表示します。

構文

Visual Basic 6.0

```
Sub AboutBox()
```

Visual C++ 6.0

```
void AboutBox()
```

戻り値

ありません。

使用例

Visual Basic 6.0

```
.AboutBox
```

Visual C++ 6.0

```
m_hsms.AboutBox();
```

特記事項

参照

3.3.2 DefProc

メッセージを受信したときのデフォルトの処理を呼び出します。

構文

Visual Basic 6.0

```
Function DefProc() As Boolean
```

Visual C++ 6.0

```
BOOL DefProc()
```

戻り値

処理を行った場合は True が、行わなかった場合は False が返ります。

使用例

Visual Basic 6.0

```
.DefProc
```

Visual C++ 6.0

```
m_ctrl.DefProc();
```

特記事項

参照

3.3.3 InvokeAlarm

SavoyGem コントロールにアラームイベントを発生させます。具体的にはホストに S5F1 アラーム報告送信(ARS)を送信します。もし指定された ALID が登録されていなかったり、無効に設定されている場合には、アラームイベントは送信されません。

構文

Visual Basic 6.0

```
Function InvokeAlarm(lALID As Long, sALCD As Integer) As Boolean
```

Visual C++ 6.0

```
BOOL InvokeAlarm(long lALID, short sALCD)
```

引数	説明
lALID	アラーム ID。ALID は事前に登録されている必要があります。
sALCD	アラームコード。実際に使用されるのは 8 ビット目だけです。

戻り値

アラームが送信された場合には True が、送信されなかった場合には False が返ります。

使用例

Visual Basic 6.0

```
.InvokeAlarm 325, &H80
```

Visual C++ 6.0

```
m_ctrl.InvokeAlarm(325, 0x80);
```

特記事項

sALCD にどのような値を指定しても下位 7 ビットは無視され、登録済みの ALCD プロパティの値に置き換えられます。ALCD はバイナリ型なので 8 ビットしかありません。このため sALCD で実際に使われるのは 8 ビット目だけということになります。このビットが 1 なら「アラームの発生」を、0 なら「アラームの解除」となります。

「アラームの解除」を送信するには、事前に「アラームの発生」が行われている必要があります。「アラームの発生」を送信すると SavoyGem コントロール内部でその ALID に対して「未解除フラグ」がセットされます。この「未解除フラグ」がセットされていない場合は、「アラームの解除」を送信することはできません。「アラームの解除」を送信すると「未解除フラグ」はリセットされます。

SEMI の仕様で、同一の ALID に対して「未解除フラグ」はオンとオフという情報だけが記録され、発生回数は記録されません。このため「アラームの発生」を連続して 2 回送信しても、「アラームの解除」を 1 回送信しただけで「未解除フラグ」はリセットされます。

「未解除フラグ」の設定情報はファイルに記録されますので、アプリケーションを終了させても復元させることができます。

参照

3.3.4 InvokeEvent

SavoyGemコントロールにイベント発生させます。具体的にはホストにS6F11 イベントレポート送信(ERS)を送信します。もし指定された CEID が登録されていないか、無効に設定されている場合には、イベントは送信されません。

構文

Visual Basic 6.0

```
Function InvokeEvent(lCEID As Long) As Boolean
```

Visual C++ 6.0

```
BOOL InvokeEvent(long lCEID)
```

引数	説明
lCEID	CEID です。

戻り値

イベントが送信された場合は True が、送信されなかった場合は False が返ります。

使用例

Visual Basic 6.0

```
.InvokeEvent 1100
```

Visual C++ 6.0

```
m_ctrl.InvokeEvent(1100);
```

特記事項

イベントにレポートがリンクされている場合は、レポートも自動的に生成されます。

参照

3.3.5 IsValidVID

指定された VID が有効かどうかを検査します。

構文

Visual Basic 6.0

```
Function IsValidVID(lVID As Long) As Boolean
```

Visual C++ 6.0

```
BOOL IsValidVID(long lVID)
```

引数	説明
lVID	変数 ID

戻り値

VID が登録されている場合は True が、登録されていない場合は False が返ります。

使用例

Visual Basic 6.0

```
If .IsValidVID(3201) Then
    ‘ OK
```

Visual C++ 6.0

```
if(m_ctrl.IsValidVID(3201)
{
    // OK
```

特記事項

参照

3.3.6 LoadData

SavoyGem データファイルを読み込みます。

構文

Visual Basic 6.0

```
Function LoadData() As Boolean
```

Visual C++ 6.0

```
BOOL LoadData()
```

戻り値

ロードできた場合は True が、できなかった場合は False が返ります。

使用例

Visual Basic 6.0

```
.DataFileName = ".¥SavoyGem.bop"
.LoadData
```

Visual C++ 6.0

```
m_ctrl.SetDataFileName(".¥SavoyGem.bop");
m_ctrl.LoadData();
```

特記事項

参照

DataFileName プロパティ

3.3.7 LoadIniFile

設定内容を INI ファイルから読み出します。

構文

Visual Basic 6.0

```
Function LoadIniFile() As Boolean
```

Visual C++ 6.0

```
BOOL LoadIniFile()
```

戻り値

ロードできた場合は True が、できなかった場合は False が返ります。

使用例

Visual Basic 6.0

```
.IniFileName = “./SavoyGem.ini”
.LoadIniFile
```

Visual C++ 6.0

```
m_ctrl.SetIniFileName(“./SavoyGem.ini”);
m_ctrl.LoadIniFile();
```

特記事項

参照

3.3.8 RegisterALID

ALID を登録します。

構文

Visual Basic 6.0

```
Function RegisterALID(lALID As Long, sALCD As Integer, lpszALTX As String) As Boolean
```

Visual C++ 6.0

```
BOOL RegisterALID(long lALID, short sALCD, LPCTSTR lpszALTX)
```

引数	説明
lALID	アラーム ID
sALCD	アラームコード
lpszALTX	アラームテキスト

戻り値

登録に成功した場合は True が、失敗した場合は False が返ります。

使用例

Visual Basic 6.0

```
.RegisterALID 5000, 1, “”
```

Visual C++ 6.0

```
m_ctrl.RegisterALID(5000, 1, “”);
```

特記事項

S5F3 アラーム報告有効/無効送信(EAS)での設定に影響しますので、アプリケーション起動時の RegisterALID メソッドの使用は推奨できません。基本的には事前に登録しておき、LoadData メソッドで読み出すようにしてください。

参照

3.3.9 RegisterCEID

CEID を登録します。

構文

Visual Basic 6.0

```
RegisterCEID(lCEID As Long, sPredefinedCEID As Integer, lpszDescription As String) As Boolean
```

Visual C++ 6.0

```
BOOL RegisterCEID(long lCEID, short sPredefinedCEID, LPCTSTR lpszDescription);
```

引数	説明
lCEID	イベント ID
sPredefinedCEID	定義済み CEID
lpszDescription	CEID の説明

戻り値

登録に成功した場合は True が、失敗した場合は False が返ります。

使用例

Visual Basic 6.0

```
.RegisterCEID 253, 1, “”
```

Visual C++ 6.0

```
m_ctrl.RegisterCEID(253, 1, “”);
```

特記事項

アプリケーション起動時の RegisterCEID メソッドの使用は推奨できません。基本的には事前に登録しておき、LoadData メソッドで読み出すようにしてください。

参照

3.3.10 RegisterVID

VID を登録します。

構文

Visual Basic 6.0

```
Function RegisterVID(lVID As Long, sType As Integer, sNodeType As Integer, lpszMin As String, lpszMax As String, lpszDefault As String, lpszUnit As String, lpszDescription As String) As Boolean
```

Visual C++ 6.0

```
BOOL RegisterVID(long lVID, short sType, short sNodeType, LPCTSTR lpszMin, LPCTSTR lpszMax, LPCTSTR lpszDefault, LPCTSTR lpszUnit, LPCTSTR lpszDescription)
```

引数	説明
lVID	変数 ID
sType	変数のタイプ（ECID, SVID, DVID のいずれか）
sNodeType	SECS-II のノード型
lpszMin	ECMIN（最小値）
lpszMax	ECMAX（最大値）
lpszDefault	ECDEF（デフォルト値）
lpszUnit	UNITS（単位）
lpszDescription	ECNAME（説明）

戻り値

登録に成功した場合は True が、失敗した場合は False が返ります。

使用例

Visual Basic 6.0

```
.RegisterVID 1100, 2, 15, “0”, “10”, “5”, “mW”, “Laser level”
```

Visual C++ 6.0

```
m_ctrl.RegisterVID(1100, 2, 15, “0”, “10”, “5”, “mW”, “Laser level”);
```

特記事項

アプリケーション起動時の RegisterVID メソッドの使用は推奨できません。基本的には事前に登録しておき、LoadData メソッドで読み出すようにしてください。

参照

3.3.11 SaveData

SavoyGem データファイルにデータを保存します。

構文

Visual Basic 6.0

```
Function SaveData() As Boolean
```

Visual C++ 6.0

```
BOOL SaveData()
```

戻り値

正常に保存された場合は True が、失敗した場合は False が返ります。

使用例

Visual Basic 6.0

```
.SaveData
```

Visual C++ 6.0

```
m_ctrl.SaveData();
```

特記事項

参照

3.3.12 Send

WorkSpace プロパティと Reply プロパティで選択されている SECS-II メッセージを送信します。

構文

Visual Basic 6.0

```
Function Send() As Boolean
```

Visual C++ 6.0

```
BOOL Send()
```

戻り値

正常に送信された場合は True が、送信できなかった場合は False が返ります。

使用例

Visual Basic 6.0

```
.Send
```

Visual C++ 6.0

```
m_ctrl.Send():
```

特記事項

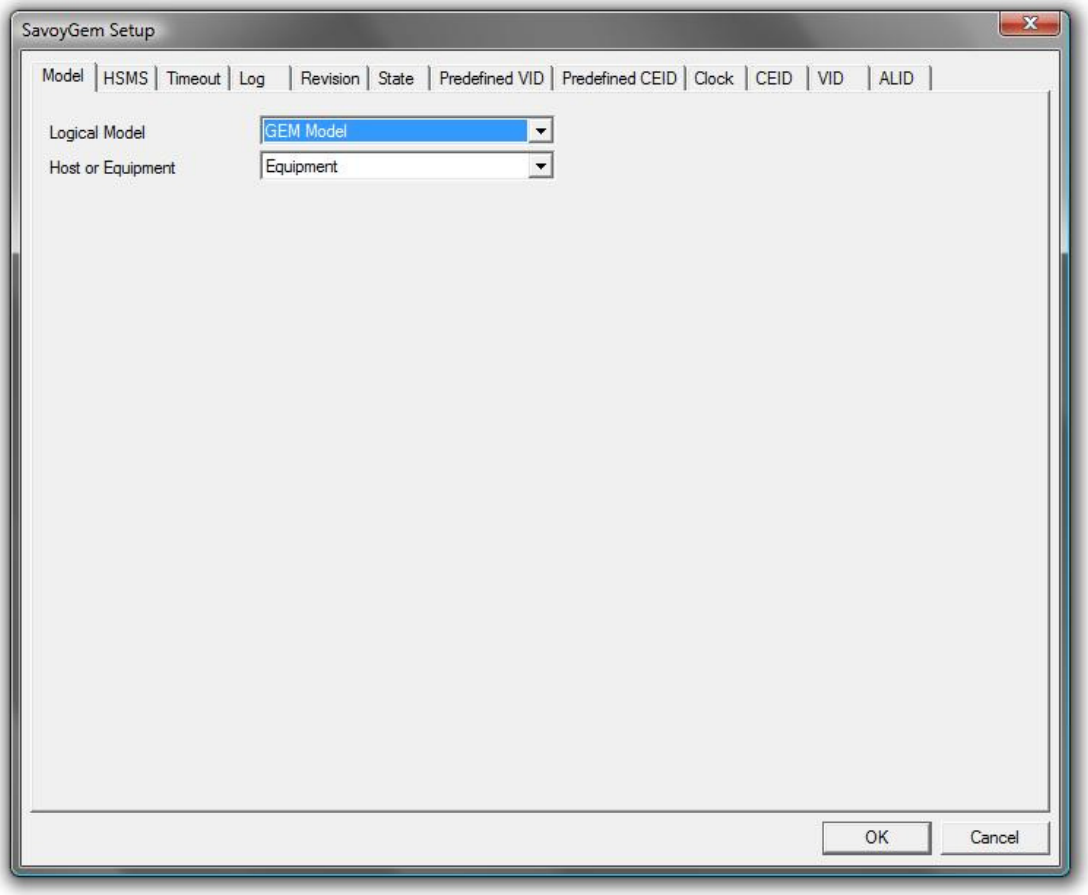
参照

WorkSpace プロパティ
Reply プロパティ

3.3.13 Setup

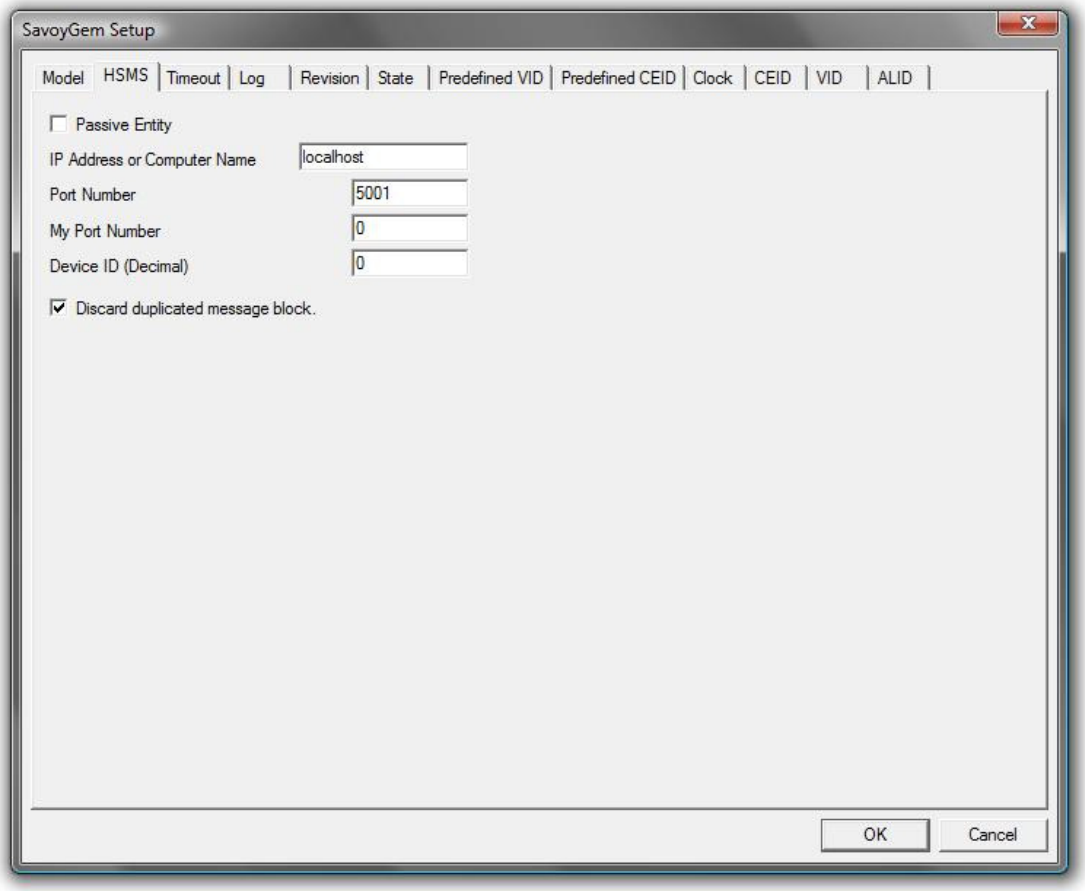
セットアップ画面を表示します。

Model タブ



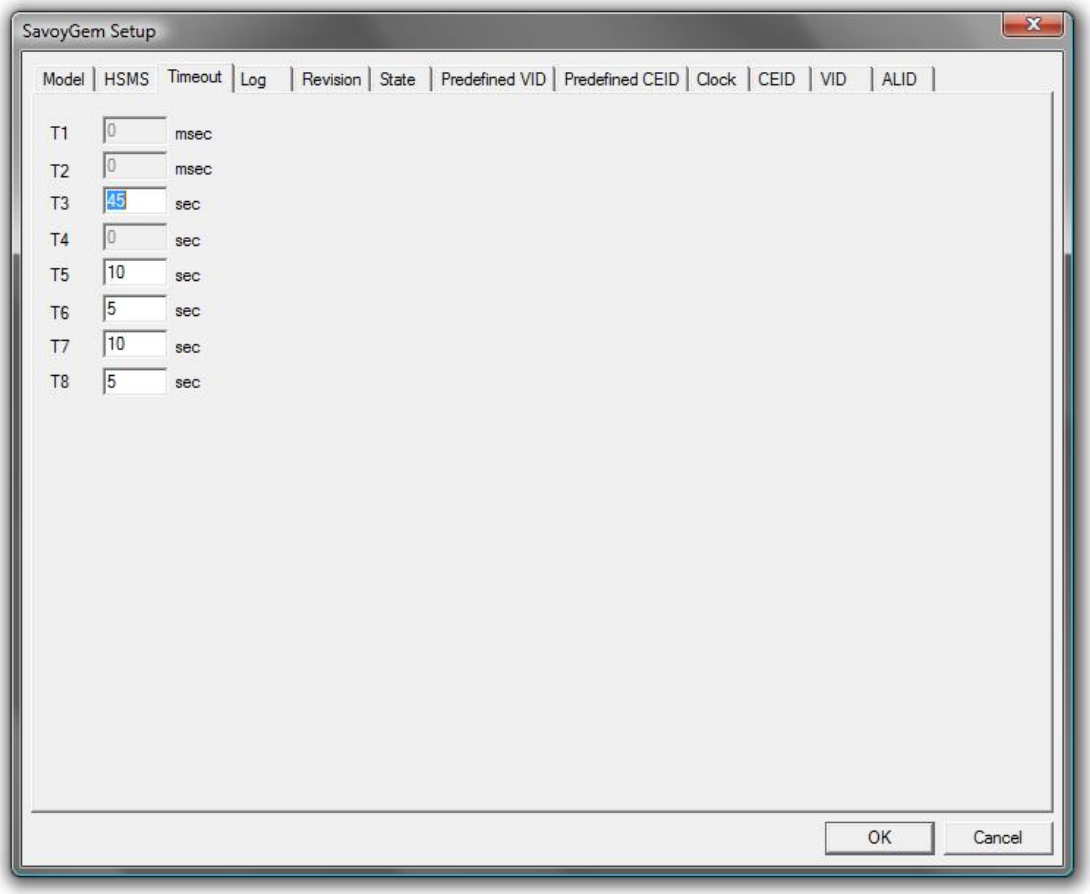
項目	説明
Logical Model	論理接続モデルを指定します。常に GEM Model を選択してください。
Host or Equipment	役割を指定します。常に Equipment を選択してください。

HSMS タブ



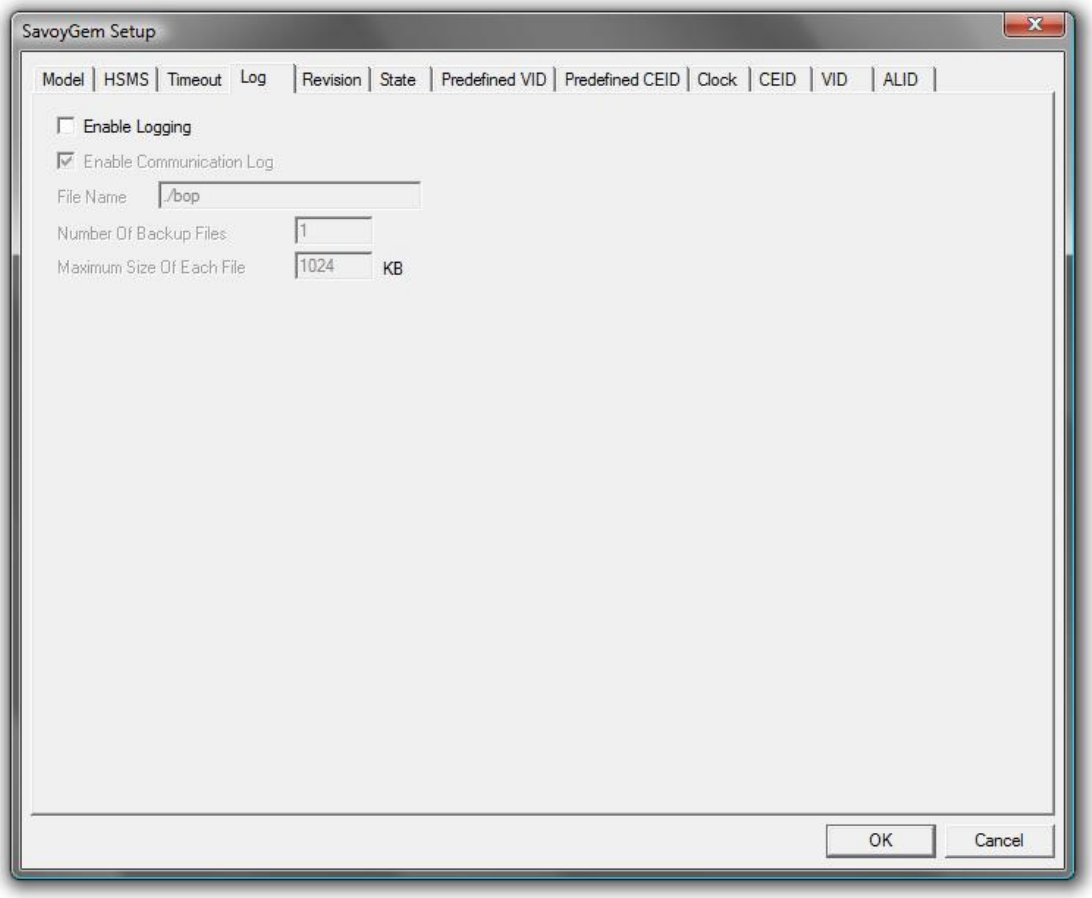
項目	説明
Passive Entity	パッシブエンティティ（サーバ）の場合はチェックをつけます。
IP Address or Computer Name	相手の IP アドレス、またはコンピュータ名。
Port Number	相手のポート番号。
My Port Number	自分のポート番号。アクティブエンティティの場合は 0 を推奨します。
Device ID (Decimal)	デバイス ID。
Discard duplicated block	二重ブロックを禁止するかどうかです。

Timeout タブ



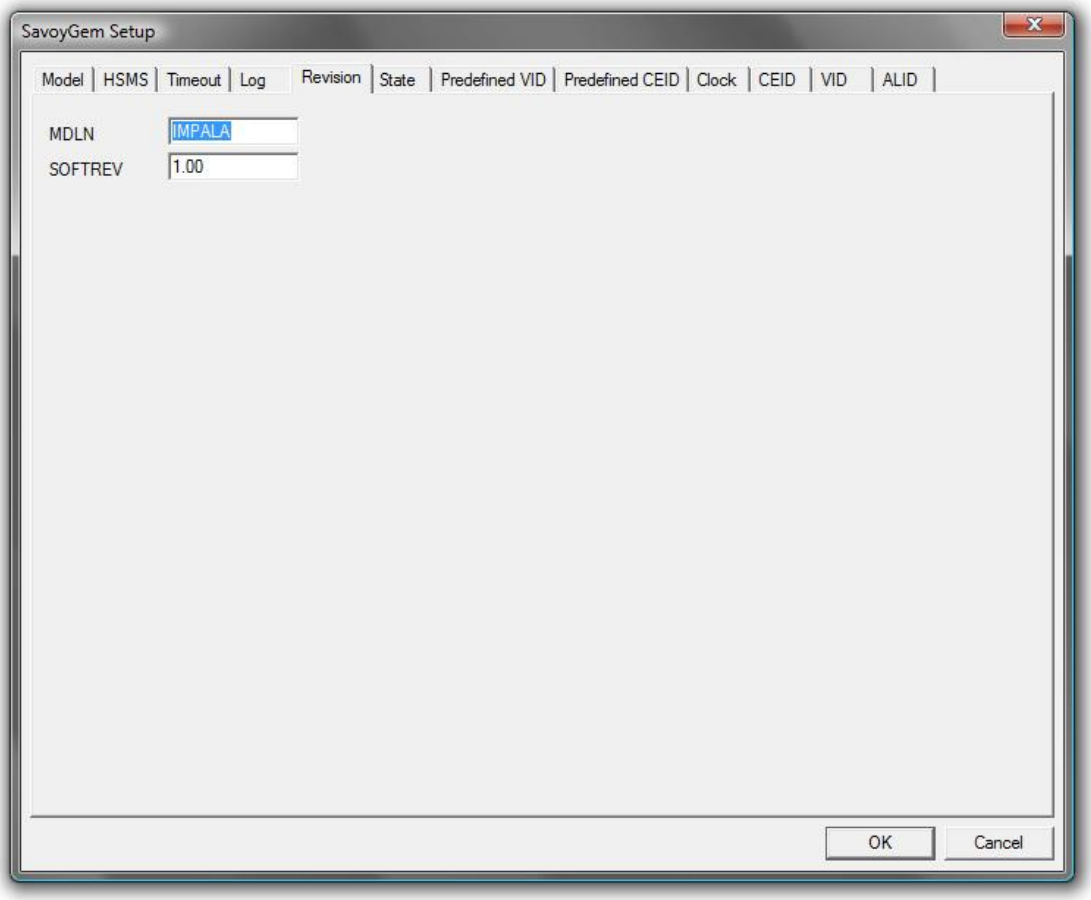
項目	説明
T3	T3 タイマー。
T5	T5 タイマー。
T6	T6 タイマー。
T7	T7 タイマー。
T8	T8 タイマー。

Log タブ



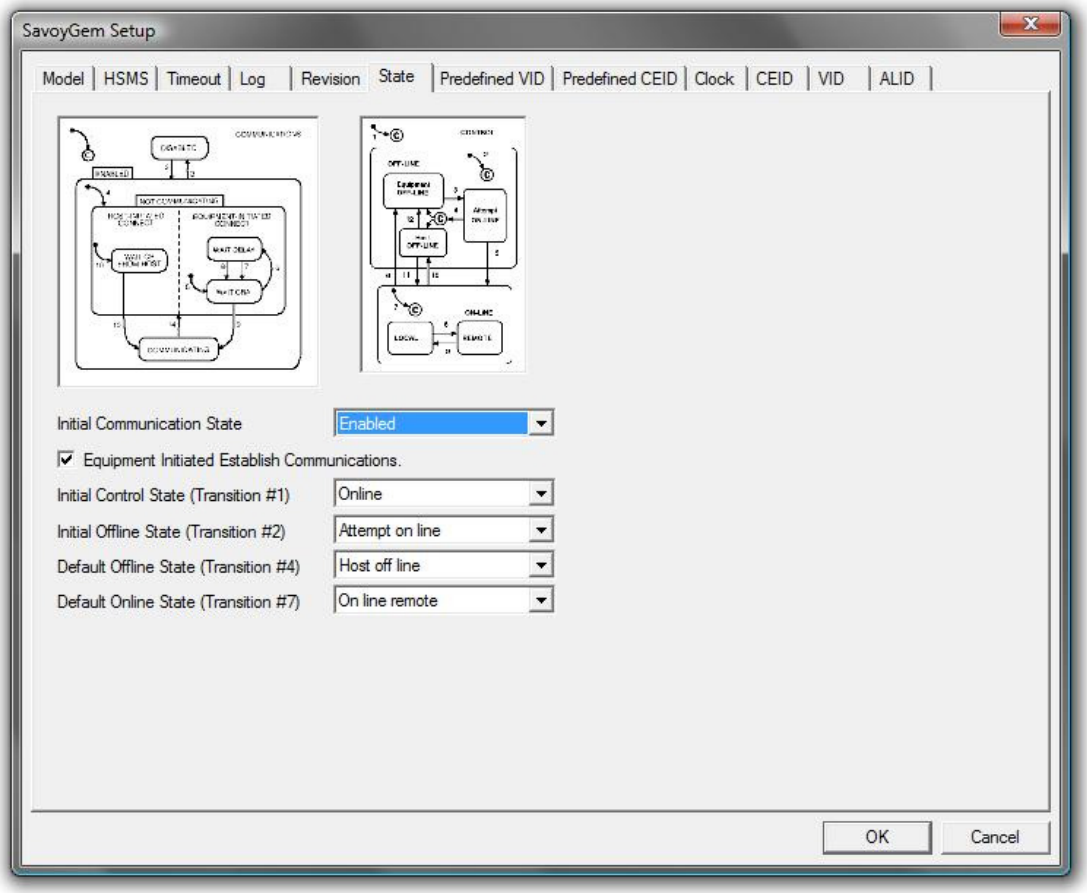
項目	説明
Enable Logging	ログに記録する場合はチェックをつけます。
Enable Communication Log	詳細の通信ログを記録する場合はチェックをつけます。
File Name	ログファイル名。
Number Of Backup Files	バックアップファイルの数。
Maximum Size Of Each File	ログファイルのサイズ。

Revision タブ



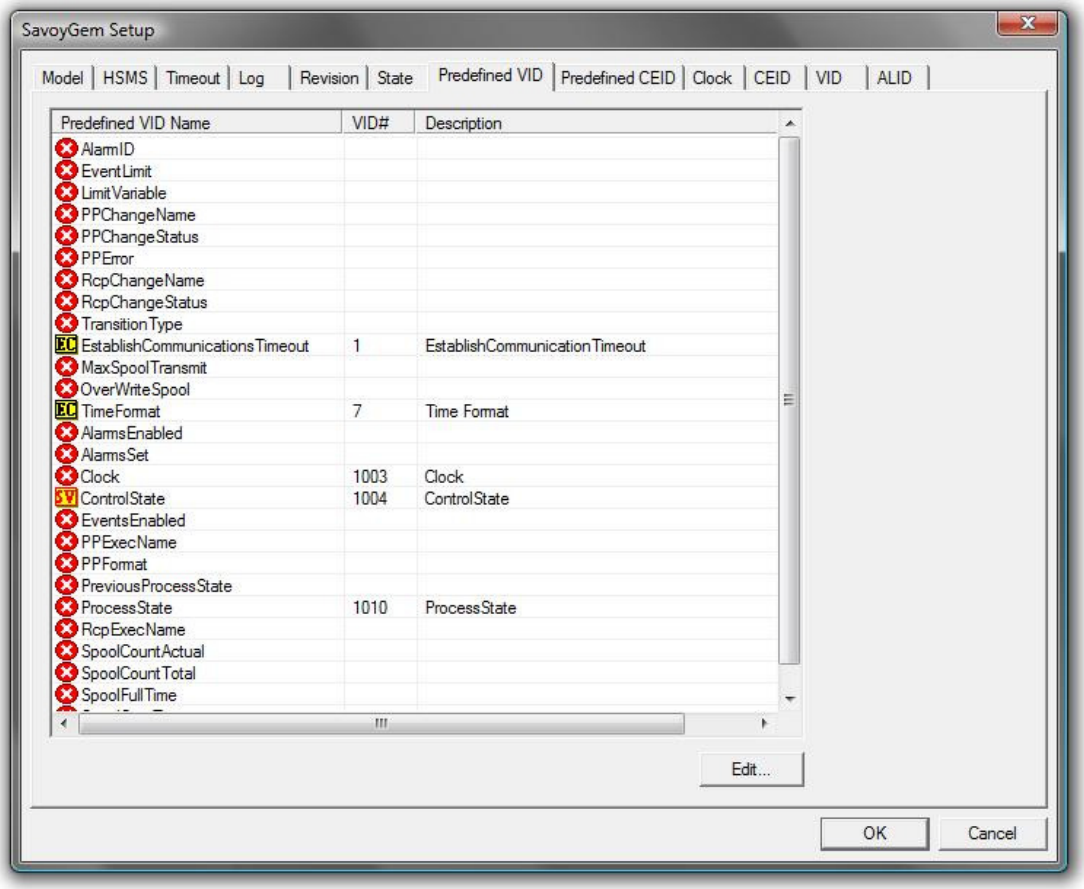
項目	説明
MDLN	装置のモデル名です。
SOFTREV	装置のリビジョン番号です。

State タブ

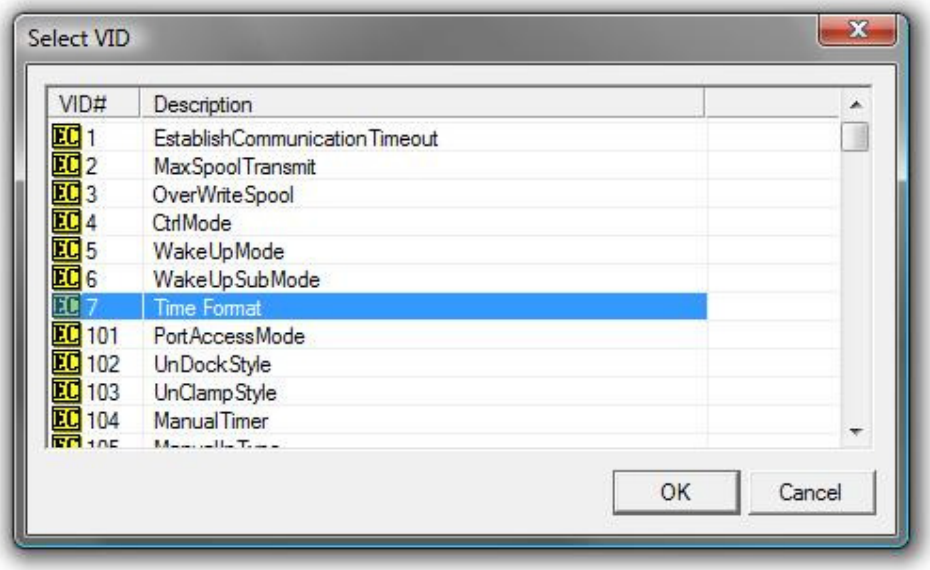


項目	説明
Initial Communication State	アプリケーション起動時に通信を有効にするかどうかです。
Equipment Initiated EC	装置から S1F13 を送信する場合は有効にします。
Initial Control State	起動時のコントロール状態モデルです。
Initial Offline State	起動時のオフライン状態です。
Default Offline State	デフォルトのオフライン状態です。
Default Online State	デフォルトのオンライン状態です。

Predefined VID タブ



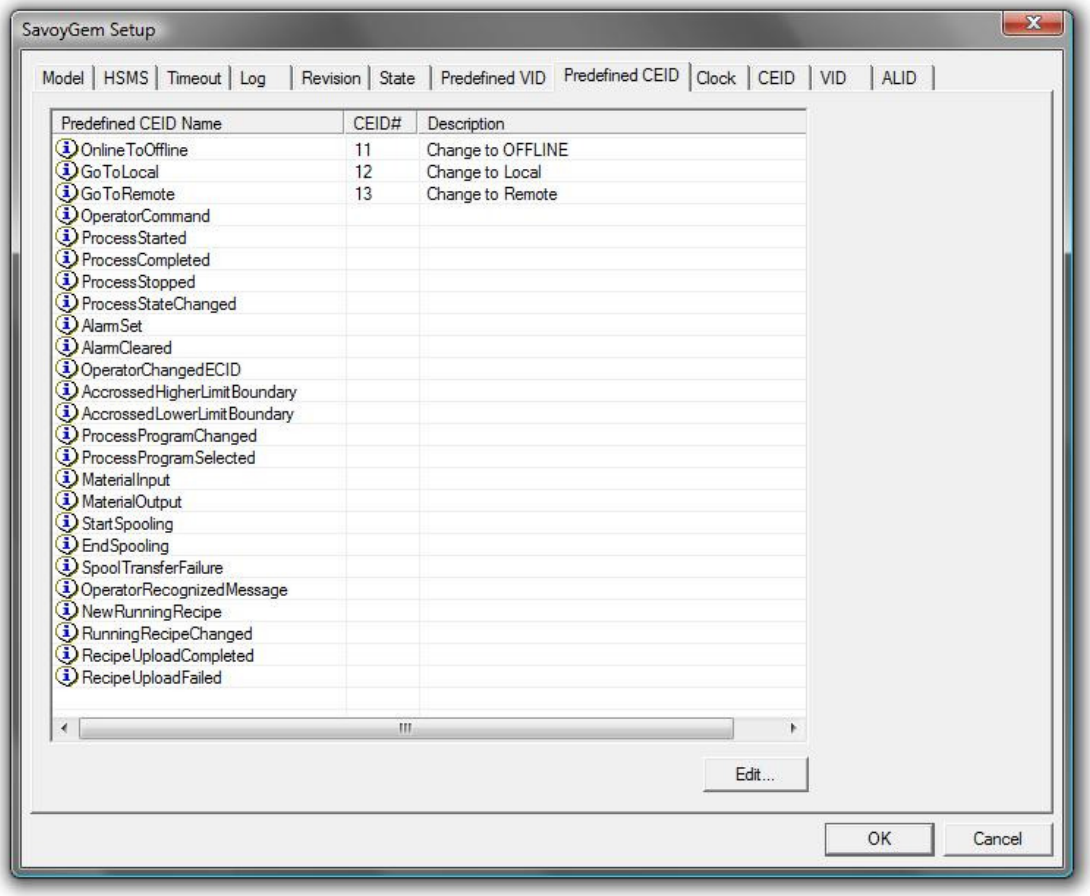
項目	説明
リスト	SavoyGem で定義済みの VID の一覧です。
Edit ボタン	VID を選択してこのボタンを押すと、編集画面が表示されます。



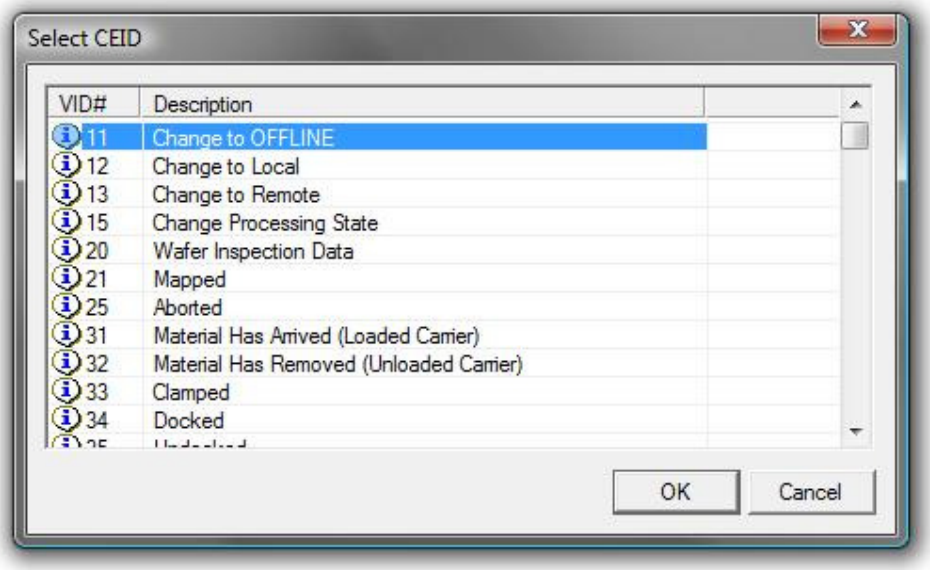
項目	説明
リスト	登録されている VID の一覧です。

OK ボタン	VID を選択してこのボタンを押すと、その VID が関連付けられます。
Cancel ボタン	編集を中止します。

Predefined CEID タブ

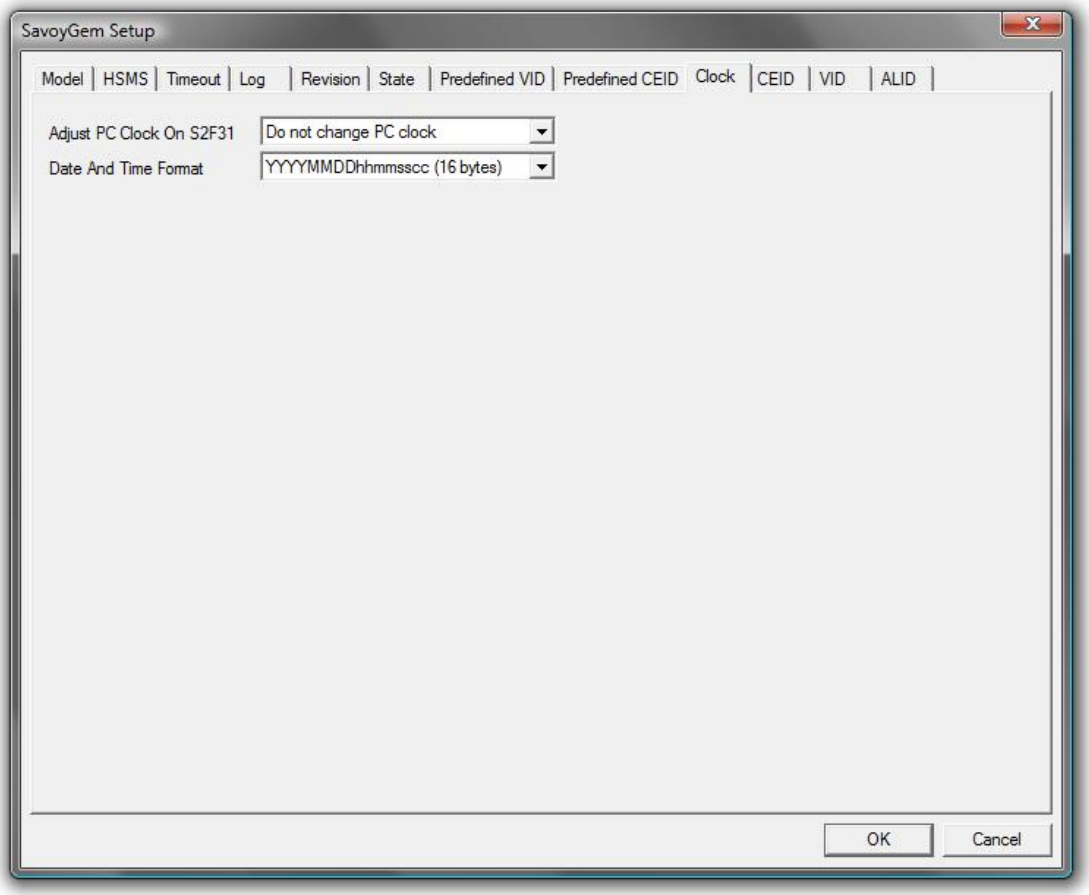


項目	説明
リスト	SavoyGem で定義済みの CEID の一覧です。
Edit ボタン	CEID を選択してこのボタンを押すと、編集画面が表示されます。



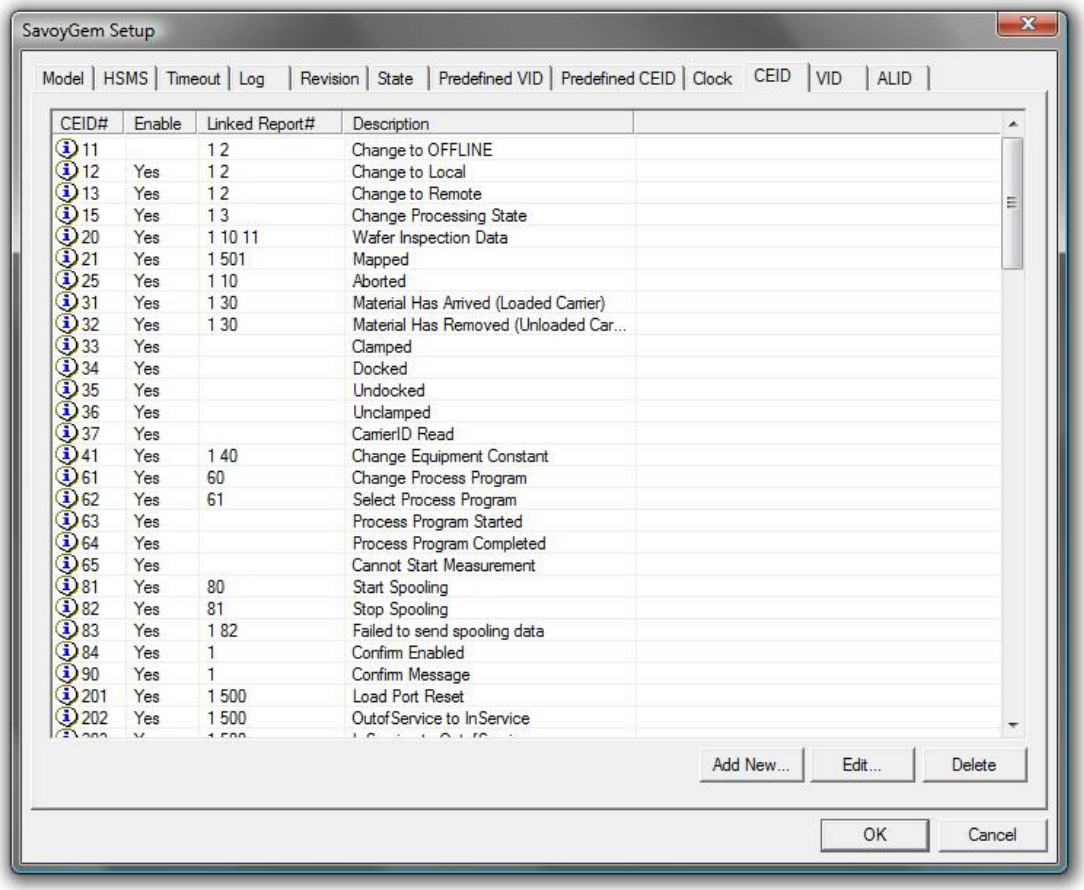
項目	説明
リスト	登録されている CEID の一覧です。
OK ボタン	CEID を選択してこのボタンを押すと、その CEID が関連付けられます。
Cancel ボタン	編集を中止します。

Clock タブ

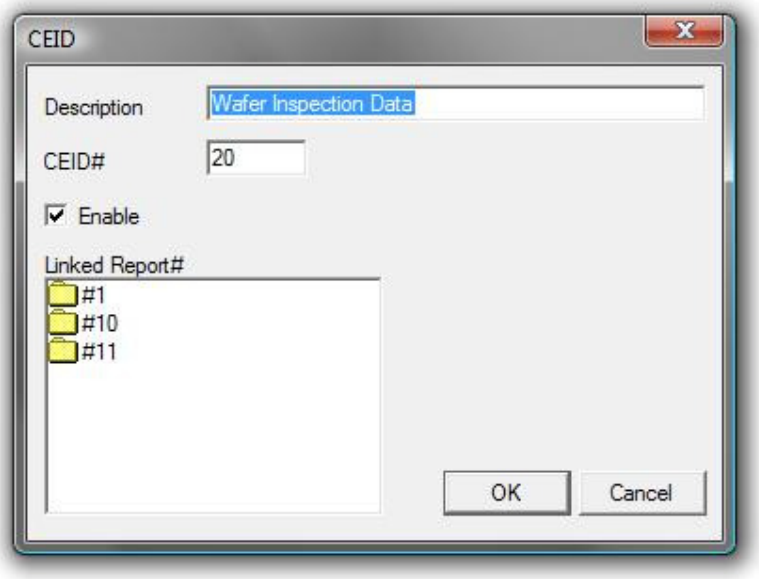


項目	説明
Adjust PC Clock on S2F31	S2F31 を受信した際に、PC の時計を更新するかどうかです。
Date And Time Format	日時の表現フォーマットを選択します。16 バイトが新しい表現です。

CEID タブ

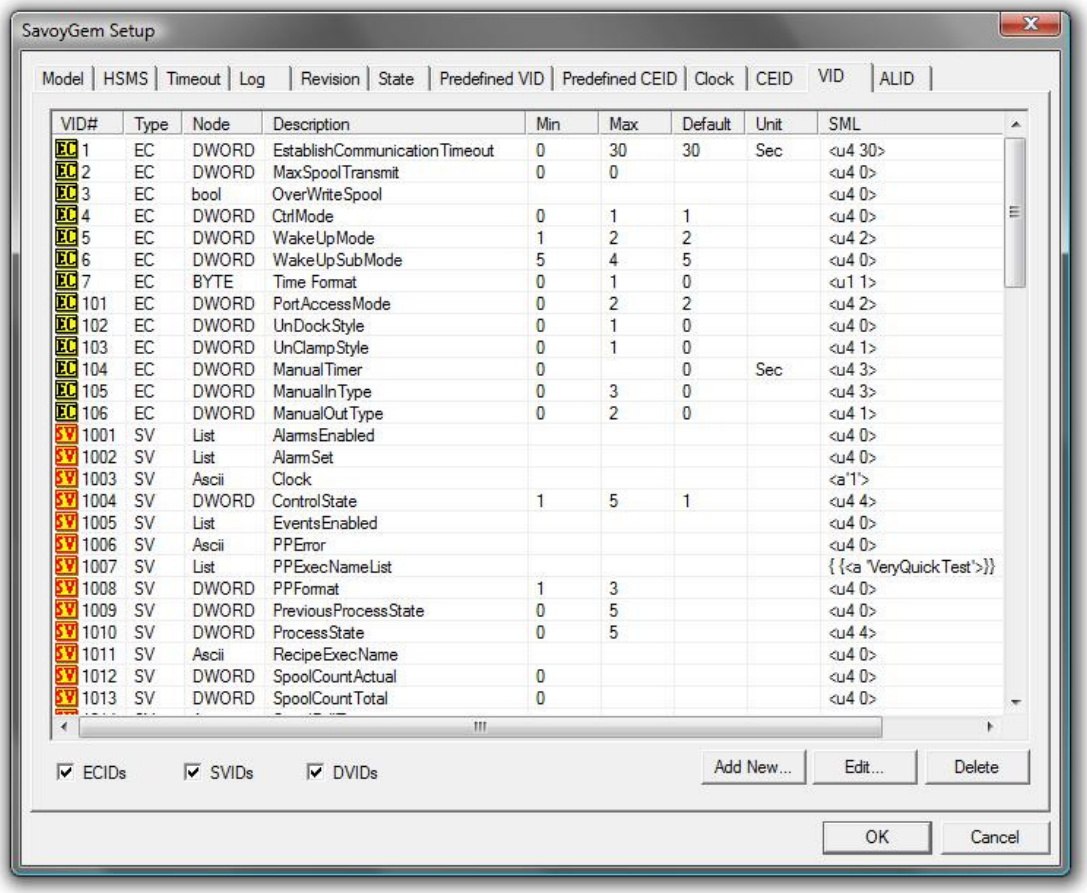


項目	説明
リスト	登録されている CEID の一覧です。
Add New ボタン	新規に CEID を登録します。
Edit ボタン	CEID を選択してこのボタンを押すと、編集画面が表示されます。
Delete ボタン	CEID を選択してこのボタンを押すと、その CEID を削除します。確認画面が表示されます。

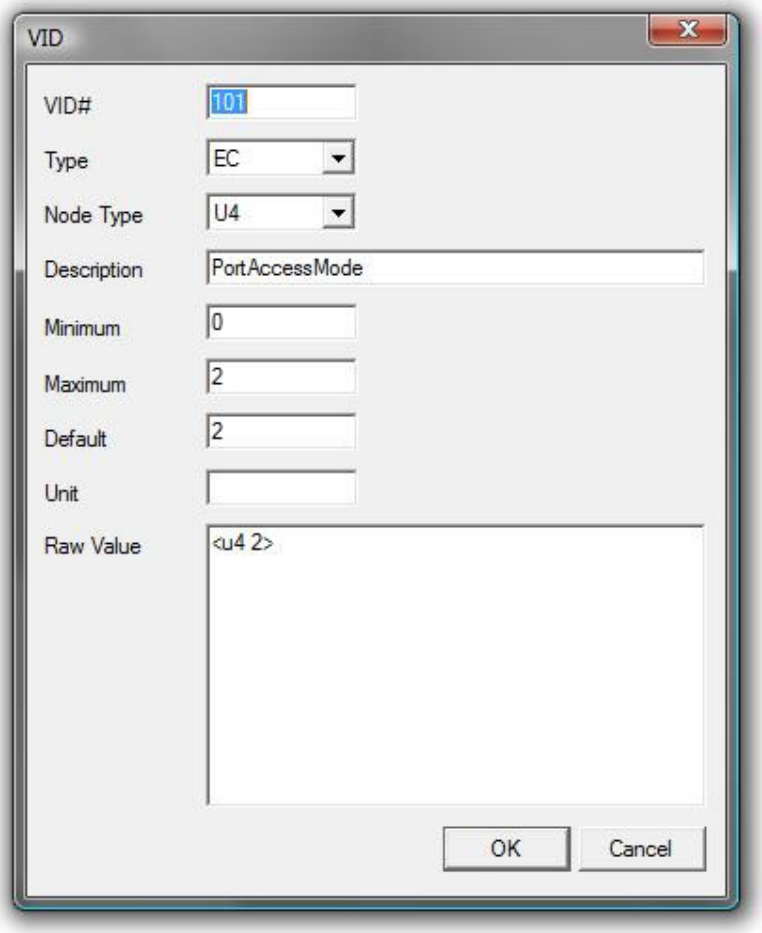


項目	説明
Description	CEID の説明です。
CEID#	CEID 番号です。
Enable	有効・無効を選択します。
Linked Report#	リンクされているレポートを表示します。
OK ボタン	設定を保存します。
Cancel ボタン	編集を中止します。

VID タブ

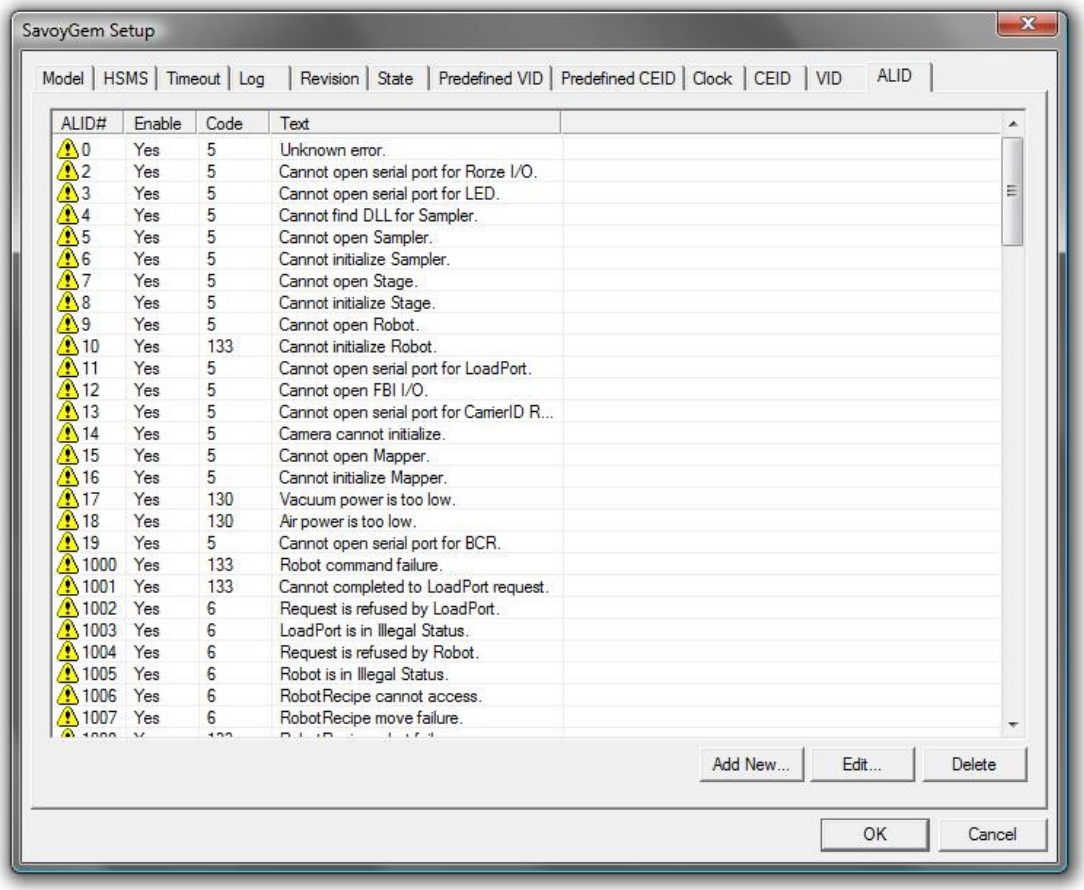


項目	説明
リスト	登録されている VID の一覧です。
ECIDs	チェックされているときは、装置定数を表示します。
SVIDs	チェックされているときは、システム変数を表示します。
DVIDs	チェックされているときは、データ変数を表示します。
Add New ボタン	新規に VID を登録します。
Edit ボタン	VID を選択してこのボタンを押すと、編集画面が表示されます。
Delete ボタン	VID を選択してこのボタンを押すと、その VID を削除します。確認画面が表示されます。

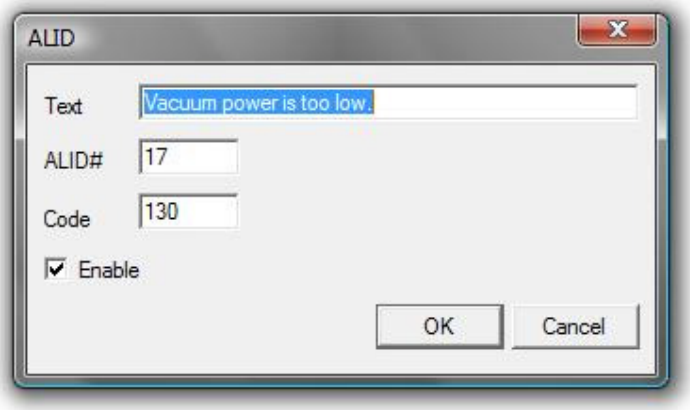


項目	説明
VID#	変数 ID 番号です。
Type	変数のタイプです。
Node Type	変数の型です。
Description	変数の説明です。
Minimum	最小値です。
Maximum	最大値です。
Default	デフォルト値です。
Unit	単位です。
Raw Value	SML 文字列を指定します。この値は随時更新されます。
OK ボタン	設定を保存します。
Cancel ボタン	編集を中止します。

ALID タブ



項目	説明
リスト	登録されている ALID の一覧です。
Add New ボタン	新規に ALID を登録します。
Edit ボタン	ALID を選択してこのボタンを押すと、編集画面が表示されます。
Delete ボタン	ALID を選択してこのボタンを押すと、その ALID を削除します。確認画面が表示されます。



項目	説明
Text	アラームテキストです。
ALID#	アラーム ID 番号です。
Code	アラームコードです。
Enable	有効・無効を設定します。

OK ボタン	設定を保存します。
Cancel ボタン	編集を中止します。

構文

Visual Basic 6.0

```
Function Setup(lpszCaption As String, lOptionFlag As Long) As Boolean
```

Visual C++ 6.0

```
BOOL Setup(LPCTSTR lpszCaption, long lOptionFlag)
```

引数	説明
lpszCaption	ダイアログボックスのキャプションタイトル。
lOptionFlag	表示するタブを指定します。下記の組み合わせになります。全てもタブ表示する場合は-1 を指定してください。

値	説明
0x0001	モデル
0x0002	HSMS
0x0004	タイムアウト
0x0008	リビジョン
0x0010	ステートモデル
0x0020	クロック
0x0040	CEID
0x0080	VID
0x0100	ALID
0x0200	ログファイル
0x0400	定義済み VID
0x0800	定義済み CEID

戻り値

設定が変更された場合は True が、変更されなかった場合は False が返ります。

使用例

Visual Basic 6.0

```
.Setup “GEM”, -1
```

Visual C++ 6.0

```
m_ctrl.Setup(“GEM”, -1);
```

特記事項

少なくとも1つのタブは指定する必要があります。

参照

3.3.14 ToALID

インデックスを ALID に変換します。インデックスとして使用可能な値は 0 から(ALIDCount - 1)までとなります。

構文

Visual Basic 6.0

```
Function ToALID(lIndex As Long) As Long
```

Visual C++ 6.0

```
long ToALID(long lIndex)
```

引数	説明
lIndex	インデックス。

戻り値

変換された ALID。インデックスが範囲外の場合はマイナス値が返ります。

使用例

Visual Basic 6.0

```
Dim lALID As Long
lALID = .ToALID(0)
```

Visual C++ 6.0

```
long lALID = m_ctrl.ToALID(0);
```

特記事項

参照

3.3.15 ToCEID

インデックスを CEID に変換します。インデックスとして使用可能な値は 0 から(CEIDCount - 1)までとなります。

構文

Visual Basic 6.0

```
Function ToCEID(lIndex As Long) As Long
```

Visual C++ 6.0

```
long ToCEID(long lIndex)
```

引数	説明
lIndex	インデックス。

戻り値

変換された CEID。インデックスが範囲外の場合はマイナス値が返ります。

使用例

Visual Basic 6.0

```
Dim lCEID As Long
lCEID = .ToCEID(0)
```

Visual C++ 6.0

```
long lCEID = m_ctrl.ToCEID(0);
```

特記事項

参照

3.3.16 ToVID

インデックスを VID に変換します。インデックスとして使用可能な値は 0 から(VIDCount - 1)までとなります。

構文

Visual Basic 6.0

```
Function ToVID(lIndex As Long) As Long
```

Visual C++ 6.0

```
long ToVID(long lIndex)
```

引数	説明
lIndex	インデックス。

戻り値

変換された VID。インデックスが範囲外の場合はマイナス値が返ります。

使用例

Visual Basic 6.0

```
Dim lVID As Long
lVID = .ToVID(0)
```

Visual C++ 6.0

```
long lVID = m_ctrl.ToVID(0);
```

特記事項

参照

3.3.17 UnregisterALID

ALID を登録解除します。

構文

Visual Basic 6.0

```
Function UnregisterALID(lALID As Long) As Boolean
```

Visual C++ 6.0

```
BOOL UnregisterALID(long lALID)
```

引数	説明
lALID	アラーム ID

戻り値

削除に成功した場合は True が、失敗した場合は False が返ります。

使用例

Visual Basic 6.0

```
.UnregisterALID 325
```

Visual C++ 6.0

```
m_ctrl.UnregisterALID(325);
```

特記事項

参照

3.3.18 UnregisterVID

VID を登録解除します。

構文

Visual Basic 6.0

```
Function UnregisterVID(lVID As Long) As Boolean
```

Visual C++ 6.0

```
BOOL UnregisterVID(long lVID)
```

引数	説明
lVID	変数 ID

戻り値

削除に成功した場合は True が、失敗した場合は False が返ります。

使用例

Visual Basic 6.0

```
.UnregisterVID 1100
```

Visual C++ 6.0

```
m_ctrl.UnregisterVID(1100);
```

特記事項

参照

3.3.19 WriteLogFile

指定された文字列をログファイルに書き込みます。

構文

Visual Basic 6.0

```
Function WriteLogFile(lpszText As String) As Boolean
```

Visual C++ 6.0

```
BOOL WriteLogFile(LPCTSTR lpszText)
```

引数	説明
lpszText	書き出す文字列

戻り値

正しくログに書き出せた場合は True が、書き出せなかった場合は False が返ります。

使用例

Visual Basic 6.0

```
.WriteLogFile “This is a test”
```

Visual C++ 6.0

```
m_ctrl.WriteLogFile(“This is a test”);
```

特記事項

参照

3.4 イベント

3.4.1 CommunicationStateChanged

通信状態が変化したときに通知されます。通信状態は以下のいずれかです。

値	説明
0	通信が無効
1	通信が中断
2	通信している

構文

Visual Basic 6.0

```
Event CommunicationStateChanged(sNewState As Integer, sPrevState As Integer)
```

Visual C++ 6.0

```
void OnCommunicationStateChanged(short sNewState, short sPrevState)
```

引数	説明
sNewState	新しい通信状態
sPrevState	前の通信状態

使用例

Visual Basic 6.0

```
Text1.Text = Format$(sPrevState) + " -->" + Format$(sNewState)
```

Visual C++ 6.0

```
TRACE("%d --> %d", sPrevState, sNewState);
```

特記事項

参照

3.4.2 Connected

HSMS 接続が成立したときに通知されます。

構文

Visual Basic 6.0

```
Event Connected(lpszIPAddress As String, lPortNumber As Long)
```

Visual C++ 6.0

```
void OnConnected(LPCTSTR lpszIPAddress, long lPortNumber)
```

引数	説明
lpszIPAddress	接続相手のIPアドレス
lPortNumber	接続相手のポート番号

使用例

Visual Basic 6.0

```
Text1.Text = “Connected - ” + lpszIPAddress + “ [“ + Format$(lPortNumber) + “]”
```

Visual C++ 6.0

```
TRACE(“Connected - %s [%d]”,lpszIPAddress,lPortNumber);
```

特記事項

参照

3.4.3 ConnectionStateChanged

HSMS 接続状態が変化したときに通知されます。HSMS 接続状態は以下のいずれかです。

値	説明
0	未接続
1	接続(未選択)
2	接続(選択)

構文

Visual Basic 6.0

```
Event ConnectionStateChanged(sNewState As Integer, sPrevState As Integer)
```

Visual C++ 6.0

```
void OnConnectionStateChanged(short sNewState, short sPrevState)
```

引数	説明
sNewState	新しい通信状態
sPrevState	前の通信状態

使用例

Visual Basic 6.0

```
Text1.Text = Format$(sPrevState) + " -->" + Format$(sNewState)
```

Visual C++ 6.0

```
TRACE("%d --> %d", sPrevState, sNewState);
```

特記事項

参照

3.4.4 ControlStateChanged

コントロール状態が変化したときに通知されます。コントロール状態は以下のいずれかです。

値	説明
0	装置オフライン
1	オンライン試行
2	ホストオフライン
3	オンラインローカル
4	オンラインリモート

構文

Visual Basic 6.0

```
Event ControlStateChanged(sNewState As Integer, sPrevState As Integer)
```

Visual C++ 6.0

```
void OnControlStateChanged(short sNewState, short sPrevState)
```

引数	説明
sNewState	新しいコントロール状態
sPrevState	前のコントロール状態

使用例

Visual Basic 6.0

```
Text1.Text = Format$(sPrevState) + " -->" + Format$(sNewState)
```

Visual C++ 6.0

```
TRACE("%d --> %d", sPrevState, sNewState);
```

特記事項

参照

3.4.5 Disconnected

HSMS 接続が切断したときに通知されます。

構文

Visual Basic 6.0

```
Event Disconnected(lpszIPAddress As String, lPortNumber As Long)
```

Visual C++ 6.0

```
void OnDisconnected(LPCTSTR lpszIPAddress, long lPortNumber)
```

引数	説明
lpszIPAddress	切断されたIPアドレス
lPortNumber	切断されたポート番号

使用例

Visual Basic 6.0

```
Text1.Text = “Disconnected - “ + lpszIPAddress + “ [“ + Format$(lPortNumber) + “]”
```

Visual C++ 6.0

```
TRACE(“Disconnected - %s [%d]”,lpszIPAddress,lPortNumber);
```

特記事項

参照

3.4.6 Problem

エラーが発生したときに通知されます。

構文

Visual Basic 6.0

```
Event Problem(sErrorCode As Integer, lpszAdditionalInfo As String)
```

Visual C++ 6.0

```
void OnProblem(short sErrorCode, LPCTSTR lpszAdditionalInfo)
```

引数	説明
sErrorCode	エラーコード。エラーコードは以下のいずれかの値です。
lpszAdditionalInfo	追加情報。現在のところは未使用となっています。

sErrorCode	列挙型	説明
1	ErrorGemHsmsLengthTooShort	レングスより短い HSMS メッセージを受信
2	ErrorGemHsmsExceedMaxLength	最大バッファサイズより長いメッセージを受信
3	ErrorGemHsmsT8Timeout	T8 タイムアウト
4	ErrorGemHsmsT6Timeout	T6 タイムアウト
5	ErrorGemHsmsT7Timeout	T7 タイムアウト
6	ErrorGemHsmsInvalidSessionID	セッション ID が不一致
7	ErrorGemHsmsInvalidPType	P タイプが未定義
8	ErrorGemHsmsControlMessageTooLong	制御メッセージなのに 10 バイト以上ある
9	ErrorGemHsmsInvalidSType	S タイプが未定義
10	ErrorGemHsmsInvalidFunction	ファンクションが正しくない
11	ErrorGemHsmsInvalidRejectReason	理由コードが不正
12	ErrorGemHsmsT3Timeout	T3 タイムアウト
13	ErrorGemHsmsT5Timeout	T5 タイムアウト

使用例

Visual Basic 6.0

```
Text1.Text = “Problem - Code : “ + Format$(sErrorCode)
```

Visual C++ 6.0

```
TRACE(“Problem - %d”,sErrorCode);
```

特記事項

参照

3.4.7 Received

SavoyGem コントロールが HSMS 接続経由で SECS-II メッセージを受信したときに通知されます。

構文

Visual Basic 6.0

```
Event Received(lpszIPAddress As String, lPortNumber As Long)
```

Visual C++ 6.0

```
void OnReceived(LPCTSTR lpszIPAddress, long lPortNumber)
```

引数	説明
lpszIPAddress	送信元のIPアドレス
lPortNumber	送信元のポート番号

使用例

Visual Basic 6.0

```
Text1.Text = “Received - ” + lpszIPAddress + “ [“ + Format$(lPortNumber) + “]”
```

Visual C++ 6.0

```
TRACE(“Received - %s [%d]”,lpszIPAddress,lPortNumber);
```

特記事項

参照

3.4.8 Sent

SECS-II メッセージが送信されたときに通知されます。

構文

Visual Basic 6.0

```
Event Sent()
```

Visual C++ 6.0

```
void OnSent()
```

使用例

Visual Basic 6.0

```
Text1.Text = “Sent”
```

Visual C++ 6.0

```
TRACE(“Sent”);
```

特記事項

参照

3.4.9 VIDChanged

VID が変更されたときに通知されます。

構文

Visual Basic 6.0

```
Event VIDChanged(lVID As Long)
```

Visual C++ 6.0

```
void OnVIDChanged(long lVID)
```

引数	説明
lVID	変化した VID

使用例

Visual Basic 6.0

```
Text1.Text = “VID Changed - ” + Format$(lVID)
```

Visual C++ 6.0

```
TRACE(“VID Changed - %d”,lVID);
```

特記事項

参照