SavoyGem ActiveX Control
User Guide

Jazz Soft, Inc.

1 Revision History

Version	Date	Name	Description
1.00	Jul, 31st, 2009	Hikaru Okada	Created as new document
1.00a	Aug, 22nd, 2009	Hikaru Okada	Splitted into separate document, since number of pages became large.
1.00b	Aug, 30th, 2009	Hikaru Okada	Mended some incorrect description.
1.00c	Oct, 12th, 2009	Hikaru Okada	Added DataBakCount property.
1.00d	Dec, 24th, 2009	Hikaru Okada	Corrected the description of ControlStateSwitch.
1.00e	Dec, 14th, 2010	Hikaru Okada	Added ALIDEnable, ALIDSet, CEIDEnable properties and RegisterCEID method.

2 Table of Contents

3 SavoyGem

SavoyGem control is an assistant product to develop SEMI E30 (GEM) compliant communication application software. In general, implementing GEM feature will take tremendous work, but SavoyGem control will reduce most of them.

Properties

Name	Description
ALIDCount	Gets the number of registered ALID.
Appearance	Gets or sets the value that determines the appearance of a SavoyGem control.
BorderStyle	Gets or sets whether the SavoyGem control has a border.
CEIDCount	Gets the number of registered CEID.
CommunicationState	Gets or sets the communication state.
ControlState	Gets or sets GEM control state.
ControlStateSwitch	Sets the control state switch.
DataFileName	Gets or sets the SavoyGem data file name.
DeviceID	Gets or sets the device ID.
DiscardDuplicatedBlock	Gets or sets whether the SavoyGem control discards the duplicated block.
Function	Gets or sets the function number in SECS-II header.
Host	Gets or sets the role of SavoyGem control.
IniFileName	Gets or sets INI file name to read/write settings.
IPAddress	Gets or sets the IP address of passive entity computer for HSMS connection.
Log	Gets or sets whether logging is enabled.
LogBakCount	Gets or sets the number of back-up file for logging.
LogCommunication	Get or sets whether logging for communication part is enabled.
LogFileName	Get or sets the log file name.
LogicalConnection	Gets or sets the logical connection.
LogSize	Gets or sets the log file size in kilobyte.
Msg	Gets or sets the message data of SECS-II.
MyPortNumber	Gets or sets local portnumber for TCP/IP connection.
Node	Gets or sets the node for operation.
NodeCount	Gets or sets the number of sub items.
NodeType	Gets or sets the node type.
NodeValue	Gets or sets the node value.
NodeValueHex	Gets or sets the node value in hexadecimal expression.
OfflineRequest	Gets or sets the setting wether SavoyGem will accept S1F15 request off-line (ROFL).
OnlineRequest	Gets or sets the setting wether SavoyGem will accept S1F17 request on-line (RONL).
Reply	Gets or sets the work space mode.
PhysicalConnection	Gets or sets the physical connection.
PortNumber	Gets or sets the port number for TCP/IP connection.
PType	Gets or sets the presentation type in SECS-II header.
Server	Gets or sets the entity type.
SessionID	Gets or sets the session ID for HSMS.
SML	Gets or sets the message in SML string.
Stream	Gets or sets the stream in SECS-II header.
SType	Gets or sets the session type in SECS-II header.
SystemBytes	Gets or sets the system bytes in SECS-II header.
T1	Gets or sets the T1 time out for SECS-I in 1/10 seconds.
T2	Gets or sets the T2 time out for SECS-I in 1/10 seconds.
T3	Gets or sets the T3 time out in seconds.
T4	Gets or sets the T4 time out for SECS-I in seconds.
T5	Gets or sets the T5 time out for HSMS in seconds.
T6	Gets or sets the T6 time out for HSMS in seconds.
T7	Gets or sets the T7 time out for HSMS in seconds.
T8	Gets or sets the T8 time out for HSMS in seconds.
Verification	Gets and sets the verification result of message structure.
VIDCount	Gets or sets the number of variable ID.
ViewStyle	Gets or sets the view style of SavoyGem control.

Wbit	Gets or sets the wait bit in SECS-II header.
WorkSpace	Gets or sets the work space for SECS-II message.

Array Properties

Name	Description
ALCD	Gets or sets the alarm code for specified alarm ID.
ALTX	Gets and sets the alarm text for specified alarm ID.
CEIDDescription	Gets or sets the description for specified collection event ID.
VIDDefault	Gets and sets the default value for specified variable ID.
VIDDescription	Gets and sets the description for specified variable ID.
VIDMax	Gets and sets the maximum value for specified variable ID.
VIDMin	Gets and sets the minimum value for specified variable ID.
VIDNodeType	Gets and sets the node type for specified variable ID.
VIDRawValue	Gets and sets the raw value for specified variable ID.
VIDType	Gets and sets the variable type for specified variable ID.
VIDUnit	Gets and sets the unit for specified variable ID.
VIDValue	Gets and sets the value for specified variable ID.

Methods

Name	Description
AboutBox	Opens version information dialog box on the screen.
DefProc	Calls default procedure when SavoyGem control received message.
InvokeAlarm	Lets SavoyGem control attempt to send alarm event.
InvokeEvent	Lets SavoyGem control attempt to send collection event.
IsValidVID	Checks whether specified variable ID is valid.
LoadData	Loads .data from SavoyGem data file.
LoadIniFile	Loads settings from INI file and initialize properties.
RegisterALID	Registers alarm ID.
RegisterVID	Registers variable ID.
SaveData	Saves data into SavoyGem data file.
Send	Send message specified by WorkSpace property and Reply property.
Setup	Opens setup dialog box on the screen.
ToALID	Converts index to alarm ID.
ToCEID	Converts index to collection event ID.
ToVID	Converts index to variable ID.
UnregisterALID	Unregisters alarm ID.
UnregisterVID	Unregisters variable ID.
WriteLogFile	Writes literal string into log file.

Events

Name	Description
CommunicationStateChanged	Notifies that communication state has been changed.
Connected	Notifies that HSMS connection has been established.
ConnectionStateChanged	Notifies that connection state has been changed.
ControlStateChanged	Notifies that control state has been changed.
Disconnected	Notifies that HSMS connection has been disconnected.
Problem	Notifies that error has occurred.
Received	Notifies that SavoyGem control received message through HSMS.
Sent	Notifies that SECS-II message has been sent.
VIDChanged	Notifies that content of variable ID has been changed.

3.1 Properties

3.1.1 ALIDCount

Gets the number of registered ALID. If this value is 0, none is registered.

Syntax

Visual Basic 6.0

```
ALIDCount As Long
```

Visual C++ 6.0

```
long GetALIDCount()
```

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Read-only property.

Since ALIDCount property returns the number of registered ALID, available index range is between 0 and (ALIDCount – 1). Use ToALID method to convert index into ALID.

See Also

3.1.2 Appearance

Gets or sets the value that determines the appearance of a SavoyGem control.

Value	Description
0	Flat
1	Etched

Syntax

Visual Basic 6.0

```
Appearance As Integer
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Persistent property.

See Also

3.1.3 BorderStyle

Gets or sets whether the SavoyGem control has a border.

Value	Description
0	No border
1	Fixed single border

Syntax

Visual Basic 6.0

```
BorderStyle As Integer
```

Visual C++ 6.0

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

Example

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
```

Remarks

Persistent property.

See Also

3.1.4 CEIDCount

Gets the number of registered CEID. If this value is 0, none is registered.

Syntax

Visual Basic 6.0

```
CEIDCount As Long
```

Visual C++ 6.0

```
long GetCEIDCount()
```

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Read-only property.

Since CEIDCount property returns the number of registered CEID, available index range is between 0 and (CEIDCount – 1). Use ToCEID method to convert index into CEID.

See Also

3.1.5 CommunicationState

Gets or sets the communication state. Communication state is one of the followings:

Value	Description
0	Communication was disabled
1	Not communicating
2	Communicating

Syntax

Visual Basic 6.0

```
CommunicationState As Integer
```

Visual C++ 6.0

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

Example

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);
```

Remarks

See Also

3.1.6 ControlState

Gets or sets GEM control state. Control state is one of followings:

Value	Description
0	Equipment off-line
1	Attempt on-line
2	Host off-line
3	On-line local
4	On-line remote

Syntax

Visual Basic 6.0

```
ControlState As Integer
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Some control status transitions are not allowed. For this reason, please use ControlStateSwitch propertyto switch between on-line and off-line state.

See Also

3.1.7 ControlStateSwitch

Sets the control state switch.

Value	Description
True	Attempt to shift from off-line to on-line
False	Shift from on-line to off-line

Some control status transitions are not allowed. For this reason, please use ControlStateSwitch propertyto switch between on-line and off-line state.

Syntax

Visual Basic 6.0

```
ControlStateSwitch As Boolean
```

Visual C++ 6.0

```
void SetControlStateSwitch(BOOL)
```

Example

Visual Basic 6.0

```
.ControlStateSwitch = True
```

Visual C++ 6.0

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

Remarks

Write-only property.

If control state actually is changed, ControlStateChanged event will occur.

See Also

3.1.8 DataBakCount

Gets or sets the number of back-up file for data file. When data file needs to be reviced, SavoyGem control will rename the file name and make a new empty data file. If number of back-up file reached to the value of DataBakCount property, SavoyGem control will delete oldest back-up file.

Syntax

Visual Basic 6.0

```
DataBakCount As Integer
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.DataBakCount = 10
```

Visual C++ 6.0

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

Remarks

Persistent property.

See Also

3.1.9 DataFileName

Gets or sets the SavoyGem data file name.

Syntax

Visual Basic 6.0

```
DataFileName As String
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.DataFileName = “.¥SavoyGem.bop”
.LoadData
```

Visual C++ 6.0

```
m_ctrl.SetDataFileName(“.¥SavoyGem.bop”);
m_ctrl.LoadData();
```

Remarks

Persistent property.

See Also

LoadData method

3.1.10 DeviceID

Gets or sets the device ID. Device ID is 15 bits starting at second bit of SECS-II header.

For HSMS data message following header structure is used.

Byte	Description	
1	Session ID	
2		
3	W	Stream
4	Function	
5	P type	
6	S type	
7	System bytes	
8		
9		
10		

For HSMS control message following header structure is used.

Byte	Description
1	Session ID
2	
3	
4	
5	P type
6	S type
7	System bytes
8	
9	
10	

Syntax

Visual Basic 6.0

```
DeviceID As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.DeviceID = 1
```

Visual C++ 6.0

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

Remarks

Persistent property.

Device ID and session ID are almost same, but device ID is 15-bit, where session ID is 16-bit.

See Also

3.1.11 DiscardDuplicatedBlock

Gets or sets whether the SavoyGem control discards the duplicated block. If this property is true and SavoyGem control received message which has identical header as previous message, SavoyGem control will treat such message as duplicated block and ignore it.

Value	Description
True	Discard duplicated block.
False	Don’t care duplicated block and it will notified through Received event.

Syntax

Visual Basic 6.0

```
DiscardDuplicatedBlock As Boolean
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.DiscardDuplicatedBlock = True
```

Visual C++ 6.0

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

Remarks

Persistent property.

See Also

3.1.12 Function

Gets or sets the function number in SECS-II header.

For HSMS data message following header structure is used.

Byte	Description	
1	Session ID	
2		
3	W	Stream
4	Function	
5	P type	
6	S type	
7	System bytes	
8		
9		
10		

Syntax

Visual Basic 6.0

```
Function As Integer
```

Visual C++ 6.0

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

Example

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
    ...
```

Remarks

See Also

3.1.13 Host

Gets or sets the role of SavoyGem control. This property should always be false, since SavoyGem was intended to use for equipment.

Value	Description
False	Equipment
True	Host

Syntax

Visual Basic 6.0

```
Host As Boolean
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.Host = False
```

Visual C++ 6.0

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

Remarks

Persistent property.

See Also

3.1.14 IniFileName

Gets or sets INI file name to read/write settings.　If INI file name is either full path name or containing relative reference of folder name, INI file will be created and read in such location.　Otherwise, INI file will be created in Windows OS system folder.　For this reason, it is highly recommended using with folder name.　If current directory is the location, add "./" at the beginning.

Either "/" (slash) or "¥" (back slash) can be used for separator of folder name.

Syntax

Visual Basic 6.0

```
IniFileName As String
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Persistent property.

SavoyGem will reserve a bunch of INI sections.　If user application would share same INI file, please be careful of naming confliction.

See Also

3.1.15 IPAddress

Gets or sets the IP address of passive entity computer for HSMS connection. IPAddress property will be ignored if the Server property is set to true, because server listens incoming connection.

When connecting local computer (same computer), use “127.0.0.1” or “” (empty) string.

It is possible to use computer name instead of IP address.

Syntax

Visual Basic 6.0

```
IPAddress As String
```

Visual C++ 6.0

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

Example

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);
```

Remarks

Persistent property.

See Also

3.1.16 Log

Gets or sets whether logging is enabled. If this property is enabled, processing information will be written in log file. If this property is disabled, nothing will be written in log file.

Value	Description
True	Write to log file
False	Do not write log file

Syntax

Visual Basic 6.0

```
Log As Boolean
```

Visual C++ 6.0

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

Example

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);
```

Remarks

Persistent property.

See Also

3.1.17 LogBakCount

Gets or sets the number of back-up file for logging.　If actual file size of log file exceeded LogSize property, SavoyGem control will rename the file name and make a new empty log file.　If number of back-up file reached to the value of LogBakCount property, SavoyGem control will delete oldest back-up file.

Syntax

Visual Basic 6.0

```
LogBakCount As Integer
```

Visual C++ 6.0

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

Example

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);
```

Remarks

Persistent property.

See Also

3.1.18 LogCommunication

Get or sets whether logging for communication part is enabled.

Syntax

Visual Basic 6.0

```
LogCommunication As Boolean
```

Visual C++ 6.0

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

Example

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);
```

Remarks

Persistent property.

See Also

3.1.19 LogFileName

Get or sets the log file name. Log file will be created in current directory.

Syntax

Visual Basic 6.0

```
LogFileName As String
```

Visual C++ 6.0

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

Example

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);
```

Remarks

Persistent property.

Please don’t include file extension. SavoyGem control will append file extension “.log” automatically.

See Also

3.1.20 LogicalConnection

Gets or sets the logical connection. Logical connection is one of the followings:

Value	Description
0	General model.
1	GEM model

Syntax

Visual Basic 6.0

```
LogicalConnection As Integer
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.LogicalConnection = 1
```

Visual C++ 6.0

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

Remarks

Persistent property.

Since SavoyGem is a product for GEM communication, LogicalConnection property should always be 1.

See Also

3.1.21 LogSize

Gets or sets the log file size in kilobyte. If actual file size of log file exceeded LogSize property, SavoyGem control will rename the file name and make a new empty log file. If number of back-up file reached to the value of LogBakCount property, SavoyGem control will delete oldest back-up file.

Syntax

Visual Basic 6.0

```
LogSize As Long
```

Visual C++ 6.0

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

Example

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);
```

Remarks

Persistent property.

See Also

3.1.22 Msg

Gets or sets the message data of SECS-II. Message data format is in hexadecimal ASCII literal string.

Syntax

Visual Basic 6.0

```
Msg As String
```

Visual C++ 6.0

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

Example

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();
```

Remarks

See Also

3.1.23 MyPortNumber

Gets or sets local portnumber for TCP/IP connection.   If SavoyGem is running as active entity, this property should be 0.   Or connection will not be re-established until TCP/IP level time-out.

When SavoyGem is running as passive entity, MyPortNumber property indicates server port number for incoming client connection.

Since some port numbers are reserved by Windows OS, the number should be grater than 1024 in general. For example http server uses port number 80.

Syntax

Visual Basic 6.0

```
MyPortNumber As Long
```

Visual C++ 6.0

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

Example

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);
```

Remarks

Persistent property.

See Also

3.1.24 Node

Gets or sets the node for operation. Node consists of “/” (slash), node number, “[“ (left bracket) and “]” (right bracket). Node number is a numeric expression starting at 1. Index number starts at 0. If node is “” (empty), it means root.

Syntax

Visual Basic 6.0

```
Node As String
```

Visual C++ 6.0

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

Example

Make following message denoted by SML structure.

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

Since SavoyGem control may have some message structure, select root node to update whole structure.

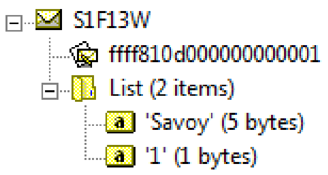
Visual Basic 6.0

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

Visual C++ 6.0

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

Running this code will create following message structure.



```
S1F13W
  ffff810d000000000001
  List (2 items)
    'Savoy' (5 bytes)
    '1' (1 bytes)
```

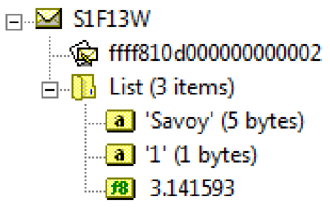
To add 3rd node, set Node property to “3”.

Visual Basic 6.0

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

Visual C++ 6.0

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


S1F13W
ffff810d000000000002
List (3 items)
'Savoy' (5 bytes)
'1' (1 bytes)
3.141593

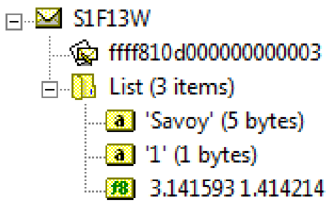
To convert 3rd node into array, set SML using same node type. This case, it is "f8" (8-byte floating point).

Visual Basic 6.0

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

Visual C++ 6.0

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


S1F13W
ffff810d000000000003
List (3 items)
'Savoy' (5 bytes)
'1' (1 bytes)
3.141593 1.414214

If 3rd node value is read using NodeValue property at this time, each member of array will be splitted with space character and "3.141593 1.414214" will be returned. If user wants to access specific member of array, use "[ ]" and index. Index starts at 0 such like C/C++/Java/C# language.

Visual Basic 6.0

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

Visual C++ 6.0

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

If "3[0]" was specified, "3.141593" will be returned. If "3[1]", "1.414214" will be returned.

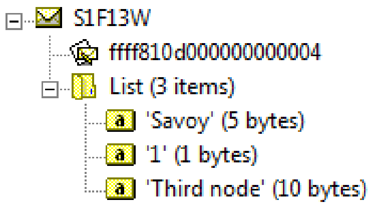
If user wants to change to different node type, set SML using different node type.

Visual Basic 6.0

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

Visual C++ 6.0

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


```
S1F13W
  ffff810d000000000004
  List (3 items)
    'Savoy' (5 bytes)
    '1' (1 bytes)
    'Third node' (10 bytes)
```

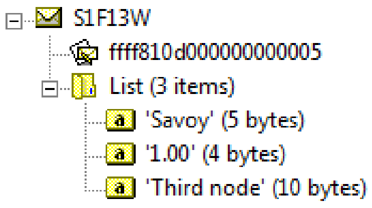
If user wants to concatenate literal strings, set SML using same node type. String is considered as an array of character.

Visual Basic 6.0

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

Visual C++ 6.0

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


```
S1F13W
  ffff810d000000000005
  List (3 items)
    'Savoy' (5 bytes)
    '1.00' (4 bytes)
    'Third node' (10 bytes)
```

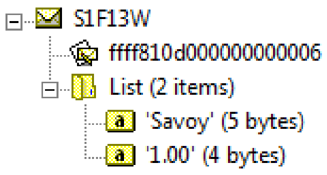
If empty SML was set, the node would be deleted.

Visual Basic 6.0

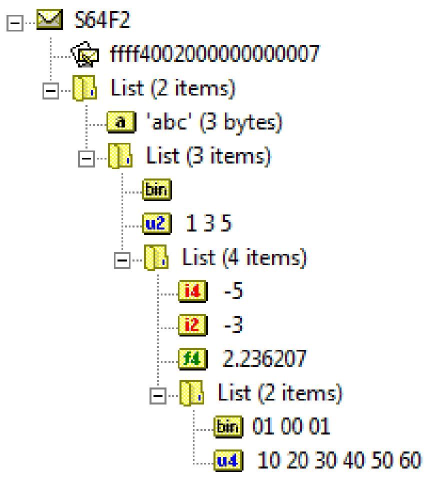
```
.Node = "3"
.SML = ""
```

Visual C++ 6.0

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


```
S1F13W
  ffff810d000000000006
  List (2 items)
    'Savoy' (5 bytes)
    '1.00' (4 bytes)
```

Using Node property, it is possible to extract value directly even from complicated message structure.


```
S64F2
  ffff4002000000000007
  List (2 items)
    'abc' (3 bytes)
    List (3 items)
      bin
      u2 1 3 5
      List (4 items)
        i4 -5
        i2 -3
        f4 2.236207
        List (2 items)
          bin 01 00 01
          u4 10 20 30 40 50 60
```

There is a 6-array node of u4 type. It is needed to specify node to extrace 4th value of it. Looking through from the root node, it would be 2nd node in list, 3rd in list, 4th in list, 2nd in list, and 4th in u4 type.

Visual Basic 6.0

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

Visual C++ 6.0

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

Setting Node property to "2/3/4/2[3]", NodeValue property returns "40".

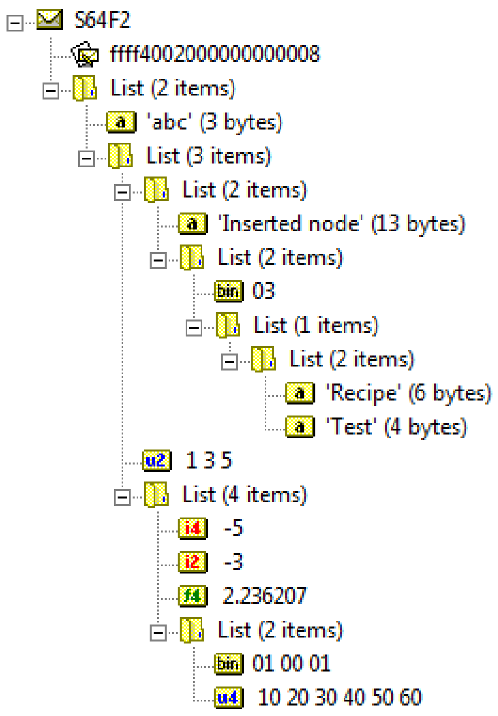
It is possible to set complicated SML structure to node.

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'>}}}}");
```



```
S64F2
    ffff4002000000000008
    List (2 items)
        'abc' (3 bytes)
        List (3 items)
            List (2 items)
                'Inserted node' (13 bytes)
                List (2 items)
                    03
                    List (1 items)
                        List (2 items)
                            'Recipe' (6 bytes)
                            'Test' (4 bytes)
            1 3 5
            List (4 items)
                -5
                -3
                2.236207
                List (2 items)
                    01 00 01
                    10 20 30 40 50 60
```

Remarks

Node resembles Windows folder structure. It may be helpful to replace “node” with “folder” in above description.

To create node, specify Node property and add SML property. To update whole message body, specify root node.

See Also

3.1.25 NodeCount

Gets or sets the number of sub items.   If node type is list, this property means the number of sub node. Otherwise, it means number of array.

Syntax

Visual Basic 6.0

```
NodeCount As Long
```

Visual C++ 6.0

```
long GetNodeCount()
```

Example

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());
```

Remarks

Read-only property.

See Also

3.1.26 NodeType

Gets or sets the node type. Node type is one of the followings:

Value	Enumeration	Description
1	SecsTypeList	List
2	SecsTypeBinary	Binary
3	SecsTypeBoolean	Boolean
4	SecsTypeAscii	ASCII string
5	SecsTypeJis	JIS 8 string
6	SecsTypeLong8	8-byte signed integer
7	SecsTypeChar	1-byte signed integer
8	SecsTypeShort	2-byte signed integer
9	SecsTypeLong	4-byte signed integer
10	SecsTypeDouble	8-bye floating point number
11	SecsTypeFloat	4-bye floating point number
12	SecsTypeDWord8	8-byte unsigned integer
13	SecsTypeByte	1-byte unsigned integer
14	SecsTypeWord	2-byte unsigned integer
15	SecsTypeDWord	4-byte unsigned integer
16	SecsTypeAscii2	2-byte ASCII string

Syntax

Visual Basic 6.0

```
NodeType As Integer
```

Visual C++ 6.0

```
short GetNodeType()
```

Example

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());
```

Remarks

Read-only property.

See Also

3.1.27 NodeValue

Gets or sets the node value. If node is numeric type, the number will be converted into decimal literal expression.

Syntax

Visual Basic 6.0

```
NodeValue As String
```

Visual C++ 6.0

```
CString GetNodeValue()
```

Example

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";
```

Remarks

Read-only property.

See Also

3.1.28 NodeValueHex

Gets or sets the node value in hexadecimal expression.

Syntax

Visual Basic 6.0

```
NodeValueHex As String
```

Visual C++ 6.0

```
CString GetNodeValueHex()
```

Example

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";
```

Remarks

Read-only property.

See Also

3.1.29 OfflineRequest

Gets or sets the setting wether SavoyGem will accept S1F15 request off-line (ROFL). If OfflineRequest property is true, SavoyGem will accept off-line request and reply S1F16 off-line acknowledgement (OFLA) with OFLACK = 0. If it is false, SavoyGem reject off-line request and reply with OFLACK = 1.

Value	Description
False	Don’t accept S1F15 request off-line (ROFL).
True	Accept S1F15 request off-line (ROFL).

Syntax

Visual Basic 6.0

```
OfflineRequest As Boolean
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.OfflineRequest = True
```

Visual C++ 6.0

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

Remarks

See Also

3.1.30 OnlineRequest

Gets or sets the setting wether SavoyGem will accept S1F17 request on-line (RONL). If OnlineRequest property is true, SavoyGem will accept on-line request and reply S1F16 on-line acknowledgement (ONLA) with ONLACK = 0. If it is false, SavoyGem reject on-line request and reply with ONLACK = 1.

Value	Description
False	Don’t accept S1F17 request on-line (RONL).
True	Accept S1F17 request on-line (RONL).

Syntax

Visual Basic 6.0

```
OnlineRequest As Boolean
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.OnlineRequest = True
```

Visual C++ 6.0

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

Remarks

See Also

3.1.31 Reply

Gets or sets the work space mode. If this property is true, it means buffer 1 is selected. If this property is false, it means buffer 0 is selected.

Value	Description
False	Select buffer 0
True	Select buffer 1

Syntax

Visual Basic 6.0

```
Reply As Boolean
```

Visual C++ 6.0

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

Example

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();
```

Remarks

See Also

WorkSpace property

3.1.32 PhysicalConnection

Gets or sets the physical connection. If PhysicalConnection property is true, physical connection model will be enabled. If false, it will be disabled.

Value	Description
False	Physical connection is disabled
True	Physical connection is enabled

Syntax

Visual Basic 6.0

```
PhysicalConnection As Boolean
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

Visual C++ 6.0

Remarks

Changing PortNumber property to true will open communication port, and then transmission becomes available. Note that communication port actually is not opened immediately, it is slightly later. Therefore, following code doesn't tell wether communication port was opened or not.

Visual Basic 6.0

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

Visual C++ 6.0

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

See Also

3.1.33 PortNumber

Gets or sets the port number for TCP/IP connection.

Since some port numbers are reserved by Windows OS, the number should be grater than 1024 in general. For example http server uses port number 80.

Syntax

Visual Basic 6.0

```
PortNumber As Long
```

Visual C++ 6.0

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

Example

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);
```

Remarks

Persistent property.

Since possive entity doesn’t attempt to connect to other computer, PortNumber property is not necessary (ignored).   For active entity, specify server port number to connect.

See Also

3.1.34 PType

Gets or sets the presentation type in SECS-II header.

For HSMS data message following header structure is used.

Byte	Description	
1	Session ID	
2		
3	W	Stream
4	Function	
5	P type	
6	S type	
7	System bytes	
8		
9		
10		

For HSMS control message following header structure is used.

Byte	Description
1	Session ID
2	
3	
4	
5	P type
6	S type
7	System bytes
8	
9	
10	

Syntax

Visual Basic 6.0

```
PType As Integer
```

Visual C++ 6.0

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

Example

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!");
```

Remarks

This property should always be 0, since SEMI E37 defines only SECS-II type at the moment.

See Also

3.1.35 Server

Gets or sets the entity type. If Server property is true, SavoyGem control will run as server. If Server property is false, SavoyGem control will run as client.

Value	Description
False	Active entity (client)
True	Passive entity (server)

Syntax

Visual Basic 6.0

```
Server As Boolean
```

Visual C++ 6.0

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

Example

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);
```

Remarks

Persistent property.

See Also

3.1.36 SessionID

Gets or sets the session ID for HSMS. Session ID is first 16 bits of SECS-II header.

For HSMS data message following header structure is used.

Byte	Description
1	Session ID
2	
3	W Stream
4	Function
5	P type
6	S type
7	System bytes
8	
9	
10	

For HSMS control message following header structure is used.

Byte	Description
1	Session ID
2	
3	
4	
5	P type
6	S type
7	System bytes
8	
9	
10	

Syntax

Visual Basic 6.0

```
SessionID As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.SessionID = &HFFFF
```

Visual C++ 6.0

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

Remarks

Device ID and session ID are almost same, but device ID is 15-bit, where session ID is 16-bit.

See Also

3.1.37 SML

Gets or sets the message in SML string. Readin SML property will convert message structure into SML literal string. It is possible to insert CR (carriage return), LF (line feed), space code, tab code in SML string to set it in SML property. They would be ignored except in some context.

Syntax

Visual Basic 6.0

```
SML As String
```

Visual C++ 6.0

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

Example

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();
```

Remarks

The grammar of the string to set SML property is as follows;

Common Notice

White space (space, tab, carriage return and line feed) is treated as only a separator. It is possible use them to improve readability of source code. But it is treated as character in comment or string expression context.

From aster “*” to the end of line is treated as comment except aster in string text.

Integer consists of numeric character “0” through “9” and minus “-“ flag. To write in hexadecimal expression, put “0x” in front of the expression. In this case, user can also use “a” through “f” and “A” through “F”. For decimal part of the number, it is possible to omit first character “0” such like “.9” as “0.9”. It also is possible to use exponential expression. There are reserved words like “true” (=1) and “false” (=0).

String is surrounded by single-quotation marks “’”. It is not allowed to contain line-break and single-quotation mark itself. So if it would need to fill such kind of characters in string, use hexadecimal expression like “0x0a”.

Bold letter portion in explanation means to describe character itself. These characters may be OK in either uppercase or lowercase letter. Refer to each explanation for an italic character. Moreover, the portion surrounded by brackets “[ ]” can be omitted.

Grammar

```
[sxxfyy[w]] Body
```

Item	Description
xx	Stream number.　Don’t insert space code between “s” and “f”.
yy	Function number.　Don’t insert space code between “f” and “w”.
w	Wait bit.　Append “w” if needed.
Body	Message body.

In order to recognize stream, function, and wait-bit as 1 lump, don't put neither space nor line-break character among them.　All of streams and functions can be omitted and only message body can also be described.

Message body

Message body is hierarchy structure.

List

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

Item	Description
NumOfItem	Number of list.　This is only for compatibility purpose with SECSIM.　SavoyGem control would ignore this number.
Body	Message body.　It is possible to insert other items here.

ASCII string

```
<a [Strings]>
```

Item	Description
Strings	ASCII literal string.

Long string can be splitted into short strings.　Moreover, it is possible to use numeric character code as follows:

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

This is same as follows:

```
<a ‘ABCDEF0123456789’>
```

2-byte string

2-byte string is treated as same kind of string as ASCII string. But no one saw this type in SEMI Standards specification.

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

Item	Description
Strings	2-byte character string for far east complicated language. This version of Savoy control can handle only DBCS (Double Byte Character Set).

JIS8 string

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

JIS8 string is treated as same kind of string as ASCII string. But no one saw this type in SEMI Standards specification.

Item	Description
Strings	JIS-8 string of text for Japanese 'katakana'.

Long string can be splitted into short strings. Moreover, it is possible to use numeric character code as follows:

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

This is same as follows:

```
<j ‘ABCDEF0123456789’>
```

Integer

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

Item	Description
Numbers	Integer. It must be one of followings.

Type	Description
i1	8-bit signed integer
i2	16-bit signed integer
i4	32-bit signed integer
i8	64-bit signed integer
u1	8-bit unsigned integer
u2	16-bit unsigned integer
u4	32-bit unsigned integer
u8	64-bit unsigned integer

It is possible to enumerate multiple numbers and it means array as follows:

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

Current version of SavoyGem cannot handle very huge number in i8 and u8.

Floating point number

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

Integer	Description
FNumbers	Floating point number. It is one of followings.

Type	Description
f4	32-bit floating point number
f8	64-bit floating point number

For example,

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

Binary

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

Item	Description
Numbers	Number.

For example,

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

Boolean

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

Item	Description
Numbers	Boolean (true or false) number.

For example,

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

See Also

3.1.38 Stream

Gets or sets the stream in SECS-II header.

For HSMS data message following header structure is used.

Byte	Description
1	Session ID
2	
3	
4	
5	P type
6	S type
7	System bytes
8	
9	
10	

Syntax

Visual Basic 6.0

```
Stream As Integer
```

Visual C++ 6.0

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

Example

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
    ...
```

Remarks

See Also

3.1.39 SType

Gets or sets the session type in SECS-II header.

Value	Description
0	SECS-II data message
1	Select.Req
2	Select.Rsp
3	Deselect.Req
4	Deselect.Rsp
5	LinkTest.Req
6	LinkTest.Rsp
7	Reject.Req
8	(not used)
9	Separate.Req
10	(not used)
11-127	
128-255	

For HSMS data message following header structure is used.

Byte	Description	
1	Session ID	
2		
3	W	Stream
4	Function	
5	P type	
6	S type	
7	System bytes	
8		
9		
10		

For HSMS control message following header structure is used.

Byte	Description
1	Session ID
2	
3	
4	
5	P type
6	S type
7	System bytes
8	
9	
10	

Syntax

Visual Basic 6.0

```
SType As Integer
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
```

Visual C++ 6.0

```
```

Remarks

See Also

3.1.40 SystemBytes

Gets or sets the system bytes in SECS-II header.

For HSMS data message following header structure is used.

Byte	Description	
1	Session ID	
2		
3	W	Stream
4	Function	
5	P type	
6	S type	
7	System bytes	
8		
9		
10		

For HSMS control message following header structure is used.

Byte	Description
1	Session ID
2	
3	
4	
5	P type
6	S type
7	System bytes
8	
9	
10	

Syntax

Visual Basic 6.0

```
SystemBytes As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.1.41 T1

Gets or sets the T1 time out for SECS-I in 1/10 seconds. Default value is 0.5 seconds.

Syntax

Visual Basic 6.0

```
T1 As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Persistent property.

This property is not used at the moment.

See Also

3.1.42 T2

Gets or sets the T2 time out for SECS-I in 1/10 seconds. Default value is 10 seconds.

Syntax

Visual Basic 6.0

```
T2 As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
Dim lT2 As Long
lT2 = .T2
```

Visual C++ 6.0

```
long lT2 = m_ctrl.GetT2();
```

Remarks

Persistent property.

This property is not used at the moment.

See Also

3.1.43 T3

Gets or sets the T3 time out in seconds. Default value is 45 seconds.

Syntax

Visual Basic 6.0

```
T3 As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
Dim lT3 As Long
lT3 = .T3
```

Visual C++ 6.0

```
long lT3 = m_ctrl.GetT3();
```

Remarks

Persistent property.

T3 timeout also is known as transaction timeout. This timer is between primary message and reply message. If it pasts more than T3 timer, it will fall into T3 timeout and equipment will send S9F9 transaction timeout (TIN).

See Also

3.1.44 T4

Gets or sets the T4 time out for SECS-I in seconds. Default value is 45 seconds.

Syntax

Visual Basic 6.0

```
T4 As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Persistent property.

This property is not used at the moment.

See Also

3.1.45 T5

Gets or sets the T5 time out for HSMS in seconds. Default value is 10 seconds.

Syntax

Visual Basic 6.0

```
T5 As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Persistent property.

T5 timeout also known as connection separation timeout. T5 property will affect only if active entity. If connection was failed or disconnected, active entity should wait more than this timer value until attempting next connection..

See Also

3.1.46 T6

Gets or sets the T5 time out for HSMS in seconds.   Default value is 5 seconds.

Syntax

Visual Basic 6.0

```
T6 As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Persistent property.

T6 timeout also is known as control transaction timeout.   This timer is between request message and response message on HSMS control message (S type is not 0).   T6 and T3 is resemble, where T3 timeout is for SECS-II data message.

T6 timeout actually is happen, when following timer became timeout.

S type	Description
1	After sending Select.req until receiving Select.rsp
3	After sending Deselect.req until receiving Deselect.rsp
5	After sending Linktest.req until receiving Linktest.rsp

See Also

3.1.47 T7

Gets or sets the T7 time out for HSMS in seconds. Default value is 10 seconds.

Syntax

Visual Basic 6.0

```
T7 As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Persistent property.

T7 timeout also is known as NOT SELECTED timeout. Although connection was established on TCP/IP level, it will cause T7 timeout if not selected. Also when received Deselect.req (shift from Selected state to Not Selected state), connection will disconnected if not selected.

See Also

3.1.48 T8

Gets or sets the T8 time out for HSMS in seconds. Default value is 5 seconds.

Syntax

Visual Basic 6.0

```
T8 As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Persistent property.

T8 timeout also is known as network inter-character timeout. Even HSMS connection was not disconnected, if data flow is stopped during transmission, receiver software cannot discriminate if it is a sequel of message or not. If it pasts more than T8 timer, it is considered as “communication failure” and eventually connection will be disconnected.

T8 timeout resembles T1 timeout on SECS-I.

See Also

3.1.49 Verification

Gets and sets the verification result of message structure. It should be one of followings:

Value	Enumeration	Description
0	VerificationCorrect	No problem.
1	VerificationUserDefined	User defined message.
2	VerificationIncorrect	Incorrect message structure.
3	VerificationIncorrectAndReply	Incorrect message structure and need to reply.
4	VerificationNoWBit	No wait bit where it supposedly has it.
5	VerificationWBit	Wait bit where it supposedly should not have it.
6	VerificationWrongDirection	The direction of message is wrong.
7	VerificationUnrecognizedStream	Unrecognized stream.
8	VerificationUnrecognizedFunction	Unrecognized function.

Syntax

Visual Basic 6.0

```
Verification As Integer
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

Visual C++ 6.0

Remarks

If SavoyGem received message, message structure would be verified. The result will be set to Verification property, and cause Received event.

See Also

3.1.50 VIDCount

Gets or sets the number of variable ID. If this value is 0, none is registered.

Syntax

Visual Basic 6.0

```
VIDCount As Long
```

Visual C++ 6.0

```
long GetVIDCount()
```

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Read-only property.

Since VIDCount property returns the number of registered VID, available index range is between 0 and (VIDCount – 1). Use ToVID method to convert index into VID.

See Also

3.1.51 ViewStyle

Gets or sets the view style of SavoyGem control.　It should be one of the followins:

Style	Value	Description
RedrawNone	0	Don't display.
RedrawHsms	1	Display only HSMS physical connection.
RedrawGem	2	Display only GEM logical connection.
RedrawNormal	3	Display all.

Syntax

Visual Basic 6.0

```
ViewStyle As Integer
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.ViewStyle = 3
```

Visual C++ 6.0

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

Remarks

Persistent property.

See Also

3.1.52 Wbit

Gets or sets the wait bit in SECS-II header.

For SECS-I following header structure is used.

Value	Description
False	No reply message expected.
True	Reply message expected.

For HSMS data message following header structure is used.

Byte	Description	
1	Session ID	
2		
3	W	Stream
4	Function	
5	P type	
6	S type	
7	System bytes	
8		
9		
10		

Syntax

Visual Basic 6.0

```
Wbit As Boolean
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
If .Wbit Then
    ‘ Reply
```

Visual C++ 6.0

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

Remarks

See Also

3.1.53 WorkSpace

Gets or sets the work space for SECS-II message. WorkSpace is a work area for message manipulation. There are 3 workspaces in SavoyGem control. Each workspace has 2 buffers; one for primary message (Reply = false), and one for reply message (Reply = true). So it is possible to manipulate up to 6 different messages. However, mostly WorkSpace 0 would be used.

WorkSpace	Reply	Description
0	False	For send and received message
0	True	For reply message of received message
1	False	For sent message
1	True	For user's convenience
2	False	For user's convenience
2	True	For user's convenience

When message was delivered and notified through Received event, the message would be put in WorkSpace 0 and Reply = false. Please note that all the incoming messages will be delivered in that area even received message was reply message. At the time when event was notified, WorkSpace 0 was selected. But once the processing is returned from event handler function to SavoyGem control, previously selected WorkSpace and Reply will be set.

For incoming primary message, "suggested reply message" would be set in Reply = true buffer. If user wants to send as is, simply change Reply property to true and call Send method. Of course, user can modify the content of message.

If user wants to send primary message, use WorkSpace = 0 and Reply = false. Once transmission was successfully done, it would be notified through Sent event, WorkSpace 1 and Reply = false would be selected automatically.

Syntax

Visual Basic 6.0

```
WorkSpace As Integer
```

Visual C++ 6.0

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

Example

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();
```

Remarks

See Also

3.2 Array Properties

3.2.1 ALCD

Gets or sets the alarm code for specified alarm ID.

ALID	Description
0	Not used
1	Personal safety
2	Equipment safety
3	Parameter control warning
4	Parameter control error
5	Irrecoverable error
6	Equipment status warning
7	Attention flag
8	Data integrity
>8	Other categories
9-63	Reserved

Syntax

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)
```

Argument	Description
lALID	Alarm ID

Example

If ALID was not registered, user cannot set value. If user wants to get the list of registered ALID, take following steps.

First, read ALIDCount property value.

Visual Basic 6.0

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

Visual C++ 6.0

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

Since ALIDCount property returns the number of registered ALID, the index should be between 0 and (ALIDCount - 1). Convert index into ALID using ToALID method.

Visual Basic 6.0

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

Visual C++ 6.0

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

Once ALID were determined, it is possible to access ALCD property and ALTX property.

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);
```

Enumerate all the ALIDs using “for” loop. Following is the entire source code.

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);
}
```

Remarks

When value was set in ALCD property, only lower 7-bit would be recorded. The highest bit of ALCD is used for "alarm set" or "alarm cleared". This bit can be changed by InvokeAlarm method (use argument).

See Also

3.2.2 ALIDEnable

Gets or sets whether the alarm is enabled or disabled for specified alarm ID.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

```
short GetALIDEnable(long lALID)
void SetALIDEnable(long lALID, short nNewValue)
```

Argument	Description
lALID	Alarm ID

Example

If ALID was not registered, user cannot set value. If user wants to get the list of registered ALID, take following steps.

First, read ALIDCount property value.

Visual Basic 6.0

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

Visual C++ 6.0

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

Since ALIDCount property returns the number of registered ALID, the index should be between 0 and (ALIDCount - 1). Convert index into ALID using ToALID method.

Visual Basic 6.0

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

Visual C++ 6.0

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

Once ALID were determined, it is possible to access ALCD property and ALTX property.

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);
```

Enumerate all the ALIDs using “for” loop.   Following is the entire source code.

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);
}
```

Remarks

See Also

3.2.3 ALIDSet

Gets or sets whether the alarm is set or reset for specified alarm ID.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

```
short GetALIDSet(long lALID)
void SetALIDSet(long lALID, short nNewValue)
```

Argument	Description
lALID	Alarm ID

Example

If ALID was not registered, user cannot set value. If user wants to get the list of registered ALID, take following steps.

First, read ALIDCount property value.

Visual Basic 6.0

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

Visual C++ 6.0

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

Since ALIDCount property returns the number of registered ALID, the index should be between 0 and (ALIDCount - 1). Convert index into ALID using ToALID method.

Visual Basic 6.0

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

Visual C++ 6.0

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

Once ALID were determined, it is possible to access ALCD property and ALTX property.

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);
```

Enumerate all the ALIDs using “for” loop.   Following is the entire source code.

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);
}
```

Remarks

See Also

3.2.4 ALTX

Gets and sets the alarm text for specified alarm ID.

Syntax

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)
```

Argument	Description
lALID	Alarm ID

Example

If ALID was not registered, user cannot set value. If user wants to get the list of registered ALID, take following steps.

First, read ALIDCount property value.

Visual Basic 6.0

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

Visual C++ 6.0

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

Since ALIDCount property returns the number of registered ALID, the index should be between 0 and (ALIDCount - 1). Convert index into ALID using ToALID method.

Visual Basic 6.0

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

Visual C++ 6.0

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

Once ALID were determined, it is possible to access ALCD property and ALTX property.

Visual Basic 6.0

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

Visual C++ 6.0

```
int nALCD = m_ctrl.GetAlcd(lALID);
```

Enumerate all the ALIDs using “for” loop.   Following is the entire source code.

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)
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);
    String strALTX = m_ctrl.GetAltx(lALID);
}
```

Remarks

The length of ALTX is limited up 40 characters by SEMI Standards, however, SavoyGem doesn’t have such regulation.   User can send any length of alarm text.

See Also

3.2.5 CEIDDescription

Gets or sets the description for specified collection event ID.

Syntax

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)
```

Argument	Description
lCEID	Collection event ID

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.2.6 CEIDEnable

Gets or sets whether the CEID is enabled or disabled for specified collection event ID.

Syntax

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)
```

Argument	Description
lCEID	Collection event ID

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.2.7 VIDDefault

Gets and sets the default value for specified variable ID.

Syntax

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)
```

Argument	Description
lVID	Variable ID

Example

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);
```

Remarks

ECDEF in S2F30 equipment constant namelist (ECN).

See Also

3.2.8 VIDDescription

Gets and sets the description for specified variable ID.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

```
CString GetVIDDescription(long lVID)
void SetVIDDescription(long lVID, LPCTSTR lpszNewValue)
```

Argument	Description
lVID	Variable ID

Example

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);
```

Remarks

VIDDescription property is treated differently by setting. If variable type is system variable (SVID), it will be treated as SVNAME in S1F12 status variable namelist reply (SVNRR).

If variable type is equipment constant (ECID), it will be treated as ECNAME in S2F30 equipment constant namelist (ECN).

If variable type is data variable (DVID), it might be treated as DVNAME. Unfortunately, GEM define neither S6F4 nor S6F8. SavoyGem control doesn’t process VIDDescription property automatically. However, user can send S6F4 and S6F8 using VIDDescription property.

See Also

3.2.9 VIDMax

Gets and sets the maximum value for specified variable ID.

Syntax

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)
```

Argument	Description
lVID	Variable ID

Example

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);
```

Remarks

ECMAX in S2F30 equipment constant namelist (ECN).

See Also

3.2.10 VIDMin

Gets and sets the minimum value for specified variable ID.

Syntax

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)
```

Argument	Description
lVID	Variable ID

Example

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);
```

Remarks

ECMIN in S2F30 equipment constant namelist (ECN).

See Also

3.2.11 VIDNodeType

Gets and sets the node type for specified variable ID.

Value	Enumeration	Description
1	SecsTypeList	List
2	SecsTypeBinary	Binary
3	SecsTypeBoolean	Boolean
4	SecsTypeAscii	ASCII string
5	SecsTypeJis	JIS 8 string
6	SecsTypeLong8	8-byte signed integer
7	SecsTypeChar	1-byte signed integer
8	SecsTypeShort	2-byte signed integer
9	SecsTypeLong	4-byte signed integer
10	SecsTypeDouble	8-bye floating point number
11	SecsTypeFloat	4-bye floating point number
12	SecsTypeDWord8	8-byte unsigned integer
13	SecsTypeByte	1-byte unsigned integer
14	SecsTypeWord	2-byte unsigned integer
15	SecsTypeDWord	4-byte unsigned integer
16	SecsTypeAscii2	2-byte ASCII string

Syntax

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)
```

Argument	Description
lVID	Variable ID

Example

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);
```

Remarks

If the value of VIDNodeType property is 0, it means “invalid type”.

See Also

3.2.12 VIDRawValue

Gets and sets the raw value for specified variable ID. Raw value contains SECS-II message structure.

Syntax

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)
```

Argument	Description
lVID	Variable ID

Example

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);
```

Remarks

If SML string was set in VIDType property, SavoyGem would make message following report setting and send S6F11 event report send (ERS). At this time the value of VIDRawValue property will be used for each VID. It is not allowed to use stream and function number in SML string.

Since SavoyGem control doesn’t check syntax and type of VID, user should set correct SML string. If user wants to set value by checking VIDNodeType property, use VIDValue property.

See Also

3.2.13 VIDType

Gets and sets the variable type for specified variable ID.

Type	Description
1	ECID
2	SVID
4	DVID

It is not allowed to use other numbers.

Syntax

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)
```

Argument	Description
lVID	Variable ID

Example

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);
```

Remarks

See Also

3.2.14 VIDUnit

Gets and sets the unit for specified variable ID.

Syntax

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)
```

Argument	Description
lVID	Variable ID

Example

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);
```

Remarks

UNITS in S2F30 equipment constant namelist (ECN).

See Also

3.2.15 VIDValue

Gets and sets the value for specified variable ID.

Syntax

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)
```

Argument	Description
lVID	Variable ID

Example

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);
```

Remarks

Following the type definition of VIDNodeType property, SML string will be generated in VIDRawValue property.

See Also

3.3 Methods

3.3.1 AboutBox

Opens version information dialog box on the screen.

Syntax

Visual Basic 6.0

```
Sub AboutBox()
```

Visual C++ 6.0

```
void AboutBox()
```

Return Value

None.

Example

Visual Basic 6.0

```
.AboutBox
```

Visual C++ 6.0

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

Remarks

See Also

3.3.2 DefProc

Calls default procedure when SavoyGem control received message.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

```
BOOL DefProc()
```

Return Value

Return true if processed, and false if not processed.

Example

Visual Basic 6.0

```
.DefProc
```

Visual C++ 6.0

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

Remarks

See Also

3.3.3 InvokeAlarm

Lets SavoyGem control attempt to send alarm event. Actually, SavoyGem will send S5F1 alarm report send (ARS). If specified alarm ID was disabled, alarm event will not be sent.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lALID	Alarm ID. ALID should be registered previously.
sALCD	Alarm code. Only highest bit will be used.

Return Value

Return true if alarm was sent, and false if not sent.

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Whichever number was specified in sALCD, lower 7-bit would be ignored and replaced by pre-registered ALCD. Since ALCD is a binary type, it is 8-bit. Therefore, only 8th bit in sALCD is valid. If this bit is 1, it means “alarm set”. If this bit is 0, it means “alarm cleared”.

To send “alarm cleared” message, “alarm set” has to be made. If “alarm set” was sent, SavoyGem control would set “not cleared” flag for such ALID internally. If “not cleared” flag was not set, it is not possible to send “alarm cleared” message. If “alarm cleared” message is sent, “not cleared” flag would be reset.

Under SEMI Standards specification, only “on” and “off” information is recorded for same ALID, and “how many” number isnot recorded. Even if “alarm set” happened more than once, just one “alarm cleared” would reset “alarm cleared” flag.

Since “alarm cleared” flag will be recorded in SavoyGem data file, it is possible to retrieve previous setting after rebooting application.

See Also

3.3.4 InvokeEvent

Lets SavoyGem control attempt to send collection event. Actually, SavoyGem control will send S6F11 event report send (ERS). If specified collection event ID was disabled, collection event will not be sent.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lCEID	Collection event ID

Return Value

Return true if event was sent, and false if not.

Example

Visual Basic 6.0

```
.InvokeEvent 1100
```

Visual C++ 6.0

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

Remarks

If report was linked to event, such report would be generated automatically.

See Also

3.3.5 IsValidVID

Checks whether specified variable ID is valid. If variable ID was not registered, this method returns false.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lVID	Variable ID

Return Value

Return true if VID was registered, and false if not registered.

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.3.6 LoadData

Loads .data from SavoyGem data file.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

```
BOOL LoadData()
```

Return Value

Return true if loading was successful, and false if failed.

Example

Visual Basic 6.0

```
.DataFileName = “.¥SavoyGem.bop”
.LoadData
```

Visual C++ 6.0

```
m_ctrl.SetDataFileName(“.¥SavoyGem.bop”);
m_ctrl.LoadData();
```

Remarks

See Also

DataFileName property

3.3.7 LoadIniFile

Loads settings from INI file and initialize properties. If loading was failed, values in persistent resource will be set.

LoadIniFile method probably is called at the beginning of application, since it retrieves saved parameters by Setup method.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

```
BOOL LoadIniFile()
```

Return Value

Return true if loading was successful. Otherwise return false. If false was returned, IniFileName property or IniSection property might be incorrect.

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.3.8 RegisterALID

Registers alarm ID.

Syntax

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)
```

Argument	Description
lALID	Alarm ID
sALCD	Alarm code
lpszALTX	Alarm text

Return Value

Return true if registration was successful, and false if failed.

Example

Visual Basic 6.0

```
.RegisterALID 5000, 1, “”
```

Visual C++ 6.0

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

Remarks

It is not recommended to use RegisterALID method at the beginning of application, because that will affect to the setting of S5F3 enable/disable alarm send (EAS). Please register previously and use LoadData method to retrieve previous setting.

See Also

3.3.9 RegisterCEID

Registers collection event ID.

Syntax

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	Event ID
sPredefinedCEID	Predefined CEID
lpszDescription	CEID description

Return Value

Return true if registration was successful, and false if failed.

Example

Visual Basic 6.0

```
.RegisterCEID 253, 1, “”
```

Visual C++ 6.0

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

Remarks

It is not recommended to use RegisterCEID method at the beginning of application.　Please register previously and use LoadData method to retrieve previous setting.

See Also

3.3.10 RegisterVID

Registers variable ID.

Syntax

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)

Argument	Description
lVID	Variable ID
sType	Variable type (one of ECID, SVID or DVID)
sNodeType	SECS-II node type
lpszMin	ECMIN (minimum value)
lpszMax	ECMAX (maximum value)
lpszDefault	ECDEF (default value)
lpszUnit	UNITS (unit)
lpszDescription	ECNAME (description)

Return Value

Return true if registration was successful, and false if failed.

Example

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”);

Remarks

It is not recommended to use RegisterVID method at the beginning of application. Please register previously and use LoadData method to retrieve previous setting.

See Also

3.3.11 SaveData

Saves data into SavoyGem data file.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

```
BOOL SaveData()
```

Return Value

Return true if saving was successful, and false if failed.

Example

Visual Basic 6.0

```
.SaveData
```

Visual C++ 6.0

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

Remarks

See Also

3.3.12 Send

Send message specified by WorkSpace property and Reply property.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

```
BOOL Send()
```

Return Value

Return true if transmission was successful. Otherwise return false.

Example

Visual Basic 6.0

```
.Send
```

Visual C++ 6.0

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

Remarks

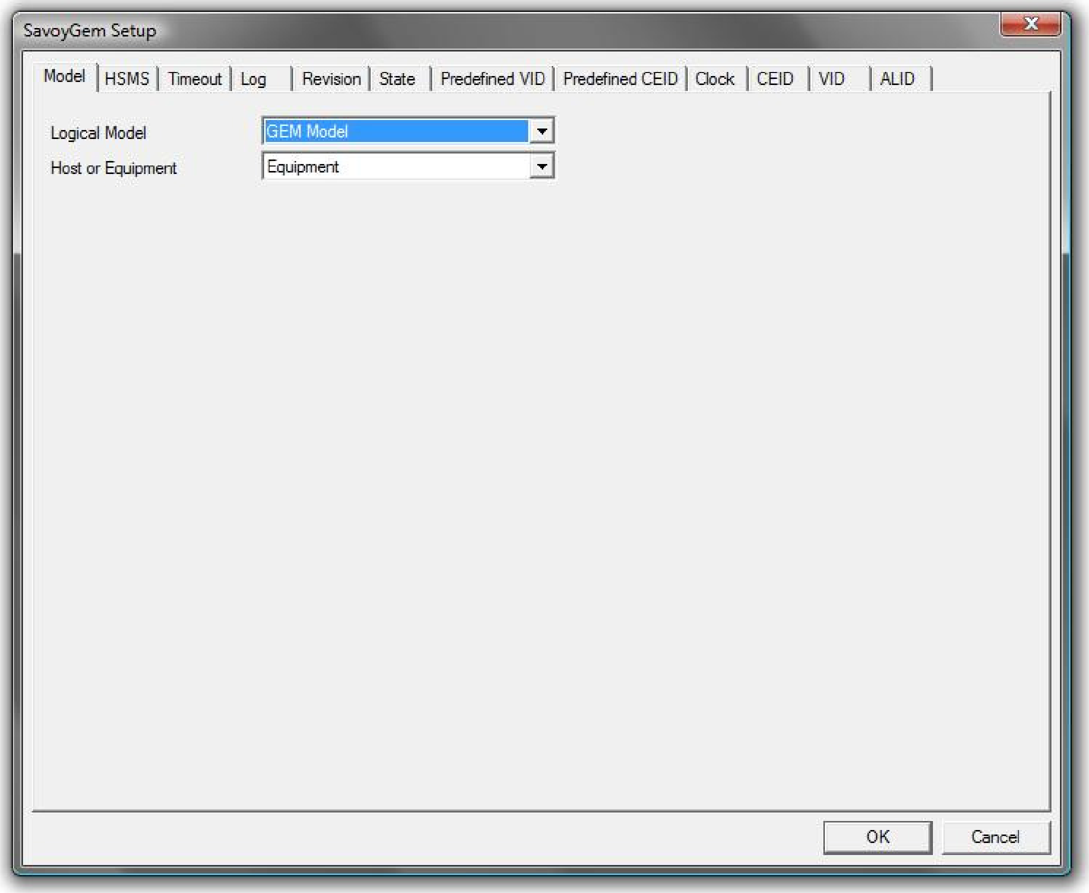
See Also

WorkSpace property, Reply property

3.3.13 Setup

Opens setup dialog box on the screen. If user modified parameter and press OK button, data will be written in INI file.

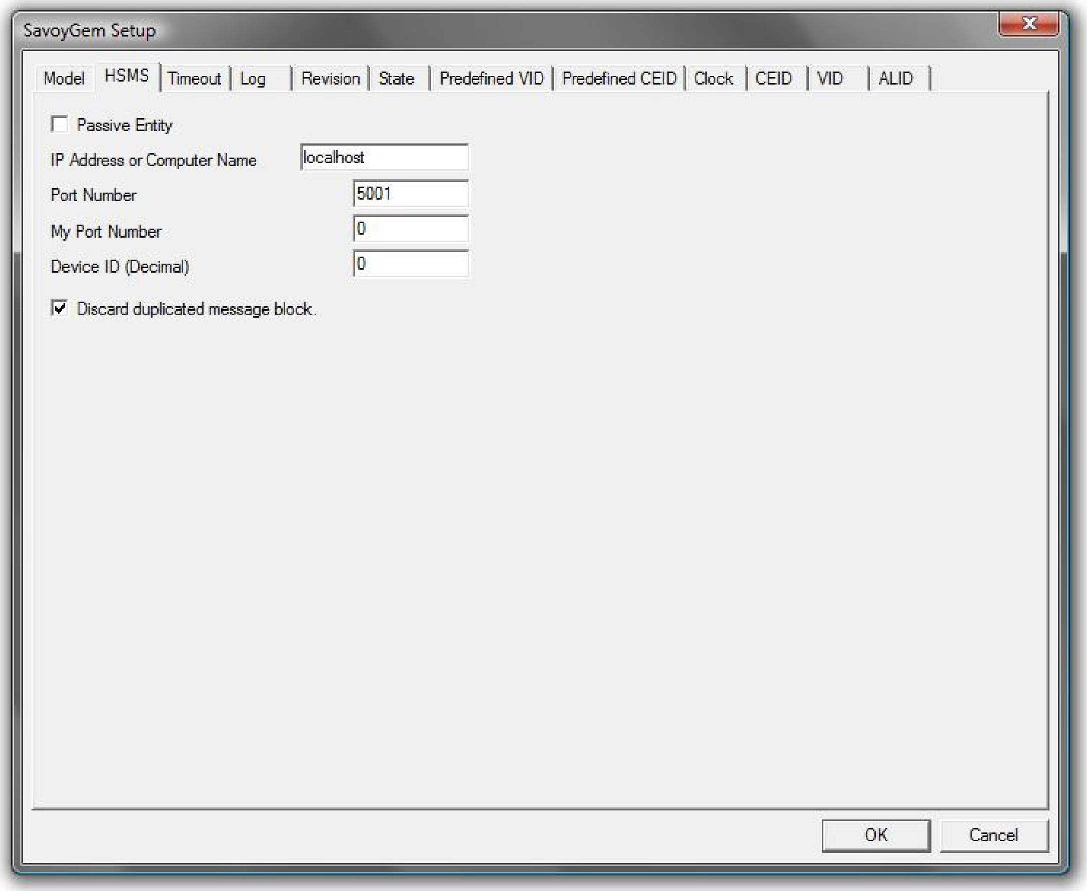
Model Tab



Logical Model: GEM Model
Host or Equipment: Equipment

Item	Description
Logical Model	Choose logical connection model. Please select GEM Model at any time.
Host or Equipment	Choose role. Please select Equipment at any time.

HSMS Tab



☐ Passive Entity

IP Address or Computer Name: localhost

Port Number: 5001

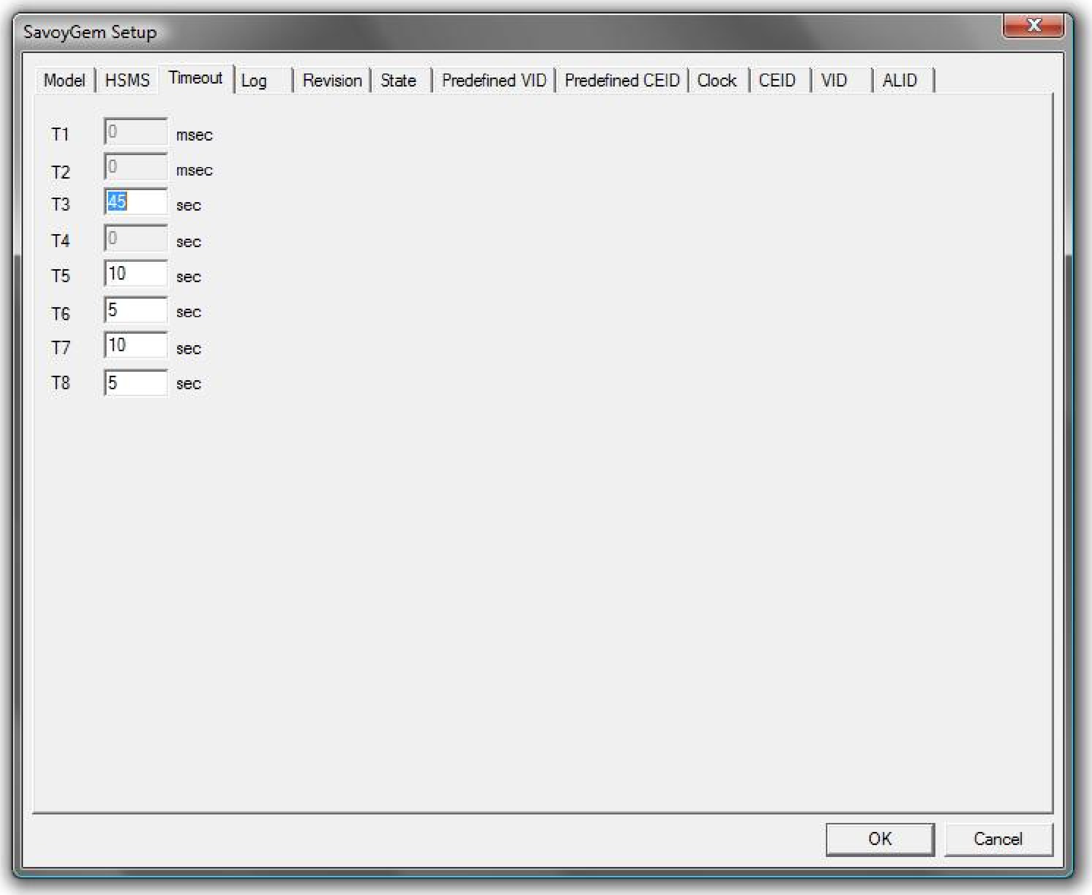
My Port Number: 0

Device ID (Decimal): 0

☑ Discard duplicated message block.

Item	Description
Passive Entity	True if passive entity. False if active entity.
IP Address or Computer Name	IP address or computer name of another computer to connect.
Port Number	Port number of another computer to connect.
My Port Number	Local port number. It is highly recommended to specify 0 for active entity.
Device ID (Decimal)	Device ID (session ID)
Discard duplicated block	Whether discard duplication block.

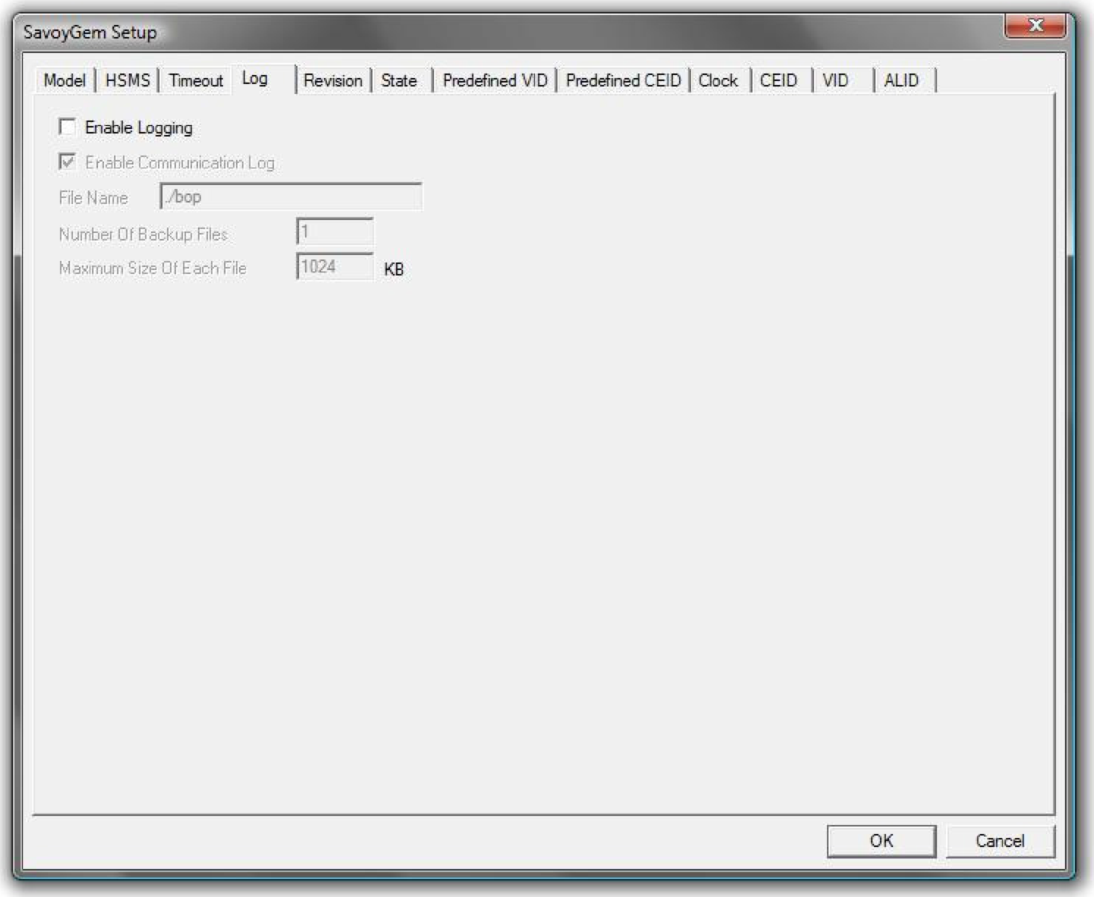
Timeout Tab



T1 0 msec
T2 0 msec
T3 45 sec
T4 0 sec
T5 10 sec
T6 5 sec
T7 10 sec
T8 5 sec

Item	Description
T3	T3 timer
T5	T5 timer
T6	T6 timer
T7	T7 timer
T8	T8 timer

Log Tab



☐ Enable Logging

☑ Enable Communication Log

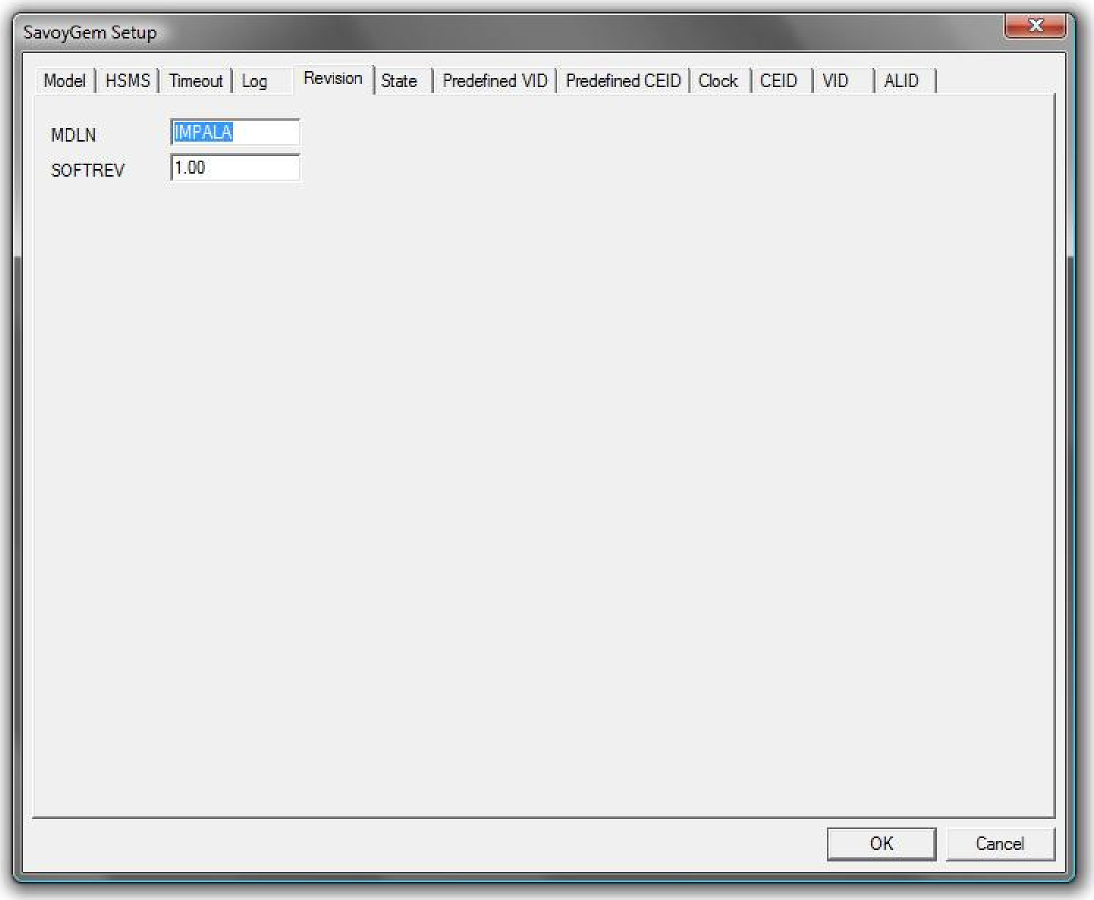
File Name ./bop

Number Of Backup Files 1

Maximum Size Of Each File 1024 KB

Item	Description
Enable logging	True if logging is enabled.
Enable communication log	True if communication logging is enabled.
File name	Log file name.
Number of backup files	Number of backup files.
Maximum size of each file	Size of log file.

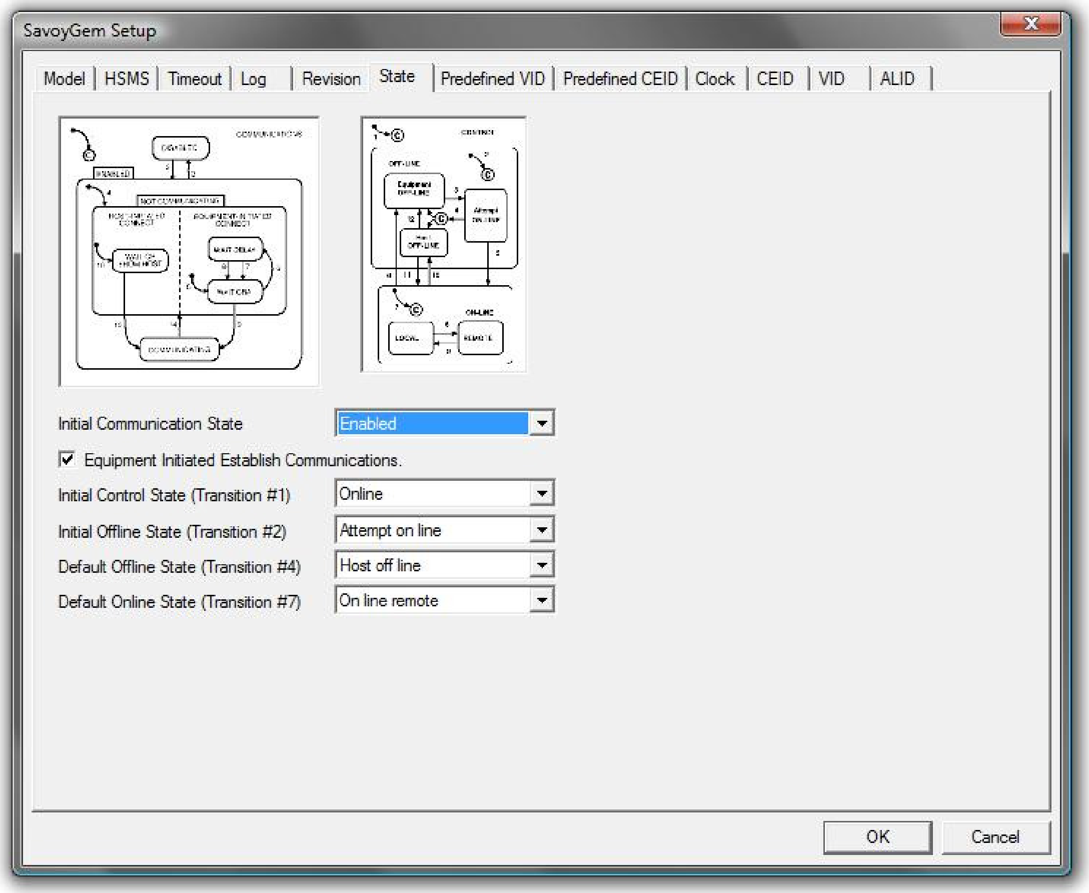
Revision Tab



MDLN: IMPALA
SOFTREV: 1.00

Item	Description
MDLN	Model name of equipment.
SOFTREV	Revision number of equipment.

State Tab



SavoyGem Setup

Model | HSMS | Timeout | Log | Revision | State | Predefined VID | Predefined CEID | Clock | CEID | VID | ALID

Initial Communication State: Enabled

[✓] Equipment Initiated Establish Communications.

Initial Control State (Transition #1): Online

Initial Offline State (Transition #2): Attempt on line

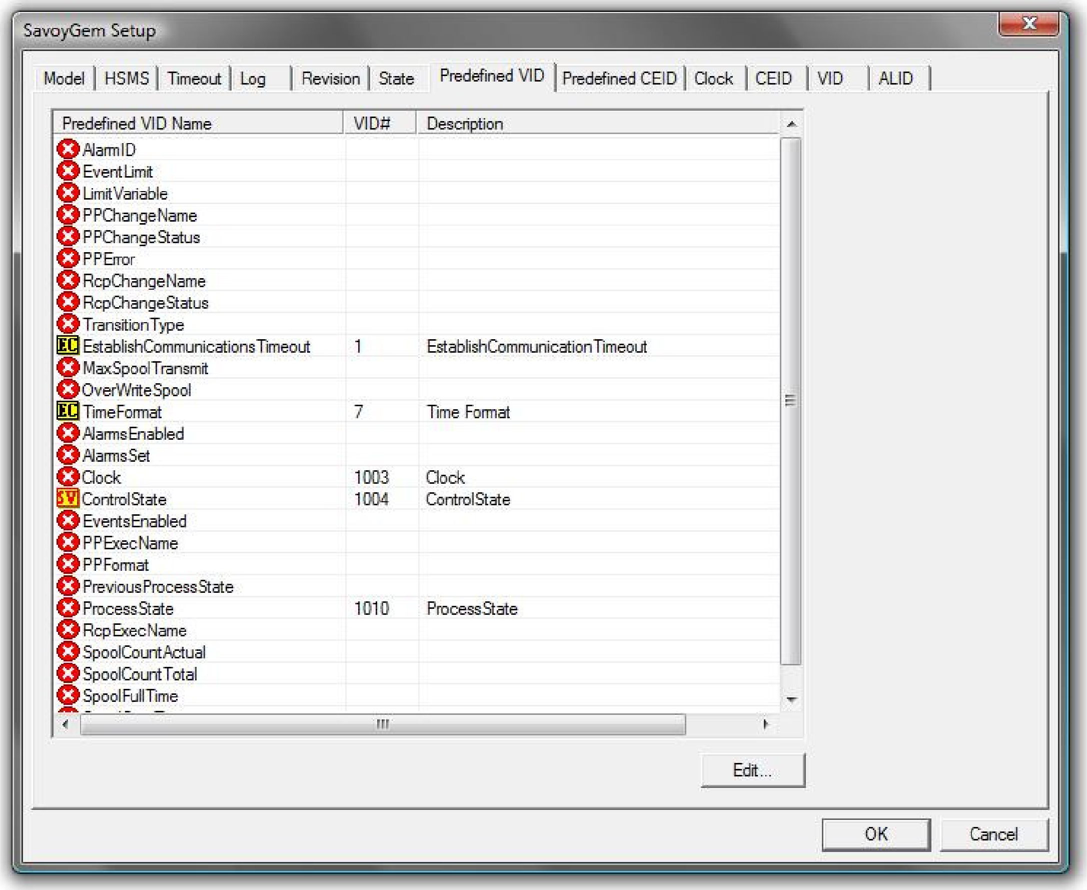
Default Offline State (Transition #4): Host off line

Default Online State (Transition #7): On line remote

OK | Cancel

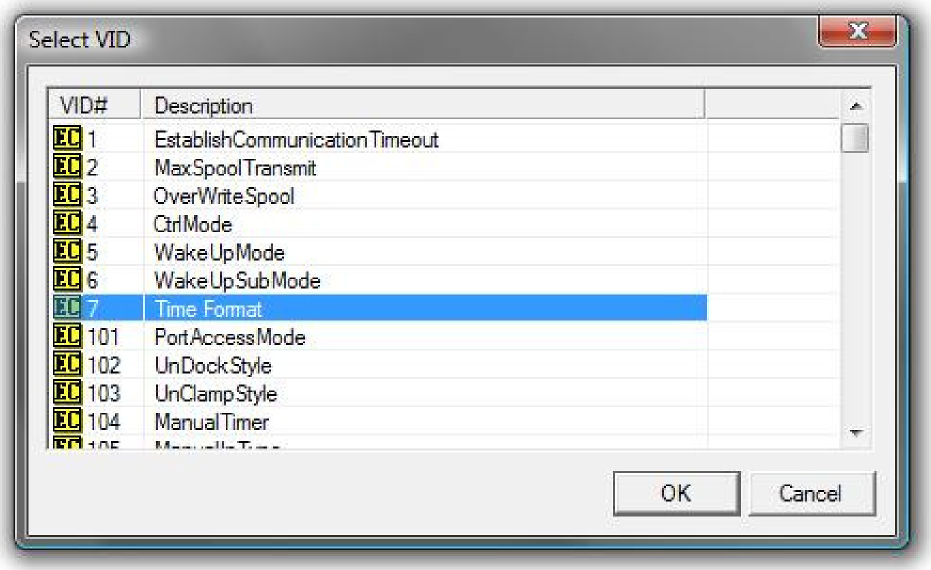
Item	Description
Initial Communication State	True if communication is enable at the beginning of application.
Equipment Initiated EC	True if equipment will send (initiate) S1F13.
Initial Control State	Initial state of control mode.
Initial Offline State	Initial state of off-line state.
Default Offline State	Default off-line state.
Default Online State	Default on-line state.

Predefined VID Tab


SavoyGem Setup
Model | HSMS | Timeout | Log | Revision | State | Predefined VID | Predefined CEID | Clock | CEID | VID | ALID
Edit...
OK
Cancel

Predefined VID Name	VID#	Description
AlarmID		
EventLimit		
LimitVariable		
PPChangeName		
PPChangeStatus		
PPError		
RcpChangeName		
RcpChangeStatus		
TransitionType		
EstablishCommunicationsTimeout	1	EstablishCommunicationTimeout
MaxSpoolTransmit		
OverWriteSpool		
TimeFormat	7	Time Format
AlarmsEnabled		
AlarmsSet		
Clock	1003	Clock
ControlState	1004	ControlState
EventsEnabled		
PPExecName		
PPFormat		
PreviousProcessState		
ProcessState	1010	ProcessState
RcpExecName		
SpoolCountActual		
SpoolCountTotal		
SpoolFullTime		

Item	Description
List	List of predefined VID in SavoyGem
Edit button	If user selects VID and presses this button, dialog box will appear.

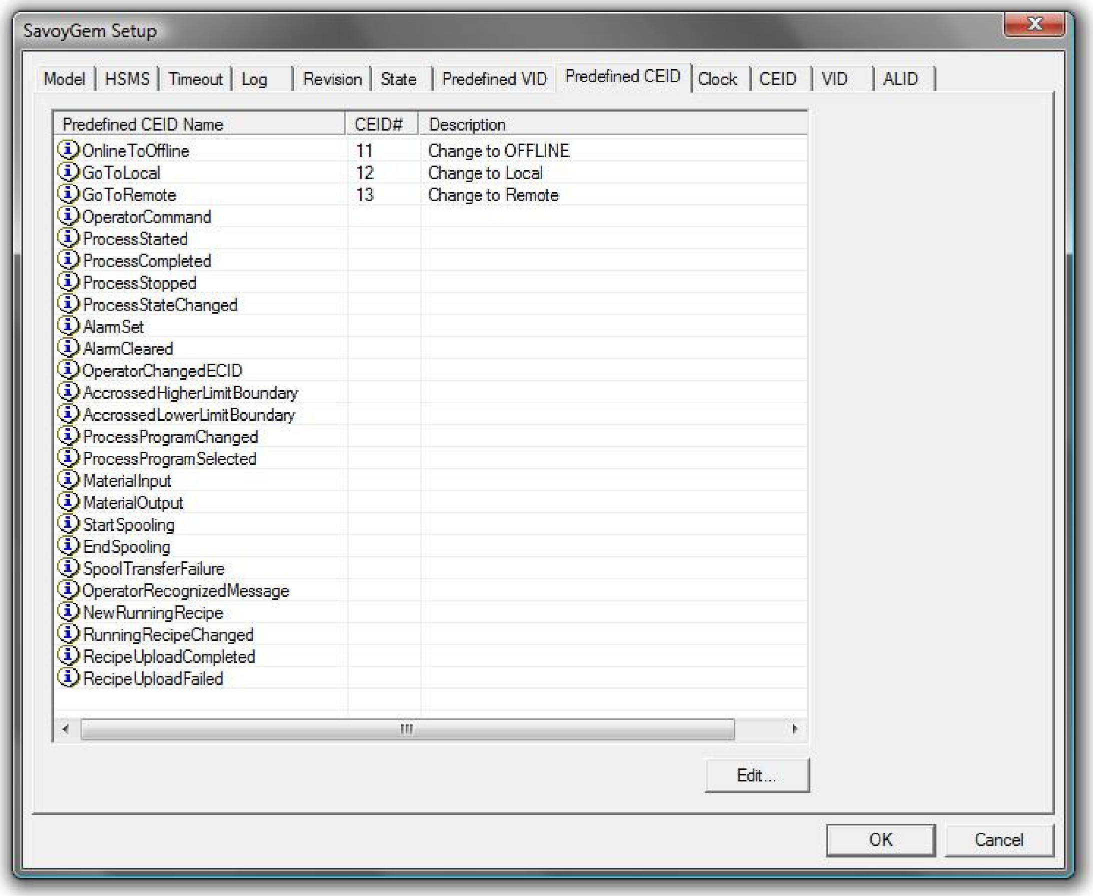


VID#	Description
1	EstablishCommunicationTimeout
2	MaxSpoolTransmit
3	OverWriteSpool
4	CtrlMode
5	WakeUpMode
6	WakeUpSubMode
7	Time Format
101	PortAccessMode
102	UnDockStyle
103	UnClampStyle
104	ManualTimer

Item	Description
List	List of preregistered VID.

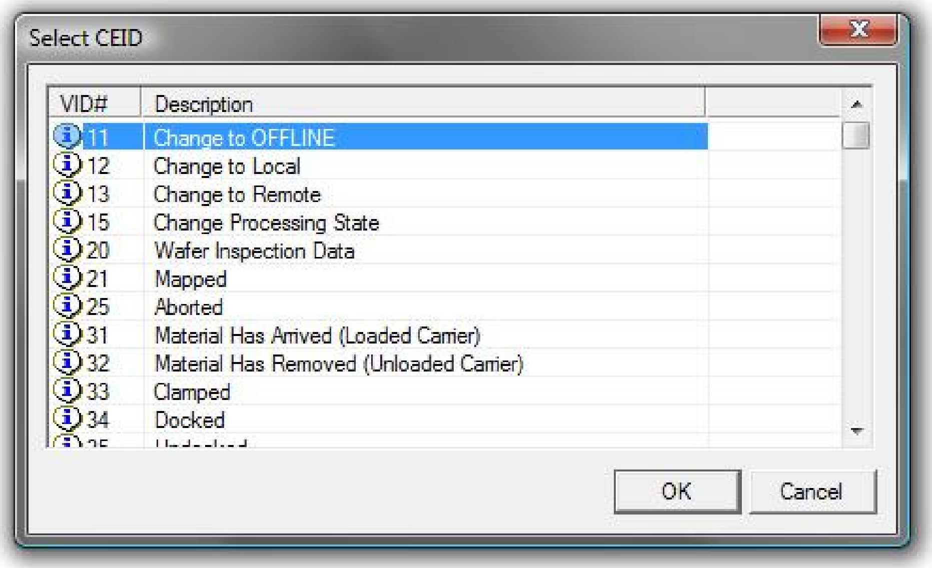
OK button	If user selects VID and presses this button, such VID will be associated.
Cancel button	Cancel modification.

Predefined CEID Tab



Predefined CEID Name	CEID#	Description
OnlineToOffline	11	Change to OFFLINE
GoToLocal	12	Change to Local
GoToRemote	13	Change to Remote
OperatorCommand		
ProcessStarted		
ProcessCompleted		
ProcessStopped		
ProcessStateChanged		
AlarmSet		
AlarmCleared		
OperatorChangedECID		
AccrossedHigherLimitBoundary		
AccrossedLowerLimitBoundary		
ProcessProgramChanged		
ProcessProgramSelected		
MaterialInput		
MaterialOutput		
StartSpooling		
EndSpooling		
SpoolTransferFailure		
OperatorRecognizedMessage		
NewRunningRecipe		
RunningRecipeChanged		
RecipeUploadCompleted		
RecipeUploadFailed		

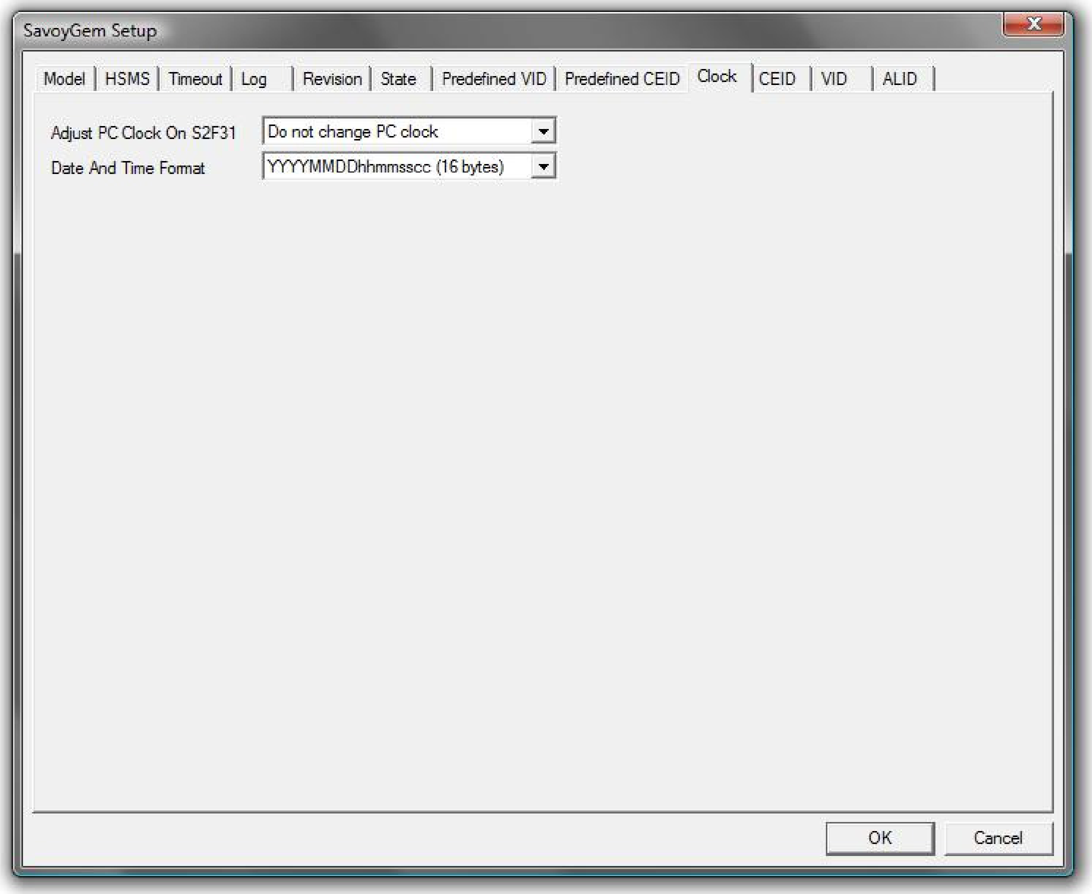
Item	Description
List	List of predefined CEID in SavoyGem.
Edit button	If user selects CEID and presses this button, dialog box will appear.



VID#	Description
11	Change to OFFLINE
12	Change to Local
13	Change to Remote
15	Change Processing State
20	Wafer Inspection Data
21	Mapped
25	Aborted
31	Material Has Arrived (Loaded Carrier)
32	Material Has Removed (Unloaded Carrier)
33	Clamped
34	Docked

Item	Description
List	List of preregistered CEID.
OK button	If user selects CEID and presses this button, such CEID will be associated.
Cancel button	Cancel modification.

Clock Tab

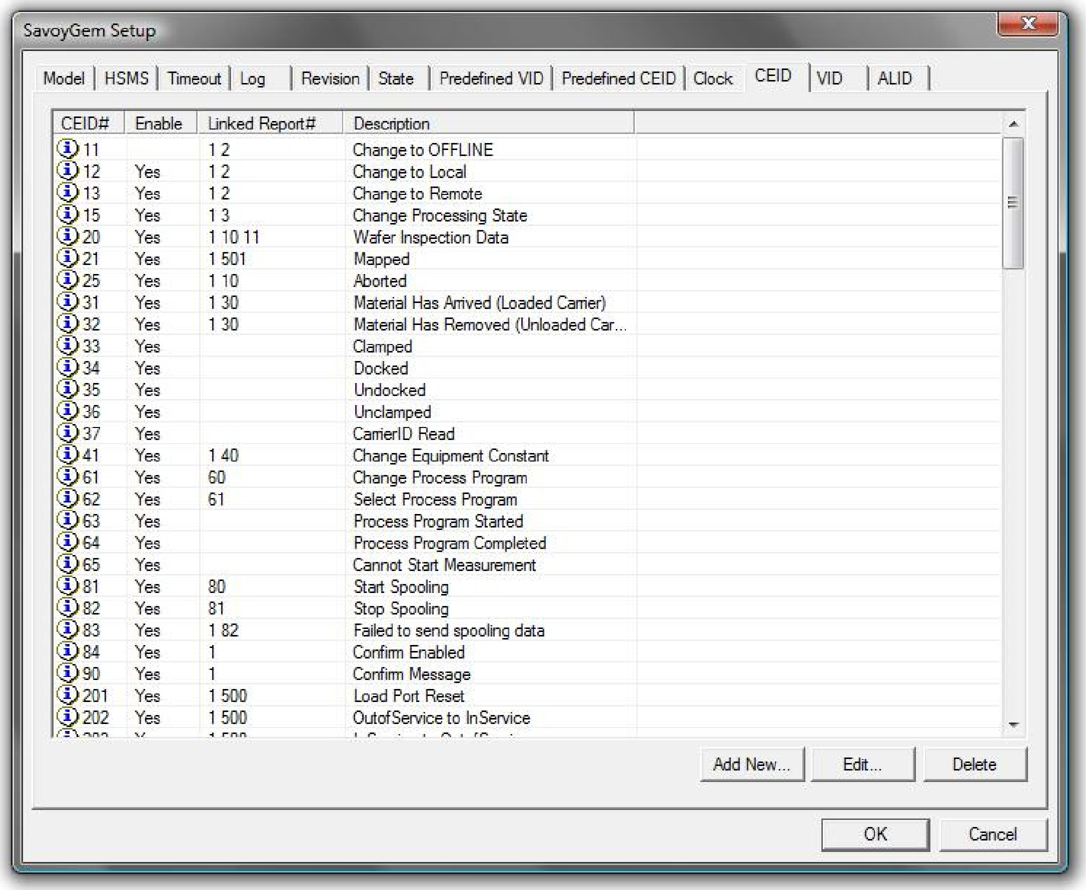


Adjust PC Clock On S2F31: Do not change PC clock

Date And Time Format: YYYYMMDDhhmmsscc (16 bytes)

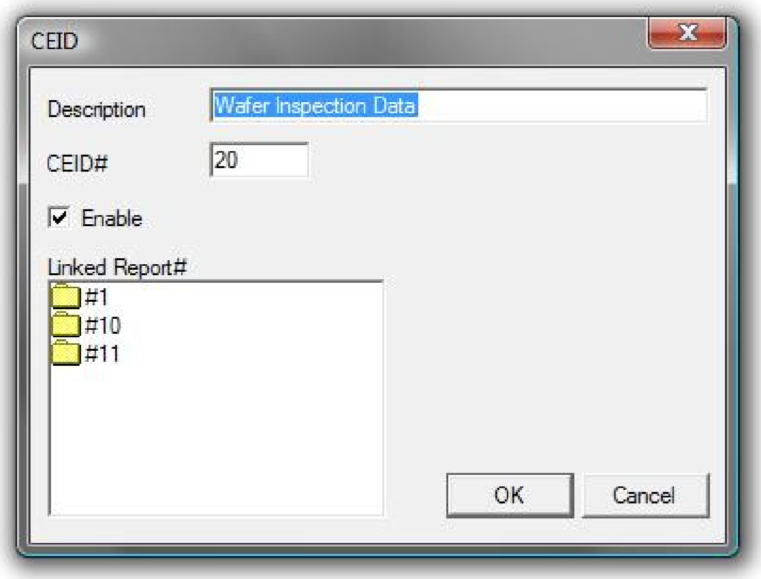
Item	Description
Adjust PC Clock on S2F31	PC clock will be adjusted if SavoyGem receives S2F31.
Date And Time Format	Choose date/time format. 16 bytes format is newer.

CEID Tab



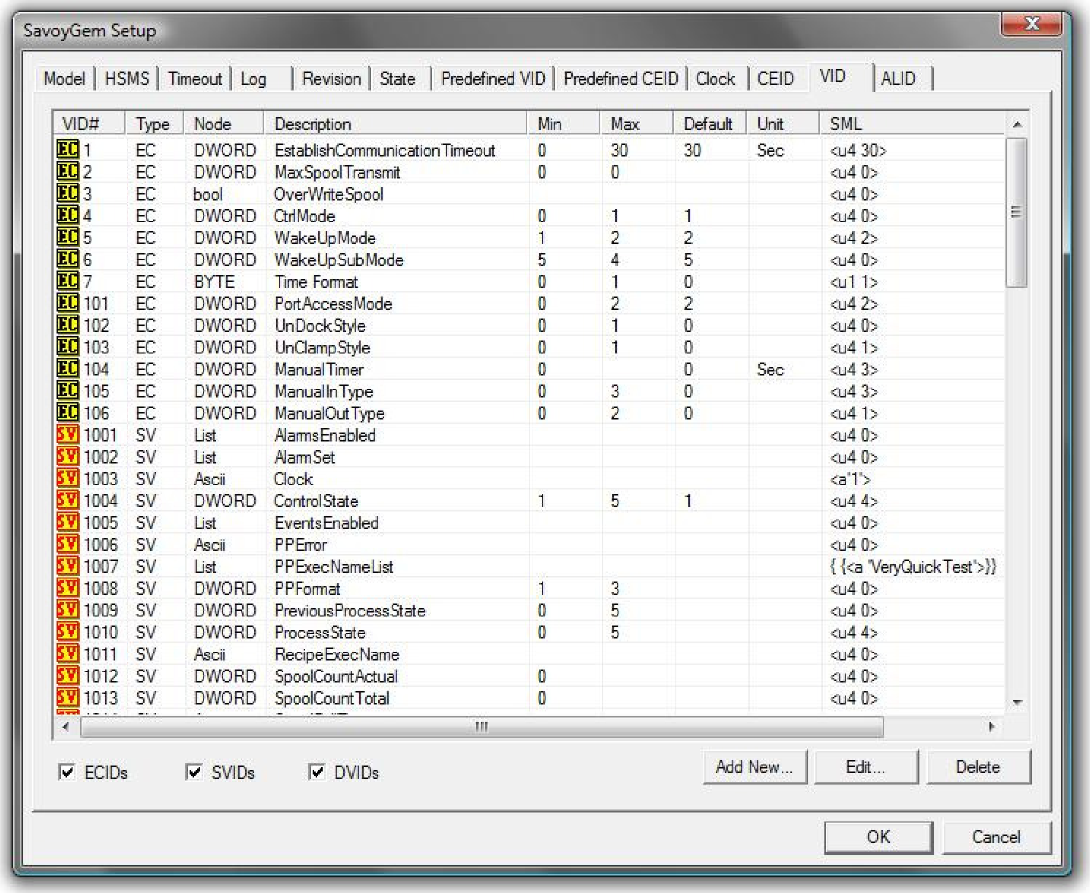
CEID#	Enable	Linked Report#	Description
11		1 2	Change to OFFLINE
12	Yes	1 2	Change to Local
13	Yes	1 2	Change to Remote
15	Yes	1 3	Change Processing State
20	Yes	1 10 11	Wafer Inspection Data
21	Yes	1 501	Mapped
25	Yes	1 10	Aborted
31	Yes	1 30	Material Has Arrived (Loaded Carrier)
32	Yes	1 30	Material Has Removed (Unloaded Car...
33	Yes		Clamped
34	Yes		Docked
35	Yes		Undocked
36	Yes		Unclamped
37	Yes		CarrierID Read
41	Yes	1 40	Change Equipment Constant
61	Yes	60	Change Process Program
62	Yes	61	Select Process Program
63	Yes		Process Program Started
64	Yes		Process Program Completed
65	Yes		Cannot Start Measurement
81	Yes	80	Start Spooling
82	Yes	81	Stop Spooling
83	Yes	1 82	Failed to send spooling data
84	Yes	1	Confirm Enabled
90	Yes	1	Confirm Message
201	Yes	1 500	Load Port Reset
202	Yes	1 500	OutofService to InService

Item	Description
List	List of preregistered CEID by user.
Add New button	Register new CEID.
Edit button	If user selects CEID and presses this button, dialog box will appear.
Delete button	If user selects CEID and presses this button, such CEID will be removed. Confirmation dialog will appear before deletion.



Item	Description
Description	Description of CEID.
CEID#	CEID number.
Enable	Enable or disable this CEID.
Linked Report#	Show linked report(s), if any.
OK button	Save settings.
Cancel button	Cancel modification.

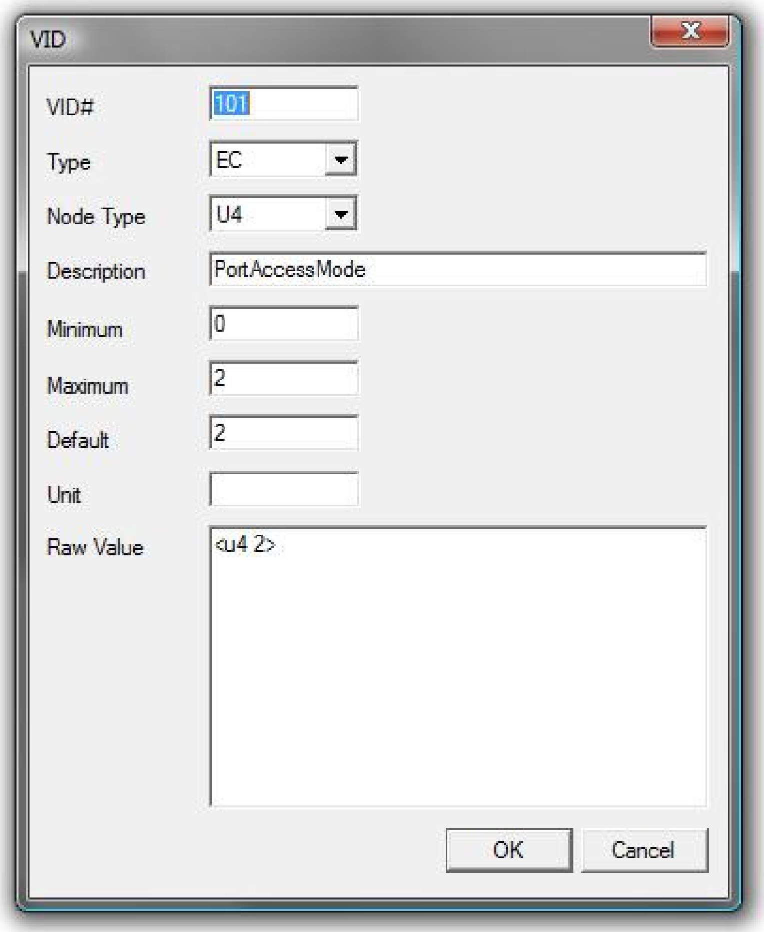
VID Tab


SavoyGem Setup

VID#	Type	Node	Description	Min	Max	Default	Unit	SML
1	EC	DWORD	EstablishCommunicationTimeout	0	30	30	Sec	<u4 30>
2	EC	DWORD	MaxSpoolTransmit	0	0			<u4 0>
3	EC	bool	OverWriteSpool					<u4 0>
4	EC	DWORD	CtrlMode	0	1	1		<u4 0>
5	EC	DWORD	WakeUpMode	1	2	2		<u4 2>
6	EC	DWORD	WakeUpSubMode	5	4	5		<u4 0>
7	EC	BYTE	Time Format	0	1	0		<u1 1>
101	EC	DWORD	PortAccessMode	0	2	2		<u4 2>
102	EC	DWORD	UnDockStyle	0	1	0		<u4 0>
103	EC	DWORD	UnClampStyle	0	1	0		<u4 1>
104	EC	DWORD	ManualTimer	0		0	Sec	<u4 3>
105	EC	DWORD	ManualInType	0	3	0		<u4 3>
106	EC	DWORD	ManualOutType	0	2	0		<u4 1>
1001	SV	List	AlarmsEnabled					<u4 0>
1002	SV	List	AlarmSet					<u4 0>
1003	SV	Ascii	Clock					<a'1'>
1004	SV	DWORD	ControlState	1	5	1		<u4 4>
1005	SV	List	EventsEnabled					<u4 0>
1006	SV	Ascii	PPError					<u4 0>
1007	SV	List	PPExecNameList					{ {<a 'VeryQuickTest'>}}
1008	SV	DWORD	PPFormat	1	3			<u4 0>
1009	SV	DWORD	PreviousProcessState	0	5			<u4 0>
1010	SV	DWORD	ProcessState	0	5			<u4 4>
1011	SV	Ascii	RecipeExecName					<u4 0>
1012	SV	DWORD	SpoolCountActual	0				<u4 0>
1013	SV	DWORD	SpoolCountTotal	0				<u4 0>

☑ ECIDs ☑ SVIDs ☑ DVIDs

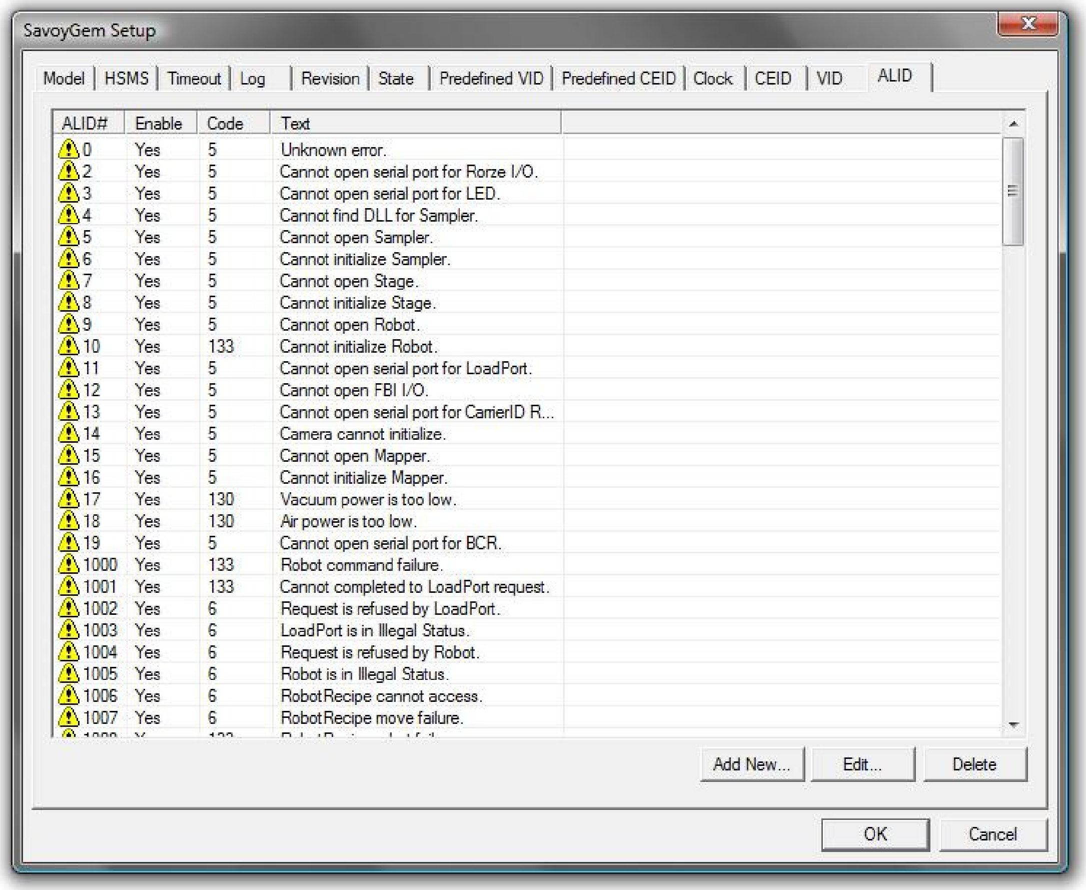
Item	Description
List	List of preregistered VID by user.
ECIDs	If this was checked, equipment constant would appear in list.
SVIDs	If this was checked, system variable would appear in list.
DVIDs	If this was checked, data variable would appear in list.
Add New button	Register new VID.
Edit button	If user selects VID and presses this button, dialog box will appear.
Delete button	If user selects VID and presses this button, such VID will be removed. Confirmation dialog will appear before deletion.



VID# 101
Type EC
Node Type U4
Description PortAccessMode
Minimum 0
Maximum 2
Default 2
Unit
Raw Value <u4 2>

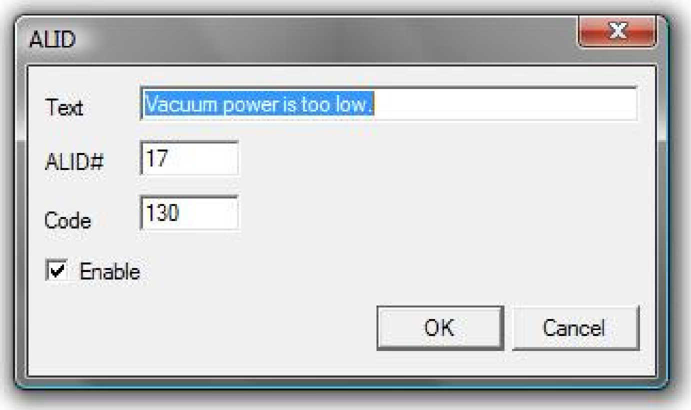
Item	Description
VID#	Variable ID number.
Type	Variable type (one of EC, SV or DV).
Node Type	Variable node type.
Description	Description of variable.
Minimum	Minimum value.
Maximum	Maximum value.
Default	Default value.
Unit	Unit of variable.
Raw Value	SML string. This value will be updated by SavoyGem all the time.
OK button	Save settings.
Cancel button	Cancel modification.

ALID Tab



ALID#	Enable	Code	Text
0	Yes	5	Unknown error.
2	Yes	5	Cannot open serial port for Rorze I/O.
3	Yes	5	Cannot open serial port for LED.
4	Yes	5	Cannot find DLL for Sampler.
5	Yes	5	Cannot open Sampler.
6	Yes	5	Cannot initialize Sampler.
7	Yes	5	Cannot open Stage.
8	Yes	5	Cannot initialize Stage.
9	Yes	5	Cannot open Robot.
10	Yes	133	Cannot initialize Robot.
11	Yes	5	Cannot open serial port for LoadPort.
12	Yes	5	Cannot open FBI I/O.
13	Yes	5	Cannot open serial port for CarrierID R...
14	Yes	5	Camera cannot initialize.
15	Yes	5	Cannot open Mapper.
16	Yes	5	Cannot initialize Mapper.
17	Yes	130	Vacuum power is too low.
18	Yes	130	Air power is too low.
19	Yes	5	Cannot open serial port for BCR.
1000	Yes	133	Robot command failure.
1001	Yes	133	Cannot completed to LoadPort request.
1002	Yes	6	Request is refused by LoadPort.
1003	Yes	6	LoadPort is in Illegal Status.
1004	Yes	6	Request is refused by Robot.
1005	Yes	6	Robot is in Illegal Status.
1006	Yes	6	RobotRecipe cannot access.
1007	Yes	6	RobotRecipe move failure.

Item	Description
List	List of preregistered ALID by user.
Add New button	Register new ALID.
Edit button	If user selects ALID and presses this button, dialog box will appear.
Delete button	If user selects ALID and presses this button, such ALID will be removed. Confirmation dialog will appear before deletion.



Text: Vacuum power is too low.
ALID#: 17
Code: 130
☑ Enable

Item	Description
Text	Alarm text.
ALID#	Alarm ID number.
Code	Alarm code.
Enable	Enable or disable this alarm.

OK button	Save settings.
Cancel button	Cancel modification.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lpszCaption	Caption title of dialog box. If this value is NULL or “” (empty) string, the string of IniSection property will be used for caption tile.
lOptionFlag	Specify which tab will appear. This would be one or combination of followings. If user wants to display all, specify -1.

Value	Description
0x0001	Model
0x0002	HSMS
0x0004	Timeout
0x0008	Revision
0x0010	State model
0x0020	Clock
0x0040	CEID
0x0080	VID
0x0100	ALID
0x0200	Log
0x0400	Predefined VID
0x0800	Predefined CEID

Return Value

If user pressed OK button and parameters were saved in INI file successfully, Setup method returns true. If user pressed Cancel button or parameter saving was failed, Setup method returns false.

Example

Visual Basic 6.0

```
.Setup “GEM”, -1
```

Visual C++ 6.0

```
m_ctrl.Setup(“GEM”, -1);
```

Remarks

At least one tab should be specified. Therefore, lOptionFlag should not be 0.

See Also

3.3.14 ToALID

Converts index to alarm ID. Available value range for index is between 0 and (ALIDCount -1).

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lIndex	Index to alarm ID

Return Value

Converted alarm ID. If index is out of range, return negative number.

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.3.15 ToCEID

Converts index to collection event ID.　Available value range for index is between 0 and (CEIDCount -1).

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lIndex	Index to collection event ID

Return Value

Converted collection event ID.　If index is out of range, return negative number.

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.3.16 ToVID

Converts index to variable ID.　Available value range for index is between 0 and (VIDCount -1).

Syntax

```
Visual Basic 6.0

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

```
Visual C++ 6.0

long ToVID(long lIndex)
```

Argument	Description
lIndex	Index to variable ID

Return Value

Converted variable ID.　If index is out of range, return negative number.

Example

```
Visual Basic 6.0

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

```
Visual C++ 6.0

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

Remarks

See Also

3.3.17 UnregisterALID

Unregisters alarm ID.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lALID	Alarm ID

Return Value

If unregistration was successful, return true.   If failed, return false.

Example

Visual Basic 6.0

```
.UnregisterALID 325
```

Visual C++ 6.0

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

Remarks

See Also

3.3.18 UnregisterVID

Unregisters variable ID.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lVID	Variable ID

Return Value

If unregistration was successful, return true. If failed, return false.

Example

Visual Basic 6.0

```
.UnregisterVID 1100
```

Visual C++ 6.0

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

Remarks

See Also

3.3.19 WriteLogFile

Writes literal string into log file.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lpszText	Literal string to be written.

Return Value

If writing to log file was successful, return true. If failed, return false.

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.4 Events

3.4.1 CommunicationStateChanged

Notifies that communication state has been changed.

Value	Description
0	Communication was disabled
1	Not communicating
2	Communicating

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
sNewState	New communication state
sPrevState	Previous communication state

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.4.2 Connected

Notifies that HSMS connection has been established.

If passive entity, there is no connection established until active entity will connect.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lpszIPAddress	IP address
lPortNumber	Port number

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.4.3 ConnectionStateChanged

Notifies that HSMS connection state has been changed.

Value	Description
0	Not connected.
1	Connected but not selected.
2	Connected and selected.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
sNewState	New connection state
sPrevState	Previous connection state

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.4.4 ControlStateChanged

Notifies that control state has been changed.

Value	Description
0	Equipment off-line
1	Attempt on-line
2	Host off-line
3	On-line local
4	On-line remote

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
sNewState	New control state
sPrevState	Previous control state

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.4.5 Disconnected

Notifies that HSMS connection has been disconnected.

Syntax

Visual Basic 6.0
Event Disconnected(lpszIPAddress As String, lPortNumber As Long)

Visual C++ 6.0
void OnDisconnected(LPCTSTR lpszIPAddress, long lPortNumber)

Argument	Description
lpszIPAddress	IP address
lPortNumber	Port number

Example

Visual Basic 6.0
Text1.Text = “Disconnected - “ + lpszIPAddress + “ [“ + Format$(lPortNumber) + “]”

Visual C++ 6.0
TRACE(“Disconnected - %s [%d]”,lpszIPAddress,lPortNumber);

Remarks

See Also

3.4.6 Problem

Notifies that error has occurred.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
sErrorCode	Error code (see below)
lpszAdditionalInfo	Additional information (not in use)

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

Error from SavoyGem

Error code	Description
-1	Failed to send message
-2	Received message bigger than buffer size
-3	(not used)
-4	T8 timeout
-5	T3 timeout
-6	T5 timeout
-7	T6 timeout
-8	T7 timeout

Error from WinSock

Error code	Description
10093	Socket has not been initialized.
10050	Network subsystem error.
10048	Socket local address is in use.
10014	Invalid user address (like invalid character).
10036	Service provider is in progress.
10049	Remote address can not be available.
10047	Cannot use specified address family for this socket.
10061	Connection has been refused.
10039	?
10022	Invalid listening socket.
10056	Already connected.
10024	?
10051	Cannot reach to network.

10055	Buffer size is not enough.
10038	Not a socket.
10060	Time out before established connection.
10035	Cannot execute right now.

See Also

3.4.7 Received

Notifies that SavoyGem control received message through HSMS.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lpszIPAddress	IP address
lPortNumber	Port number

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also

3.4.8 Sent

Notifies that SECS-II message has been sent.

Syntax

Visual Basic 6.0

```
Event Sent()
```

Visual C++ 6.0

```
void OnSent()
```

Example

Visual Basic 6.0

```
Text1.Text = “Sent”
```

Visual C++ 6.0

```
TRACE(“Sent”);
```

Remarks

See Also

3.4.9 VIDChanged

Notifies that content of variable ID has been changed.

Syntax

Visual Basic 6.0

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

Visual C++ 6.0

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

Argument	Description
lVID	Variable ID

Example

Visual Basic 6.0

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

Visual C++ 6.0

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

Remarks

See Also