ITLSSPProc eSSP DLL

v1.0

INTELLIGENCE IN VALIDATION

Innovative Technology
INTELLIGENCE IN VALIDATION

Automated Transactions
INTELLIGENCE IN VALIDATION

Bellis Technology
INTELLIGENCE IN VALIDATION

Table of Contents.

1 INTRODUCTION.

This document describes the interface to the ITLSSPProc eSSP DLL.
This dll has been developed to assist in the implementation of eSSP in Windows based system by providing a mechanism to send formatted and encrypted packets to an eSSP enabled target across a serial link from a Windows host.

2 VERSION HISTORY.

Version 1.0 – Initial document release – 11th November 2009

3 STRUCTURE DEFINITIONS.

The dll interface requires defined structures:

3.1 C STRUCTURE DEFINITIONS.

```
/* this structure is used by the host to store the full encryption key. The FixedKey bytes are
defined by the host and must match the slave fixed key  */
typedef struct{
            unsigned __int64 FixedKey;        // 8 byte number for fixed host key
            unsigned __int64 EncryptKey;     // 8 Byte number for variable key
}SSP_FULL_KEY;


/* this structure is required for the key exchange process */
typedef struct{
            unsigned __int64 Generator;
            unsigned __int64 Modulus;
            unsigned __int64 HostInter;
            unsigned __int64 HostRandom;
            unsigned __int64 SlaveInterKey;
            unsigned __int64 SlaveRandom;
            unsigned __int64 KeyHost;
            unsigned __int64 KeySlave;
}SSP_KEYS;

/* Port status code enumeration for ResponseStatus element of  */
typedef enum{
            PORT_CLOSED,
            PORT_OPEN,
            PORT_ERROR,
            SSP_REPLY_OK,
            SSP_PACKET_ERROR,
            SSP_CMD_TIMEOUT,
}PORT_STATUS;

/* a structure to define an SSP command   */
typedef struct{
            SSP_FULL_KEY Key;        // the full key
            unsigned long BaudRate;  // baud rate of the packet
            unsigned long Timeout;  // how long in ms to wait for a reply from the slave
            unsigned char PortNumber;  // the serial com port number of the host
            unsigned char SSPAddress; // the SSP address of the slave
            unsigned char RetryLevel;   // how many retries to the slave for non-response
            unsigned char EncryptionStatus;   // is this an encrypted command 0 – No, 1 - Yes
            unsigned char CommandDataLength; // Number of bytes in the command
            unsigned char CommandData[255];  //  Array containing the command bytes
            unsigned char ResponseStatus;   // Response Status (PORT_STATUS enum)
            unsigned char ResponseDataLength;   // how many bytes in the response
            unsigned char ResponseData[255]; // an array of response data
            unsigned char IgnoreError; // flag to suppress error box (0 – display,1- suppress)
}SSP_COMMAND;
```

```
/* a structure to define an SSP Packet  */
typedef struct{
        unsigned short packetTime;    // the time in ms taken for reply response
        unsigned char PacketLength;  // The length of SSP packet
        unsigned char PacketData[255]; // packet data array
}SSP_PACKET;

/* A structure to define SSP packet info for log file and display purposes  */
typedef struct{
        unsigned char* CommandName;
        unsigned char* LogFileName;
        unsigned char Encrypted;
        SSP_PACKET Transmit;
        SSP_PACKET Receive;
        SSP_PACKET PreEncryptedTransmit;
        SSP_PACKET PreEncryptedRecieve;
}SSP_COMMAND_INFO;
```

3.2 VISUAL BASIC STRUCTURE DEFINITIONS.

```
Public Type EightByteNumber
   LoValue As Long
   Hivalue As Long
End Type


Public Type SSP_FULL_KEY
   FixedKeyLowValue As Long
   FixedKeyHighValue As Long
   EncryptKeyLowValue As Long
   EncryptkeyHighValue As Long
End Type

Public Type SSP_KEYS
   Generator As EightByteNumber
   Modulus As EightByteNumber
   HostInter As EightByteNumber
   HostRandom As EightByteNumber
   SlaveInterKey As EightByteNumber
   SlaveRandom As EightByteNumber
   KeyHost As EightByteNumber
   KeySlave As EightByteNumber
End Type

Public Enum PORT_STATUS
   PORT_CLOSED
   port_open
   PORT_ERROR
   ssp_reply_ok
   SSP_PACKET_ERROR
   SSP_CMD_TIMEOUT
End Enum
```

```
Public Type SSP_COMMAND
    Key As SSP_FULL_KEY
    BaudRate As Long
    Timeout As Long
    PortNumber As Byte
    sspAddress As Byte
    RetryLevel As Byte
    EncryptionStatus As Byte
    CommandDataLength As Byte
    CommandData(254) As Byte
    ResponseStatus As Byte
    ResponseDataLength  As Byte
    ResponseData(254) As Byte
End Type


Public Type SSP_PACKET
    PacketTime As Integer
    PacketLength As Byte
    PacketData(254) As Byte
End Type

Public Type SSP_COMMAND_INFO
    CommandName As String
    LogFileName As String
    Encrypted As Byte
    Transmit As SSP_PACKET
    Recieve As SSP_PACKET
    PreEncryptTransmit As SSP_PACKET
    PreEncryptRecieve As SSP_PACKET
End Type
```

4 API DECLARATIONS

Visual Basic™ 6

```
Public Declare Function OpenSSPComPort Lib "ITLSSPProc.dll" (ByRef sspc As SSP_COMMAND) As Integer
Public Declare Function CloseSSPComPort Lib "ITLSSPProc.dll" () As Integer
Public Declare Function OpenSSPComPort2 Lib "ITLSSPProc.dll" (ByRef sspc As SSP_COMMAND) As Integer
Public Declare Function CloseSSPComPort2 Lib "ITLSSPProc.dll" () As Integer
Public Declare Function OpenSSPComPortUSB Lib "ITLSSPProc.dll" (ByRef sspc As SSP_COMMAND) As Integer
Public Declare Function CloseSSPComPortUSB Lib "ITLSSPProc.dll" () As Integer
Public Declare Function InitiateSSPHostKeys Lib "ITLSSPProc.dll" (ByRef Key As SSP_KEYS, ByRef sspc As
SSP_COMMAND) As Integer
Public Declare Function CreateSSPHostEncryptionKey Lib "ITLSSPProc.dll" (ByRef Key As SSP_KEYS) As Integer
Public Declare Function SSPSendCommand Lib "ITLSSPProc.dll" (ByRef sspc As SSP_COMMAND, ByRef sspinfo As
SSP_COMMAND_INFO) As Integer
```

5 FUNCTION DECLARATIONS.

5.1.0.1 OpenSSPComPort

Parameters:
Pointer to SSP_COMMAND structure
Returns:
WORD 0 for fail, 1 for success

Description:
Opens a serial communication port for SSP data transmission and reception on the host.

Requirements before calling:
SSP_COMMAND structure elements BaudRate and PortNumber need to be correctly filled.

Result after calling:
If function returns 1, host serial port PortNumber is now open for serial comms.

5.1.0.2 OpenSSPComPort2

Parameters:
Pointer to SSP_COMMAND structure
Returns:
WORD 0 for fail, 1 for success

Description:
Opens a serial communication port for SSP data transmission and reception on the host. This opens an additional com port to the port in OpenSSPComPort so that two devices with different serial ports may be used from the same host.

Requirements before calling:
SSP_COMMAND structure elements BaudRate and PortNumber need to be correctly filled. One of the SSP devices used when two ports are open must have an SSP address of 0. (SMART payout or BNV).

Result after calling:
If function returns 1, host serial port PortNumber is now open for serial comms.

5.1.0.3 OpenSSPComPortUSB

Parameters:
Pointer to SSP_COMMAND structure
Returns:
WORD 0 for fail, 1 for success

Description:
Opens a serial communication port for SSP data transmission and reception on the host. This function is used when the host has two or more SSP devices (with different SSP address) connected to the same SSP bus.

Requirements before calling:
SSP_COMMAND structure elements BaudRate and PortNumber need to be correctly filled.

Result after calling:
If function returns 1, host serial port PortNumber is now open for serial comms.

5.1.0.4 CloseSSPComPort

Parameters:

None

Returns:

WORD	0 for fail, 1 for success

Description:

Closes the serial communication port on the host corresponding to the OpenSSPComPort function

Requirements before calling:

An open communication port with PortNumber opened in OpenSSPComPort. Note that calling this function if the port is already closed will have no effect and will still return 1.

Result after calling:

If function returns 1, host serial port PortNumber is now closed for serial comms.

5.1.0.5 CloseSSPComPort2

Parameters:

None

Returns:

WORD	0 for fail, 1 for success

Description:

Closes the serial communication port on the host corresponding to the OpenSSPComPort2 function

Requirements before calling:

An open communication port with PortNumber opened in OpenSSPComPort2. Note that calling this function if the port is already closed will have no effect and will still return 1.

Result after calling:

If function returns 1, host serial port PortNumber is now closed for serial comms.

5.1.0.6 CloseSSPComPortUSB

Parameters:

None

Returns:

WORD	0 for fail, 1 for success

Description:

Closes the serial communication port on the host corresponding to the OpenSSPComPortUSB function

Requirements before calling:

An open communication port with PortNumber opened in OpenSSPComPortUSB. Note that calling this function if the port is already closed will have no effect and will still return 1.

Result after calling:

If function returns 1, host serial port PortNumber is now closed for serial comms.

5.1.0.7 InitiateSSPHostKeys

Parameters:
Pointer to the start of SSP_KEY structure,
Pointer to SSP_COMMAND structure

Returns:
WORD 0 for fail, 1 for success

Description:
Function to create encryption Modulus, Generator and Host Inter numbers. These numbers will be sent to the slave during the key exchange process.

Requirements before calling:

SSP_COMMAND structure element PortNumber need to be correctly filled with the host serial port number.

Result after calling:
SSP encryption packet counter is reset to 0 for that host port number.
SSP_KEY structure will be filled with number values in array order:

Generator	valid
Modulus	valid
HostInter	valid
HostRandom	empty
SlaveInterKey	empty
SlaveRandom	empty
KeyHost	empty
KeySlave	empty

5.1.0.8 CreateSSPHostEncryptionKey

Parameters:
Pointer to the start of SSP_KEY structure,

Returns:
WORD 0 for fail, 1 for success

Description:
Call this function to create the your host key using the SSP_KEY structure populated first by InitiateSSPHostKeys function. This host key will then match the slave key

Requirements before calling:
An SSP_KEY structure populated by call InitiateSSPHostKeys and then sending the Generator and Modulus numbers to the slave (via SSP packets) to populate the SlaveInterKey element of this structure.

Result after calling:
The KeyHost element of the SSP_KEYS structure contains the 8-byte encryption key to combine with the 8-byte fixed key of the host to create the full 128-bit eSSP encryption key for this system.

5.1.0.9 SSPSendCommand

Parameters:

Pointer to SSP_COMMAND structure.
Pointer to SSP_COMMAND_INFO structure.

Returns:

WORD	0 for fail, 1 for success

Description:
Compiles a full ssp packet given a command array, with optional SSP encryption and sends to the slave. The host then waits for a reply, checks it's validity and decrypts if required. The function will retry for the number of times specified in Retrylevel parameter after waiting Timeout milliseconds for a response from the slave.

Requirements before calling:
An open communication port with PortNumber opened in one of the OpenSSPComPort functions.

Result after calling:
The function returns 1 for a successfull transaction – the SSP_COMMAND structure elements ResponseData and ResponseDataLength contains the slave reply data and the ResponseStatus element will be set to SSP_REPLY_OK.
If the function returns 0, the SSP_COMMAND structure elements ResponseData and ResponseDataLength will contain invalid data and the ResponseStatus element will contain the reason for failure as one of the PORT_STATUS enum elements.

6 SENDING AN eSSP COMMAND.

To send an eSSP command, first the host serial port must be opened by sending the OpenSSPComPort command.
We have declared a variable of the type SSP_COMMAND sspCommand and set the parameters:

```
sspCommand.BaudRate = 9600
sspCommand.PortNumber = 1
sspCommand.sspAddress = 0
sspCommand.timeout = 500
sspCommand.ignoreerror = 0
```

Open the serial port by calling:

```
OpenSSPComPort(sspCommand)
```

Test the result of this, if 0 then we have an error, which needs to be investigated. If the result is 1 we have an open serial port.

We have also declared a variable of the type SSP_COMMAND_INFO sspInfo
We can now compile our command. We want to send an SSP sync command (0x11) we will send this as an unencrypted command so no encryption key data is necessary.

```
sspCommand.CommandDataLength = 1
sspCommand.CommandData = 0x11
sspCommand.EncryptionStatus = 0
```

Call the send command function

```
SSPSendCommand(sspCommand, sspInfo)
```

If the result of this is not 0, we can then test the .ResponseStatus element of the structure. An SSP_REPLY_OK value shows a correct transaction.
The .ResponseData and .ResponseDataLength elements can then be examined for SSP reply data.

7 KEY EXCHANGE PROCESS

This section describes how the user can implement the dll functions to create a full system encryption key recognisable by both the host and the slave. This example also demonstrates the SSP command procedure. Note that for multiple slaves, each slave will have a different full key even if the fixed key parts are the same so the host will have to track individual keys for each slave as well as the encryption packet count.
Below is an example of a key exchange process using Visual Basic™ 6

```
Public Function NegotiateKeyExchange(sspc As SSP_COMMAND, sspInfo As SSP_COMMAND_INFO) As Boolean
Dim sspKey As SSP_KEYS
Dim i As Integer

  ' dll function call
  If InitiateSSPHostKeys(sspKey, sspc) = 0 Then
    MsgBox "Error initiating host key modulus or generator values set to zero", vbExclamation,App.ProductName
    Exit Function
  End If

  ' send sync command
  sspc.CommandDataLength = 1
  sspc.EncryptionStatus = 0
  sspc.CommandData(0) = SYNC_CMD
  sspInfo.CommandName = "SYNC"
  If Not TransmitSSPCommand(sspc, sspInfo) Then Exit Function


  sspc.CommandDataLength = 9
  sspc.EncryptionStatus = 0
  sspc.CommandData(0) = cmd_SSP_SET_GENERATOR
  For i = 0 To 3
    sspc.CommandData(1 + i) = CByte(RShift(sspKey.Generator.LoValue, 8 * i) And &HFF)
    sspc.CommandData(5 + i) = CByte(RShift(sspKey.Generator.Hivalue, 8 * i) And &HFF)
  Next i
  sspInfo.CommandName = "SSP_SET_GENERATOR"
  If Not TransmitSSPCommand(sspc, sspInfo) Then Exit Function


  sspc.CommandDataLength = 9
  sspc.EncryptionStatus = 0
  sspc.CommandData(0) = cmd_SSP_SET_MODULUS
  For i = 0 To 3
    sspc.CommandData(1 + i) = CByte(RShift(sspKey.Modulus.LoValue, 8 * i) And &HFF)
    sspc.CommandData(5 + i) = CByte(RShift(sspKey.Modulus.Hivalue, 8 * i) And &HFF)
  Next i
  sspInfo.CommandName = "SSP_SET_MODULUS"
  If Not TransmitSSPCommand(sspc, sspInfo) Then Exit Function


  sspc.CommandDataLength = 9
  sspc.EncryptionStatus = 0
  sspc.CommandData(0) = cmd_SSP_REQ_KEY_EXCHANGE
  For i = 0 To 3
    sspc.CommandData(1 + i) = CByte(RShift(sspKey.HostInter.LoValue, 8 * i) And &HFF)
    sspc.CommandData(5 + i) = CByte(RShift(sspKey.HostInter.Hivalue, 8 * i) And &HFF)
  Next i
  sspInfo.CommandName = "cmd_SSP_REQ_KEY_EXCHANGE"
  If Not TransmitSSPCommand(sspc, sspInfo) Then Exit Function
  sspKey.SlaveInterKey.LoValue = 0
  sspKey.SlaveInterKey.Hivalue = 0
  For i = 0 To 3
    sspKey.SlaveInterKey.LoValue = sspKey.SlaveInterKey.LoValue + (CLng(sspc.ResponseData(1 + i)) * (256 ^ i))
    sspKey.SlaveInterKey.Hivalue = sspKey.SlaveInterKey.Hivalue + (CLng(sspc.ResponseData(5 + i)) * (256 ^ i))
  Next i


  ' we can now calculate our host key
  If CreateSSPHostEncryptionKey(sspKey) = 0 Then
    MsgBox "Error creating host key", vbExclamation, App.ProductName
    Exit Function
  End If

 ' load the SSP_COMMAND structure with the key
  sspc.Key.EncryptKeyLowValue = sspKey.KeyHost.LoValue
  sspc.Key.EncryptkeyHighValue = sspKey.KeyHost.Hivalue

  NegotiateKeyExchange = True

End Function
```

```
Public Function TransmitSSPCommand(sspc As SSP_COMMAND, sspInfo As SSP_COMMAND_INFO) As Boolean
Dim szaddTx As String, szAddRx As String, szadd As String
Dim i As Integer

   If sspc.EncryptionStatus Then
      If sspc.Key.EncryptkeyHighValue = 0 And sspc.Key.EncryptKeyLowValue = 0 Then
         MsgBox "The host has no key set", vbExclamation, App.ProductName
  '        Call CloseSSPComPort
         Exit Function
      End If
   End If
   ‘ dll call
   Call **SSPSendCommand**(sspc, sspInfo)

 ‘ here we can use the SSP_COMMAND_INFO structure to get the packet data for log or debug purposes
   szaddTx = ""
   For i = 0 To sspInfo.Transmit.PacketLength - 1
      If sspInfo.Transmit.PacketData(i) < &H10 Then
         szadd = "0" & Hex(sspInfo.Transmit.PacketData(i))
      Else
         szadd = Hex(sspInfo.Transmit.PacketData(i))
      End If
      szaddTx = szaddTx & szadd & " "
   Next i
   szAddRx = ""
   For i = 0 To sspInfo.Recieve.PacketLength - 1
      If sspInfo.Recieve.PacketData(i) < &H10 Then
         szadd = "0" & Hex(sspInfo.Recieve.PacketData(i))
      Else
         szadd = Hex(sspInfo.Recieve.PacketData(i))
      End If
      szAddRx = szAddRx & szadd & " "
   Next i



   If sspc.ResponseStatus <> ssp_reply_ok Then
      MsgBox "SSP error to " & sspInfo.CommandName & " command " & GetSSPReplyStatus(sspc.ResponseStatus), vbExclamation,
App.ProductName
      Exit Function
   End If

   If sspc.ResponseData(0) <> OK Then
      MsgBox "Non OK response to command " & sspInfo.CommandName & Chr(13) & Chr(10) & GetGenericData(sspc.ResponseData(0)),
vbExclamation, App.ProductName
      Exit Function
   End If

   TransmitSSPCommand = True

End Function
```