SavoySecsII 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	Dec, 22nd, 2009	Hikaru Okada	HSMS property is now supported.

2 Table of Contents

3 SavoySecsII

SavoySecsII control is an assistant product to develop SEMI E5 (SECS-II) compliant application software. SavoySecsII control can be used for either equipment side development or host side development. Usually SavoySecsII control will be used with SavoyHsms and/or SavoySecsI control.

Properties

Name	Description
Appearance	Gets or sets the value that determines the appearance of a SavoySecsII control.
Async	Gets or sets the value whether SavoySecsII control processes SML string in background thread.
BlockNumber	Gets or sets the block number in SECS-II header.
BorderStyle	Gets or sets whether the SavoySecsII control has a border.
DeviceID	Gets or sets the device ID.
Ebit	Gets or sets the end bit in SECS-II header.
Error	Gets whether SML string processing was failed.
ErrorDialog	Gets or sets whether error message dialog box will appear in case SML string processing was not successfully done.
Function	Gets or sets the function number in SECS-II header.
Host	Gets or sets the role of SavoySecsII control.
HSMS	Gets or sets whether SavoySecsII is best match for HSMS or SECS-I.
Msg	Gets or sets the message data of SECS-II.
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.
PType	Gets or sets the presentation type in SECS-II header.
Rbit	Gets or sets the reverse bit in SECS-II header.
SessionID	Gets or sets the session ID for HSMS.
SML	Gets or sets the message in SML literal string.
SourceID	Gets or sets the source ID in SECS-II header.
Stream	Gets or sets the stream in SECS-II header.
SType	Gets or sets the session type in SECS-II header.
SuggestedReplyMsg	Gets the most appropriate reply message determined by veryfying message structure.
SystemBytes	Gets or sets the system bytes in SECS-II header.
TransactionID	Gets or sets the transaction ID in SECS-II header.
Wbit	Gets or sets the wait bit in SECS-II header.
XML	Gets or sets the message of SECS-II in XML literal string.

Methods

Name	Description
AboutBox	Opens version information dialog box on the screen.
Reply	Initializes SECS-II header as reply message of specified message.
Reset	Initializes internal data structure and parameters.
Verify	Verifies message in memory.

Event

Name	Description
Ready	Notifies that SML string has been processed in background thread.

3.1 Properties

3.1.1 Appearance

Gets or sets the value that determines the appearance of a SavoySecsII 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

3.1.2 Async

Gets or sets the value whether SavoySecsII control processes SML string in background thread.

Value	Description
False	Processing is not done in background thread.
True	Processing is done in background thread and its completion will be reported via Ready event.

Syntax

Visual Basic 6.0

```
Async As Boolean
```

Visual C++ 6.0

```
BOOL GetAsync()
void SetAsync (BOOL)
```

Example

Visual Basic 6.0

```
.Async = False
```

Visual C++ 6.0

```
m_ctrl.SetAsync(false);
```

Remarks

Persistent property.

This property is not in use at the moment.

See Also

3.1.3 BlockNumber

Gets or sets the block number in SECS-II header. This property is used only by SECS-I.

For SECS-I following header structure is used.

Byte	Description
1	R | Device ID
2	
3	W | Stream
4	Function
5	E | Block number
6	
7	Source ID
8	
9	Transaction ID
10	

Syntax

Visual Basic 6.0

```
BlockNumber As Long
```

Visual C++ 6.0

```
long GetBlockNumber()
void SetBlockNumber(long)
```

Example

Visual Basic 6.0

```
Dim lBlock As Long
lBlock = .BlockNumber
```

Visual C++ 6.0

```
long lBlock = m_ctrl.GetBlockNumber();
```

Remarks

If BlockNumber property is not 1 on received SECS-I message, the message was multi-block message.

BlockNumber property should always be 1, when sending message. If message size exceeds maximum size of one block, SavoySecsI control will automatically convert it in multi-block message.

See Also

3.1.4 BorderStyle

Gets or sets whether the SavoySecsII 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     ' border
```

Visual C++ 6.0

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

Remarks

Persistent property.

See Also

3.1.5 DeviceID

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

For SECS-I following header structure is used.

Byte	Description
1	R | Device ID
2	
3	W | Stream
4	Function
5	E | Block number
6	
7	Source ID
8	
9	Transaction ID
10	

Syntax

Visual Basic 6.0

```
DeviceID As Long
```

Visual C++ 6.0

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

Example

Visual Basic 6.0

```
.DeviceID = 0     ' Device ID is zero
```

Visual C++ 6.0

```
m_ctrl.SetDeviceID(0);     // Device ID is zero
```

Remarks

Persistent property.

Device ID parameter will be reset by calling Reset method.

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

See Also

3.1.6 Ebit

Gets or sets the end bit in SECS-II header. This property is used only by SECS-I.

Value	Description
False	Not final block
True	Final block

For SECS-I following header structure is used.

Byte	Description
1	R | Device ID
2	
3	W | Stream
4	Function
5	E | Block number
6	
7	Source ID
8	
9	Transaction ID
10	

Syntax

Visual Basic 6.0

```
Ebit As Boolean
```

Visual C++ 6.0

```
BOOL GetEbit()
void SetEbit(BOOL)
```

Example

Visual Basic 6.0

```
If .Ebit = False Then
   ' Never comes here
End If
```

Visual C++ 6.0

```
if(!m_ctrl.GetEbit())
{
   // Never comes here
}
```

Remarks

End bit of incoming SECS-I message is always true. Because SavoySecsI control will notify Received event after the final block was received.

See Also

3.1.7 Error

Gets whether SML string processing was failed.

Value	Description
False	No error
True	SML string was not processed successfully.

Syntax

Visual Basic 6.0

```
Error As Boolean
```

Visual C++ 6.0

```
BOOL GetError()
void SetError(BOOL)
```

Example

Visual Basic 6.0

```
.SML = Text1.Text
If .Error Then
    ...
```

Visual C++ 6.0

```
m_ctrl.SetSml(m_strText1);
if(m_ctrl.GetError())
    ...
```

Remarks

Read-only property.

See Also

3.1.8 ErrorDialog

Gets or sets whether error message dialog box will appear in case SML string processing was not successfully done.

Value	Description
False	Do not show error dialog box
True	Show error dialog box

Syntax

Visual Basic 6.0

```
ErrorDialog As Boolean
```

Visual C++ 6.0

```
BOOL GetErrorDialog()
void SetErrorDialog(BOOL)
```

Example

Visual Basic 6.0

```
.ErrorDialog = False
```

Visual C++ 6.0

```
m_ctrl.SetErrorDialog(false);
```

Remarks

Persistent property.

See Also

3.1.9 Function

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

For SECS-I following header structure is used.

Byte	Description
1	R | Device ID
2	
3	W | Stream
4	Function
5	E | Block number
6	
7	Source ID
8	
9	Transaction ID
10	

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 = 2 And .Fucntion = 42 Then
   ' s2f42
   ...
```

Visual C++ 6.0

```
if(m_ctrl.GetStream()==2 && m_ctrl.GetFucntion()==42)
{
   // s2f42
   ...
```

Remarks

See Also

3.1.10 Host

Gets or sets the role of SavoySecsII control. This property will affect to the result verified by Verify method, and SuggestedReplyMsg property.

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

3.1.11 HSMS

Gets or sets whether SavoySecsII is best match for HSMS or SECS-I. Default value is HSMS.

Value	Description
False	SECS-I
True	HSMS

Syntax

Visual Basic 6.0

```
HSMS As Boolean
```

Visual C++ 6.0

```
BOOL GetHsms()
void SetHsms(BOOL)
```

Example

Visual Basic 6.0

```
.HSMS = true
```

Visual C++ 6.0

```
m_ctrl.SetHsms(true);
```

Remarks

Persistent property.

See Also

3.1.12 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

```
Private Sub SavoySecsI1_Read (ByVal pszMsg As String)
With SavoySecsII1
    .Msg = pszMsg
    Select Case .Stream
    Case 1
        Select Case .Fucntion
        Case 1
            's1f1
            ...
```

Visual C++ 6.0

```
void Cxxx::OnxxxRead(LPCTSTR pszMsg)
{
    m_ctrl.SetMsg(pszMsg);
    switch(m_ctrl.GetStream())
    {
    case 1:
        switch(m_ctrl.GetFucntion())
        {
        case 1:
            // s1f1
            ...
```

Remarks

See Also

3.1.13 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 SavoySecsII control may have some message structure, select root node to update whole structure.

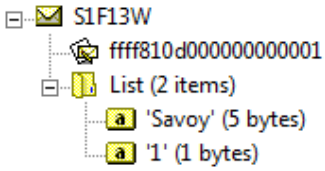
Visual Basic 6.0

```
.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.



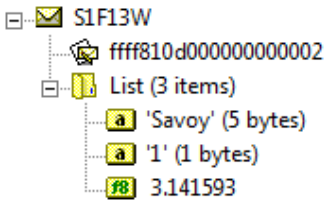
To add 3rd node, set Node property to "3".

Visual Basic 6.0

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

Visual C++ 6.0

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



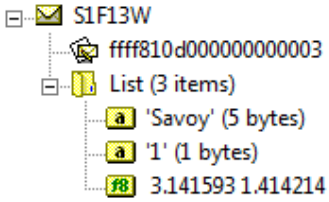
To convert 3rd node into array, set SML using same node type. This case, it is “f8” (8-byte floating point).

Visual Basic 6.0

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

Visual C++ 6.0

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



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.

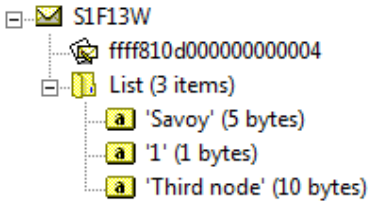
Visual Basic 6.0

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

Visual C++ 6.0

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

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



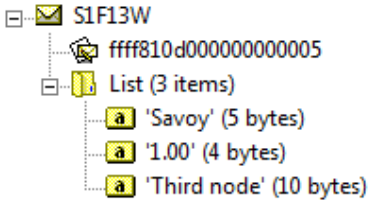
If user wants to concatenate literal strings, set SML using same node type. String is considered as an array of character.

Visual Basic 6.0

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

Visual C++ 6.0

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



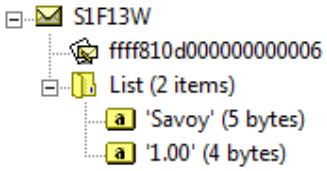
If empty SML was set, the node would be deleted.

Visual Basic 6.0

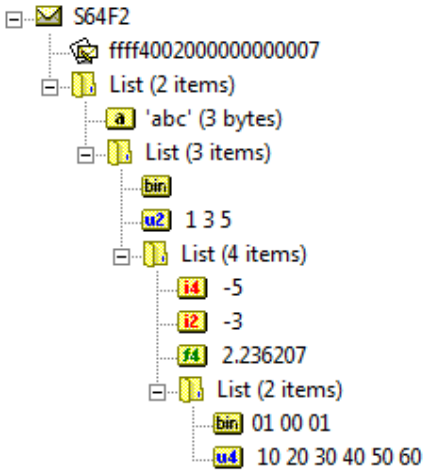
```
.Node = "3"
.SML = ""
```

Visual C++ 6.0

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



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



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”.

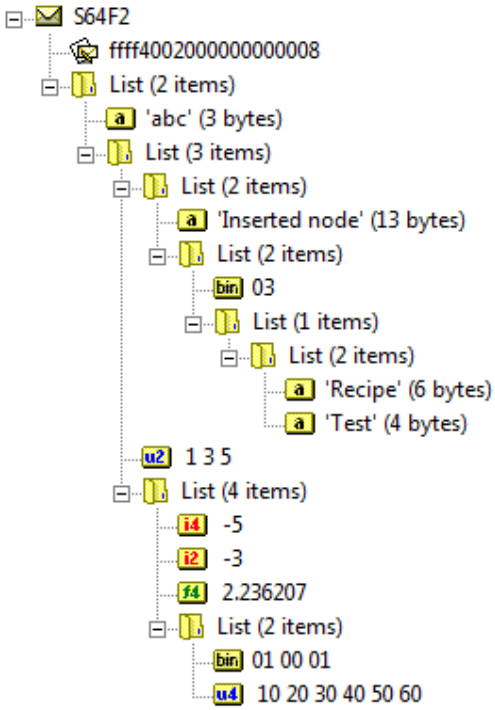
It is possible to set complicated SML structure to node.

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



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

3.1.14 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

3.1.15 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

3.1.16 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

3.1.17 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

3.1.18 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

3.1.19 Rbit

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

Value	Description
False	Host to equipment
True	Equipment to host

For SECS-I following header structure is used.

Byte	Description
1	R | Device ID
2	
3	W | Stream
4	Function
5	E | Block number
6	
7	Source ID
8	
9	Transaction ID
10	

Syntax

Visual Basic 6.0

```
Rbit As Boolean
```

Visual C++ 6.0

```
BOOL GetRbit()
void SetRbit(BOOL)
```

Example

Visual Basic 6.0

```
If .Rbit Then
   MsgBox "Invalid reverse-bit!"
End If
```

Visual C++ 6.0

```
if(m_ctrl.GetRbit())
   MessageBox("Invalid reverse-bit!");
```

Remarks

See Also

3.1.20 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

```
If .SessionID <> &HFFFF Then
   MsgBox "Invalid Session ID!"
End If
```

Visual C++ 6.0

```
if(m_ctrl.GetSessionID()!=0xffff)
   MessageBox("Invalid Session ID!");
```

Remarks

See Also

3.1.21 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

```
.SML = "s1f13w{<a'Savoy'><a'1.00'>}"
```

Visual C++ 6.0

```
m_ctrl.SetSml("s1f13w{<a'Savoy'><a'1.00'>}");
```

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

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. SavoySecsII 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 SavoySecsII 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

3.1.22 SourceID

Gets or sets the source ID in SECS-II header.

For SECS-I following header structure is used.

Byte	Description
1	R | Device ID
2	
3	W | Stream
4	Function
5	E | Block number
6	
7	Source ID
8	
9	Transaction ID
10	

Syntax

Visual Basic 6.0

```
SourceID As Long
```

Visual C++ 6.0

```
long GetSourceID()
void SetSourceID(long)
```

Example

Visual Basic 6.0

```
ctrl1.SourceID = ctrl2.SourceID
```

Visual C++ 6.0

```
m_ctrl1.SetSourceID(m_ctrl2.GetSourceID());
```

Remarks

See Also

3.1.23 Stream

Gets or sets the stream in SECS-II header.

For SECS-I following header structure is used.

Byte	Description
1	R | Device ID
2	
3	W | Stream
4	Function
5	E | Block number
6	
7	Source ID
8	
9	Transaction ID
10	

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

```
Select Case .Stream
Case 6
    Select Case .Fucntion
    Case 11
        's6f11
        ...
```

Visual C++ 6.0

```
switch(m_ctrl.GetStream())
{
case 6:
    switch(m_ctrl.GetFucntion())
```

```
{
case 11:
        // s6f11
        ...
```

Remarks

See Also

3.1.24 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

```
If .SType = 9 Then
   MsgBox "Received Separate.Req!"
End If
```

Visual C++ 6.0

```
if(m_ctrl.GetSType()==9)
   MessageBox("Received Separate.Req!");
```

Remarks

See Also

3.1.25 SuggestedReplyMsg

Gets the most appropriate reply message determined by veryfying message structure.

Syntax

Visual Basic 6.0

```
SuggestedReplyMsg As String
```

Visual C++ 6.0

```
CString GetSuggestedReplyMsg()
```

Example

Visual Basic 6.0

```
SavoySecsII1.Msg = SavoySecsII2.SuggestedReplyMsg
```

Visual C++ 6.0

```
m_send.SetMsg(m_receive.GetSuggestedReplyMsg());
```

Remarks

See Also

3.1.26 SystemBytes

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

For SECS-I following header structure is used.

Byte	Description
1	R | Device ID
2	
3	W | Stream
4	Function
5	E | Block number
6	
7	Source ID
8	
9	Transaction ID
10	

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

```
ctrl1.SystemBytes = ctrl2.SystemBytes
```

Visual C++ 6.0

```
m_ctrl1.SetSystemBytes(m_ctrl2.GetSystemBytes());
```

Remarks

System bytes are 4-byte area and consist of source ID and transaction ID. System bytes in reply message should be identical with the ones in primary message.

See Also

3.1.27 TransactionID

Gets or sets the transaction ID in SECS-II header.

For SECS-I following header structure is used.

Byte	Description
1	R | Device ID
2	
3	W | Stream
4	Function
5	E | Block number
6	
7	Source ID
8	
9	Transaction ID
10	

Syntax

Visual Basic 6.0

```
TransactionID As Long
```

Visual C++ 6.0

```
long GetTransactionID()
void SetTransactionID(long)
```

Example

Visual Basic 6.0

```
ctrl1.TransactionID = ctrl2.TransactionID
```

Visual C++ 6.0

```
m_ctrl1.SetTransactionID(m_ctrl2.GetTransactionID());
```

Remarks

See Also

3.1.28 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 SECS-I following header structure is used.

Byte	Description
1	R | Device ID
2	
3	W | Stream
4	Function
5	E | Block number
6	
7	Source ID
8	
9	Transaction ID
10	

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 (.Fucntion Mod 2) And .Wbit Then
   ' Send default reply message
   ...
End If
```

Visual C++ 6.0

```
if(m_ctrl.GetFucntion() %2 && m_ctrl.GetWbit())
{
   // Send default reply message
   ...
}
```

Remarks

If primary message requested reply message, wait bit will be true.

See Also

3.1.29 XML

Gets or sets the message of SECS-II in XML literal string.

Syntax

Visual Basic 6.0

```
XML As String
```

Visual C++ 6.0

```
CString GetXml()
void SetXml(LPCTSTR)
```

Example

Visual Basic 6.0

```
Text1.Text = .XML
```

Visual C++ 6.0

```
m_text1.Format("%s",(LPCTSTR)m_ctrl.GetSml());
```

Remarks

This property is not in use at the moment.

See Also

3.2 Methods

3.2.1 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

3.2.2 Reply

Initializes SECS-II header as reply message of specified message. If specified message is a HSMS control message, SavoySecsII control will remove message body. Otherwise, message body will not be affected.

Syntax

Visual Basic 6.0

```
Sub Reply(lpszMsgHeader As String)
```

Visual C++ 6.0

```
void Reply(LPCTSTR lpszMsgHeader)
```

Argument	Description
lpszMsgHeader	Set Msg property value of primary message.

Return Value

None.

Example

Visual Basic 6.0

```
.SML = "<b 0>"
.Reply pszMsg
SavoySecsI1.Send .Msg
```

Visual C++ 6.0

```
m_ctrl.SetSml("<b 0>");
m_ctrl.Reply(pszMsg);
m_secs.Send(m_ctrl.GetMsg());
```

Remarks

See Also

3.2.3 Reset

Initializes internal data structure and parameters.

Syntax

Visual Basic 6.0

```
Sub Reset()
```

Visual C++ 6.0

```
void Reset()
```

Return Value

None.

Example

Visual Basic 6.0

```
.Reset
.Stream = 1
.Fucntion = 13
.Wbit = True
```

Visual C++ 6.0

```
m_ctrl.Reset();
m_ctrl.SetStream(1);
m_ctrl.SetFucntion(13);
m_ctrl.SetWbit(true);
```

Remarks

See Also

3.2.4 Verify

Verifies message in memory.

Syntax

Visual Basic 6.0

```
Verify As Integer
```

Visual C++ 6.0

```
short Verify()
```

Return Value

Verification result. 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.

Example

Visual Basic 6.0

```
Dim nResult As Integer
nResult = .Verify()
```

Visual C++ 6.0

```
Int nResult = m_ctrl.Verify();
```

Remarks

If verified message was primary message, suggested reply message would be set to SuggestedReplyMsg property.

See Also

3.3 Events

3.3.1 Ready

Notifies that SML string has been processed in background thread.

Syntax

Visual Basic 6.0

```
Event Ready()
```

Visual C++ 6.0

```
void OnReady()
```

Example

Visual Basic 6.0

```
Text1.Text = “SML string has been processed”
```

Visual C++ 6.0

```
TRACE(“SML string has been processed”);
```

Remarks

This event is not in use at the moment.

See Also