TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

1.1.1 The User Functions System enables a user to define his own calculation within proDAS, using measured or calculated data and generating results, which may be processed further in proDAS. It consists of an editor and an interpreter, the User Function Interpreter. This specification defines the interfaces offered by the User Function Interpreter.

1.2 Scope

1.2.1 This document is intended for programmers of the COM client components using the COM interface(s) specified herein as well as the programmers of the server program offering the COM interface(s).

1.3 Applicable Documents

Number	Title
ES78001.2620	Functional Requirements Document for proDAS
ES78024.2655	Engineering Specification for User Functions
ICD78031.2665	Interface Control Document for Windows Database Server
ICD78031.2661	Interface Control Document for Configuration Server

1.4 Codes and Standards

N/A

1.5 Abbreviations and Definitions

Term	Definition
COM	Component Object Model
DLL	Dynamic Link Library
EHP	External Hook Program
ICD	Interface Control Document
IDL	Interface Definition Language
MTA	Multithreaded Apartment
NaN	Not a Number

UF	User Function
UFI	User Function Interpreter

2. DESIGN

2.1 Introduction

2.1.1 The server shall be an out-of-process server, i.e. the server shall be an executable with the extension *.exe*.

2.1.2 The UFI shall run at the machine where the client was launched, that is, there may be more than one UFI instance running within the proDAS system and the UFI must not be accessed over the network.

2.1.3 The server shall be thread-safe, i.e. the server shall be able to service several clients simultaneously.

2.1.4 The server shall be self-registering by calling it with the argument “/RegServer”. Registration will take place when proDAS is installed.

2.1.5 The server shall have a timeout of 5 seconds, i.e. it shall terminate 5 seconds after the last connection has been destroyed.

2.1.6 The server shall be implemented as multithreaded apartment server (MTA).

2.2 Interface IHookExposed

2.2.1 General

2.2.1.1 This interface will be exposed by all external hook programs.

2.2.2 Design

2.2.2.1 The identification of this interface shall be “UserFunctions.HookExposed”.

2.2.2.2 This interface shall be a dispatch interface.

2.2.2.3 This interface shall be an automation interface.

2.2.3 Methods and Properties

2.2.3.1 Property ConfigID

```
[propget, id(1)] HRESULT ConfigID([out, retval] long *pVal);
[propput, id(1)] HRESULT ConfigID([in] long newVal);
```

Argument Name	Description
*pVal, newVal	This identifies the test configuration. By default, the default configuration will be selected.

A change of the configuration ID during calculations will not be allowed. If a client tries to do this an error shall be generated.

This will return E_FAIL if one tries to set an invalid configuration id. UFI will place an error message string in its error object which can be accessed with the *IErrors* property.

2.2.3.2 Property Parameter

```
[propget, id(3)]
HRESULT Parameter([in] BSTR param, [out, retval] VARIANT *pVal);
[propput, id(3)]
HRESULT Parameter([in] BSTR param, [in] VARIANT newVal);
```

Argument Name	Description
param	Each EHP may define its own parameters which can be set or read, respectively, using this property. This identifies the parameter to be set or read.

Argument Name	Description
*pVal, newVal	This identifies the value of the parameter identified by *param*. The type of the VARIANT depends on the parameter. It will be documented with the concrete EHP.

The UFI will allow any valid user function names of the current configuration as parameter names for *param*, while the second parameter has to be a VARIANT of type BOOL indicating whether the user function is to be enabled or not. A value of TRUE shall enable the specified function.

For the UFI, all cyclical user functions will be enabled by default.

This will return E_FAIL if one tries to disable or enable an undefined User Function. UFI will place an error message string in its error object which can be accessed with the *IErrors* property.

2.2.3.3 Property InputChannels

```
[propget, id(4)]
HRESULT InputChannels([out, retval] VARIANT *paVal);
```

Argument Name	Description
*paVal	This will identify the channel names which the interpreter requires to calculate any enabled user function accordingly. This will be a one-dimensional SAFEARRAY with element type BSTR where indexing will start with 1.

For the UFI, all cyclic user functions will be enabled by default. That is, the **paVal* will contain the channel names required as input for the cyclic user functions.

2.2.3.4 Property InputValues

```
[propput, id(5)]
HRESULT InputValues([in] VARIANT aVal);
```

Argument Name	Description
aVal	This shall be an array identifying the channel values which the UFI will use for its calculations. This will be a one-dimensional SAFEARRAY with element

Argument Name	Description
	type double where indexing will start with 1.

The size of the input array depends on the setting for the *UseCustomChannels* property.

The default setting for this property will be an array of *NaN*.

This will return E_FAIL if one tries to set a different number of values as channels are defined by the property *InputChannels* or the property *CustomChannels* if usage of custom channels is enabled. UFI will place an error message string in its error object which can be accessed with the *IErrors* property.

2.2.3.5 Property InputQualities

```
[propput, id(6)]
HRESULT InputQualities([in] VARIANT aVal);
```

Argument Name	Description
aVal	This will be an array identifying the channel qualities as a string. This shall be a one-dimensional SAFEARRAY with element type BSTR where indexing will start with 1.

Possible values for the quality are *GOOD*, *SUSPECT* and *BAD.*

The size of the input array depends on the setting for the *UseCustomChannels* property.

The default for this property will be an array of strings with the value *BAD*.

This will return E_FAIL if one tries to set a different number of qualities as channels are defined by the property *InputChannels* or the property *CustomChannels* if the usage of custom channels is enabled. UFI will place an error message string in its error object which can be accessed with the *IErrors* property.

This will return E_FAIL if one tries to set an undefined quality string. An error message string will be placed in the error object of the UFI.

2.2.3.6 Property OutputChannels

```
[propget, id(7)]
HRESULT OutputChannels([out, retval] VARIANT *paVal);
```

Argument Name	Description
*paVal	This will identify the channel names which the interpreter will need for saving the results of the calculations. This will be a one-dimensional SAFEARRAY with element type BSTR where indexing will start with 1.

For the UFI, all cyclic user functions will be enabled by default. That is, the **paVal* will contain the names of the channels which shall store the output data for the cyclic user functions.

2.2.3.7 Property OutputValues

```
[propget, id(8)]
HRESULT OutputValues([out, retval] VARIANT *paVal);
```

Argument Name	Description
*paVal	This shall be an array identifying the resulting channel values which the UFI has calculated. This will be a one-dimensional SAFEARRAY with element type double where indexing will start with 1.

The size of the output array depends on the setting for the *UseCustomChannels* property.

The default for this property will be an array of *NaN*.

2.2.3.8 Property OutputQualities

```
[propget, id(9)]
HRESULT OutputQualities([out, retval] VARIANT *paVal);
```

Argument Name	Description
*paVal	This shall be an array of strings identifying the resulting channel qualities which the UFI has calculated. This will be a one-dimensional SAFEARRAY with element type BSTR where indexing will start with 1.

Possible values for the quality are *GOOD*, *SUSPECT* and *BAD*.

The default for this property will be an array of *BAD* qualities.

The size of the output array depends on the setting for the *UseCustomChannels* property.

2.2.3.9 Property CustomChannels

```
[propput, id(10)]
HRESULT CustomChannels([in] VARIANT aVal);
```

Argument Name	Description
aVal	This enables a client to define a set of channels for which it will provide the values and qualities. The defined set of channels will be the same for input and output data. This shall be a one-dimensional SAFEARRAY with element type BSTR where indexing starts with 1.

Using this will override the *InputChannels* and *OutputChannels* properties, that is, the EHP shall assume its input and output channels are the channels defined by this property if these settings are enabled. For enabling the custom channels the client must set the *UseCustomChannels* property to TRUE.

A client may use this property if it does not want to search for the requested and returned channels to provide or retrieve the values and qualities respectively but instead transmit all the data and store it with the changes made by the EHP. The EHP shall do the search to find the channels it needs.

The default for this is an empty array.

2.2.3.10 Property UseCustomChannels

```
[propget, id(11)]
HRESULT UseCustomChannels([out, retval] BOOL *pVal);
[propput, id(11)]
HRESULT UseCustomChannels([in] BOOL newVal);
```

Argument Name	Description
*pVal, newVal	This identifies if the *CustomChannels* settings will be used. If this is TRUE the *CustomChannels* are enabled. The default value will be FALSE.

Enabling the usage of the custom channels will assume that the client will use the

InputValues and *InputQualities* for setting the input and *OutputValues* and *OutputQualities* for retrieving the resulting data accordingly.

This will return E_FAIL if no *CustomChannels* are defined (that is, the *CustomChannels* property is empty). UFI will place an error message string in its error object which can be accessed with the *IErrors* property.

2.2.3.11 Method Calculate

```
[id(12)] HRESULT Calculate();
```

This will perform a single calculation of the EHP. For the UFI, all enabled user functions will be executed.

This uses the actual values given by the *InputValues* and *InputQualities* properties. If these properties are not used the default values are used. This will not result in an error but in undefined result values and qualities.

2.2.3.12 Property Errors

```
[propget, id(13)] HRESULT Errors([out, retval] LPDISPATCH *pVal);
```

Argument Name	**Description**
*pVal	This will return an interface to the UFIs error message buffer.

Any client may receive description for errors with this interface.

2.2.4 Events Fired

2.2.4.1 There are no events.

2.2.5 Usage Conditions and Restrictions

2.2.5.1 First, a client has to specify a configuration with the *ConfigID* property if it does not want to use the default configuration. If the *ConfigID* property has an invalid value the UFI will raise an error and place an error message in its error object which the client may receive with the *IErrors* interface of the UFI.

2.2.5.2 Enabling and disabling a User Function with the *Parameter* property has to be done before a client reads the *InputChannels* property, because of the set of channels needed for calculation may have be changed. If a client enables a User Function after reading the input channels it will not provide the input values for all channels. The UFI will set the default value for channels to *NaN* with quality *BAD*. These values would be used for calculation if a channel is not initialised.

2.2.5.3 Before a client may use the *Calculate* method, it has to provide the channel values and qualities to the EHP. If a client does not do so the default values are used (see 2.2.3.4, 2.2.3.5). The UFI will not generate an error message.

2.2.5.4 If usage of custom channels is enabled, then the size of the input data arrays have to be the same as the size of the defined custom channels. If *UseCustomChannels* is disabled, then the size for the input data arrays has to be the same as the size of the array returned by *InputChannels*. If the numbers for input values and qualities does not match the number of defined channels an error will be generated (see 2.2.3.10).

2.2.5.5 After calling *Calculate* the client has to store the results. If *UseCustomChannels* is set the EHP returns all the channels defined. Otherwise, the results are identified by *OutputChannels*.

2.2.6 Persistent Data

2.2.6.1 No data are saved persistently.

2.2.7 Examples

2.2.7.1 Example 1

```
‘ This uses the Windows Database server for calculations
‘ on fullset data. The result values of the performed
‘ calculation will replace values in the fullset for the
‘ appropriate channels.
‘
‘ As this demonstrates the usage of the interpreter using
‘ InputChannels and OutputChannels, UseCustomChannels will
‘ be FALSE, which is its default value.
OPTION EXPLICIT
ON ERROR RESUME NEXT

‘ -----> connecting windows database server
DIM DBS, TestInfo, Fullset
DIM DBSErrors

SET DBS = CreateObject("DatabaseServer.TestResult")
SET DBSErrors = DBS.Errors

DBS.ResetAllFilters()
DBS.Filter(1) = "MTU M-15"    ‘ filter test cell
DBS.Filter(2) = "MyTest"      ‘ filter test
DBS.RetrieveTestInfo          ‘ do the filter operation
SET TestInfo = DBS.Item(1)    ‘ get the first test info object
VerifyErrors DBSErrors, "Connecting DBS"

‘ get fullset
TestInfo.RetrieveFullsets     ‘ load the fullsets from the DB
```

```
SET Fullset = TestInfo(1)     ‘ get the first fullset
VerifyErrors DBSErrors, "Getting data source"
‘ <-----

‘ Connect EHP
DIM EHP
DIM EHPErrors

SET EHP = CreateObject("UserFunctions.HookExposed")
SET EHPErrors = EHP.Errors

‘ Setup the EHP
‘ As the default all cyclic user functions are enabled
EHP.ConfigID = 23793    ‘ Set configuration id
VerifyErrors EHPErrors, "Setting configuration id"

‘ Providing input data
DIM ArrInValues, ArrInQualities ‘ input arrays to be created
DIM ArrInput                    ‘ required input channels
DIM Channel                     ‘ current channel accessed
DIM Cnt                         ‘ counter for accessing the array
Cnt = 1

‘ Retrieving channel names of required input data
ArrInput = EHP.InputChannels

‘ Build the input data arrays
DIM ChannelName
FOR EACH ChannelName IN ArrInput

    ‘ Find channel
    SET Channel = Fullset.Find(ChannelName)
    VerifyErrors DBSErrors, "Data Source find channel"

    ‘ Get channel values
    ArrInValues(Cnt) = Channel.Value
    VerifyErrors DBSErrors, "Data Source getting value"

    ArrInQualities(Cnt) = Channel.Quality
    VerifyErrors DBSErrors, "Data Source getting quality"

    SET Channel = Nothing
    Cnt = Cnt + 1
NEXT

‘ Setting input data
EHP.InputValues = ArrInValues
VerifyErrors EHPErrors, "External Hook Input Values"

EHP.InputQualities = ArrInQualities
VerifyErrors EHPErrors, "External Hook Input Values"

‘ Perform the Calculation of the EHP
EHP.Calculate
```

```
‘ Retrieve results
DIM ArrOutValues, ArrOutQualities ‘ output arrays
ArrOutValues = EHP.OutputValues
ArrOutQualities = EHP.OutputQualities

‘ Store the output data arrays
‘ (reuse of ChannelName, Channel, Cnt and DataSource)
DIM ArrOutput                    ‘ delivered output channels
ArrOutput = EHP.OutputChannels

Cnt = 1
FOR EACH ChannelName IN ArrOutput

    ‘ Find channel
    SET Channel = Fullset.Find(ChannelName)
    VerifyErrors DBSErrors, "Data Source find channel"

    ‘ Store channel data
    Channel.Value = ArrOutValues(Cnt)
    VerifyErrors DBSErrors, "Data Source storing value"

    Channel.Quality = ArrOutQualities(Cnt)
    VerifyErrors DBSErrors, "Data Source storing quality"

    SET Channel = Nothing
    Cnt = Cnt + 1
NEXT

‘ Store the results in the same fullset
TestInfo.UpdateFullset(Fullset)
‘ Job was done ...

‘ ---------------------------------
‘ used for verifying errors of both the external hook
‘ program and the database server
SUB VerifyErrors (ErrorObj, sWhere)
   IF Err.Number <> 0 THEN
      DIM Msg
      Msg = "Error " & Err.Number & " in " & sWhere & ": " &_
            vbCRLF & Err.Description & vbCRLF
      WHILE NOT ErrorObj.IsEmpty
         Msg = Msg & vbCRLF & ErrorObj.Message
      WEND
      MsgBox Msg
   END IF
END SUB
```

2.2.7.2 Example 2

```
‘ As this demonstrates using CustomChannels the
‘ UseCustomChannels property must be set to TRUE
‘ explicitly.
‘
‘ The code for connecting the Database Server and for
‘ verifying errors can be found in example 1.
OPTION EXPLICIT

‘ Connecting data source
‘ ...

‘ Connect EHP object
DIM EHP
SET EHP = CreateObject("UserFunctions.HookExposed")

DIM EHPErrors
SET Errors = EHP.Errors

‘ Setup EHP
EHP.ConfigID = 23793
VerifyErrors EHPErrors, "Setting configuration id"

‘ Set CustomChannels to all channels in fullset
EHP.CustomChannels = Fullset.Names
VerifyErrors DBSErrors, "Fullset getting all channel names"

EHP.UseCustomChannels = TRUE        ‘ enable CustomChannels
VerifyErrors EHPErrors, "External Hook enabling custom channels"

‘ Provide the data of all channels
EHP.InputValues = Fullset.Values
VerifyErrors DBSErrors, "Fullset getting all values"
VerifyErrors EHPErrors, "External Hook setting input values"

EHP.InputQualities = Fullset.Qualities
VerifyErrors DBSErrors, "Fullset getting all qualities"
VerifyErrors EHPErrors, "External Hook setting input qualities"

‘ Perform calculation
EHP.Calculate

‘ Store all channel data
Fullset.Values = EHP.OutputValues
VerifyErrors DBSErrors, "Fullset storing all values"

Fullset.Qualities = EHP.OutputQualities
VerifyErrors DBSErrors, "Fullset storing all qualities"

‘ Store results
....
```

2.2.7.3 Example 3

```
‘ This demonstrates the usage of the Parameter property
‘ for enabling an additional function for execution beside the
‘ cyclic ones.
OPTION EXPLICIT

‘ connect data source
‘ ...

‘ Connect EHP object
DIM EHP
DIM EHPErrors

SET EHP = CreateObject("UserFunctions.HookExposed")
SET EHPErrors = EHP.Errors

‘ Setup EHP
EHP.ConfigID = 23973
VerifyErrors EHPErrors, "Setting configuration id"

‘ Enable function "Range Check"
EHP.Parameter("Range Check") = TRUE
VerifyErrors EHPErrors, "Enable function"

‘ Set CustomChannels to all channels of data source
EHP.CustomChannels = DataSource.Names
VerifyErrors DBSErrors, "Data Source getting all channel names"

EHP.UseCustomChannels = TRUE        ‘ enable CustomChannels
VerifyErrors EHPErrors, "External Hook enabling custom channels"

‘ Provide the data of all channels
EHP.InputValues = DataSource.Values
VerifyErrors DBSErrors, "Data Source getting all values"
VerifyErrors EHPErrors, "External Hook setting input values"

EHP.InputQualities = DataSource.Qualities
VerifyErrors DBSErrors, "Data Source getting all qualities"
VerifyErrors EHPErrors, "External Hook setting input qualities"

‘ Perform calculation
EHP.Calculate

‘ Store all channel data
DataSource.Values = EHP.OutputValues
VerifyErrors DBSErrors, "Data Source storing all values"

DataSource.Qualities = EHP.OutputQualities
VerifyErrors DBSErrors, "Data Source storing all qualities"

‘ Store results
...
```

2.3 Interface ISimulation

2.3.1 General

2.3.1.1 This interface exposes specific simulation functionality of the UFI for the User Function Editor. This interface extends the *IHookExposed* interface. Several properties and methods will be disabled for *ISimulation*. This interface is used by the User Function Editor only.

2.3.2 Design

2.3.2.1 The identification of this interface shall be “UserFunctions.Simulation”.

2.3.2.2 The *ISimulation* interface shall contain the *IHookExposed* interface.

2.3.3 Methods and Properties

2.3.3.1 Property ExecutionCount

```
[propget, id(21)]
HRESULT ExecutionCount([out, retval] long *pVal);
[propput, id(21)]
HRESULT ExecutionCount([in] long newVal);
```

Argument Name	Description
*pVal, newVal	This identifies the number of cycles the UFI shall execute during simulation. The default value for this will be 1.

2.3.3.2 Property Configuration

```
[propput, id(22)] HRESULT Configuration([in] LPDISPATCH pCfg);
```

Argument Name	Description
pCfg	This identifies the Application object of the Configuration Server. The current configuration of this object will be used by the UFI as context for the function to be simulated. The default value will be NULL.

This will return E_FAIL if *pCfg* is not a valid configuration.

2.3.3.3 Property UserFunction

```
[propput, id(23)] HRESULT UserFunction([in] LPDISPATCH pFunc);
```

Argument Name	Description
pFunc	This identifies the user function object, i.e. an object implementing the IUserFunction interface defined in Interface Control Document for Configuration Server, which the UFI is to simulate. The default value will be NULL.

This will return E_FAIL if *pFunc* is not a valid User Function.

2.3.3.4 Method Go

```
[id(24)] HRESULT Go();
```

This will start a simulation. The events *Initialisation()*, *FirstTime()* and *LastTime()* will be interpreted accordingly. The enabled user function will be executed *ExecutionCount* times. This will return E_FAIL if no valid configuration nor User Function are set.

2.3.3.5 Method Stop

```
[id(25)] HRESULT Stop();
```

This will stop the current running simulation. It can be used to finish a simulation run in a loop with a high count.

2.3.3.6 Property Duration

```
[propget, id(26)] HRESULT Duration([out, retval] float *pVal);
```

Argument Name	Description
*pVal	This will return the approximate duration of the simulated user function during a single cycle. The default will be 0.0.

This will return E_FAIL if no valid configuration nor User Function are set.

2.3.4 Inherited Methods and Properties

2.3.4.1 Property ConfigID

2.3.4.1.1 This inherited property has no effect for simulations, it will return E_NOTIMPL.

2.3.4.2 Property Parameters

2.3.4.2.1 This inherited property has no effect for simulations, it will return E_NOTIMPL.

2.3.4.3 Property InputChannels

2.3.4.3.1 It will behave as described in *IHookExposed*. Additionally it will return E_FAIL if no valid configuration and User Function are set.

2.3.4.4 Property InputValues

2.3.4.4.1 It will behave as described in *IHookExposed*. In contrast to this the UFI will set the default values of all channels to 0.0. Additionally it will return E_FAIL if no valid configuration and User Function are set.

2.3.4.5 Property InputQualities

2.3.4.5.1 It will behave as described in *IHookExposed*. In contrast to this the UFI will set the default qualities of all channels to GOOD. Additionally it will return E_FAIL if no valid configuration and User Function are set.

2.3.4.6 OutputChannels

2.3.4.6.1 It will behave as described in *IHookExposed*. Additionally it will return E_FAIL if no valid configuration and User Function are set.

2.3.4.7 OutputValues

2.3.4.7.1 It will behave as described in *IHookExposed*. Additionally it will return E_FAIL if no valid configuration and User Function are set.

2.3.4.8 OutputQualities

2.3.4.8.1 It will behave as described in *IHookExposed*. Additionally it will return E_FAIL if no valid configuration and User Function are set.

2.3.4.9 CustomChannels

2.3.4.9.1 This inherited property has no effect for simulations, it will return E_NOTIMPL.

2.3.4.10 Property UseCustomChannels

2.3.4.10.1 This inherited property has no effect for simulations; it will return E_NOTIMPL.

2.3.4.11 Method Calculate

2.3.4.11.1 This inherited method has no effect for simulations; it will return E_NOTIMPL.

2.3.4.12 Property Errors

2.3.4.12.1 This will behave as described for the *IHookExposed* interface returning errors occurred during simulation and execution of the enabled functions.

2.3.5 Events Fired

2.3.5.1 Event Finished

```
[id(1)] void Finished();
```

This event signals the end of the simulation started with the method *Go*.

2.3.5.2 Event CycleFinished

```
[id(2)] void CycleFinished(long remaining);
```

This event signals that one execution cycle was completed.

2.3.6 Usage Conditions and Restrictions

2.3.6.1 The editor has to provide its Configuration Server *IApplication* interface and the interface of a User Function object to be simulated to the UFI, before it can access the *InputValues* and *InputQualities* properties for setting the input data.

2.3.6.2 The only client known until now is the User Function Editor, which uses this interface for its simulation purposes.

2.3.6.3 The client may set a number for the execution count property before calling *Go()* to start the simulation.

2.3.6.4 After simulation the client may access the *OutputValues* and *OutputQualities* properties for showing the results.

2.3.6.5 The Errors property provides error information that occurred during simulation.

2.3.7 Persistent Data

2.3.7.1 No data are saved persistently.

2.3.8 *Example*

```
‘ VBScript used for demonstration purposes only,
‘ this can normally not be done by a VBScript client but
‘ the User Function Editor does so.

‘ Connect the simulation object
DIM SIM
SET SIM = CreateObject("UserFunctions.Simulation")

‘ Get the errors object
DIM Errors
SET Errors = SIM.Errors

‘ Setup simulation
SIM.Configuration = My_CS_Application_Interface
VerifyErrors Errors, "Set configuration"

SIM.UserFunction = My_UserFunction_Interface
VerifyErrors Errors, "Set User Function"

SIM.ExecutionCount = 1              ‘ this is the default

‘ Deliver simulation values
DIM ArrInChannels
ArrInChannels = SIM.InputChannels
VerifyErrors Errors, "Simulation getting input channel names"

DIM ArrInValues, ArrInQualities
DIM ChannelName
DIM Cnt
Cnt = 1

‘ Set default input values
FOR EACH ChannelName IN ArrInChannels
    ArrInValues(Cnt) = CSng(0.0)   ‘ this is the default
    ArrInQualities(Cnt) = CStr("GOOD") ‘ GOOD is the default
    Cnt = Cnt + 1
NEXT

SIM.InputValues = ArrInValues
VerifyErrors Errors, "Simulation setting values"

SIM.InputQualities = ArrInQualities
VerifyErrors Errors, "Simulation setting qualities"

 ‘ Simulate the function ExecutionCount times
SIM.Go
IF NOT Errors.IsEmpty THEN
    DIM MsgText
    MsgText = "Simulation Errors:" & vbCRLF
    WHILE NOT Errors.IsEmpty
        MsgText = MsgText & "Error : " & Errors.Message & vbCRLF
    WEND
    MsgBox MsgText
    WScript.Quit
END IF

‘ Show the simulation results
DIM MsgRes
MsgRes = "Simulation Results:" & vbCRLF

DIM ArrOutChannels, ArrOutValues, ArrOutQualities
ArrOutChannels = SIM.OutputChannels
VerifyErrors Errors, "Simulation getting out put channel names"

ArrOutValues = SIM.OutputValues
VerifyErrors Errors, "Simulation getting result values"

ArrOutQualities = SIM.OutputQualities
VerifyErrors Errors, "Simulation getting qualities"

Cnt = 1
FOR EACH ChannelName IN ArrOutChannels
      MsgRes = MsgRes &_
               ChannelName & " Value: " & ArrOutValues(Cnt) &_
               ", Quality: " & ArrOutQualities(Cnt) & vbCRLF
NEXT
MsgRes = MsgRes & "Time delay: " & SIM.Duration & vbCRLF &_
         "Simulation successfull finished. "
MsgBox MsgRes
```

2.4 Interface IErrors

2.4.1 General

2.4.1.1 This interface represents the buffer for the error messages generated when checking, simulating and executing user functions.

2.4.2 Design

2.4.2.1 This interface shall be a dispatch interface.

2.4.2.2 This interface shall be an automation interface.

2.4.2.3 This interface shall not be directly created. It shall be created by the *Errors* property of the interface *IHookExposed*.

2.4.2.4 Note that this interface will not implement a collection interface.

2.4.3 Methods and Properties

2.4.3.1 Property IsEmpty

```
[propget, id(1)] HRESULT IsEmpty([out, retval] BOOL *pVal);
```

Argument Name	Description
*pVal	This will indicate the availability of error messages. If FALSE there are error messages in the message buffer. The default value will be TRUE.

2.4.3.2 Property Message

```
[propget, id(2)] HRESULT Message([out, retval] BSTR *pVal);
```

Argument Name	Description
*pVal	This will remove the oldest error message from the message buffer and return the message text. As the default the message buffer will be empty.

2.4.3.3 Method Clear

```
[id(3)] HRESULT Clear();
```

This will clean the message buffer even if there are unread error messages in it.

2.4.4 Events Fired

2.4.4.1 No events shall be fired

2.4.5 Usage Conditions and Restrictions

2.4.5.1 There are no usage conditions and restriction.

2.4.6 Persistent Data

2.4.6.1 No data are saved persistently.

2.4.7 Examples

```
SUB VerifyErrors (ErrorObj, sWhere)
   IF Err.Number <> 0 THEN
      DIM Msg
      Msg = "Error " & Err.Number & " in " & sWhere & vbCRLF &_
            Err.Description & vbCRLF
      WHILE NOT ErrorObj.IsEmpty
         Msg = Msg & vbCRLF & ErrorObj.Message
      WEND
      MsgBox Msg
   END IF
END SUB
```