Customizing the Agilent TestExec SL Operator Interface using Visual Basic

Application Note

Overview

This application note describes how users of the Agilent TestExec SL software can customize the operator user interface using Visual Basic.

Operator Interface Overview

Operator interface is the user interface designed for system operators to control the test system in the manufacturing environment. It is usually simplified for non-technical operators and has a lower level of access to the testplan and features of TestExec SL.

Different users will have different requirements for the user interface. TestExec SL offers a customizable user interface that can vary in the appearance, features, interactivity and access levels required. It may range from simple customization to include just basic functions of start and stop buttons, and a pass/fail indicator; or it can be as complicated as to contain multiple-level interfaces for user interactions, with access to external hardware and database sources.

TestExec SL allows you to create custom operator interfaces with Microsoft® Visual Basic and other programming platforms that support the ActiveX controls (e.g. Microsoft Visual C++). Generally, using Visual Basic is the preferred method because of its simplicity and flexibility amongst the textual programming environment.

The best way to begin learning how to customize an operator interface is through understanding the working examples. This application note will familiarize you with basic features of the operator interface and some enhancements that you may consider adding to your customized interface.

Agilent Technologies

Understanding Agilent's TestExec SL ActiveX Control

In order to follow through the example below, you should be familiar with Visual Basic 6 terminology and concept, its integrated development environment (IDE) and the use of ActiveX controls. The operator interface will communicate with TestExec SL through a special set of TestExec SL ActiveX controls.

TestExec SL ActiveX control library is a collection of automation objects with predefined properties and methods. The Visual Basic code controls TestExec SL as a separate process, changing its setting (properties), sending it commands (methods), and waiting for progress to occur (events). Remember to add the "Agilent TestExec SL control library" from the Visual Basic Components List to the toolbox before using them in your interface.

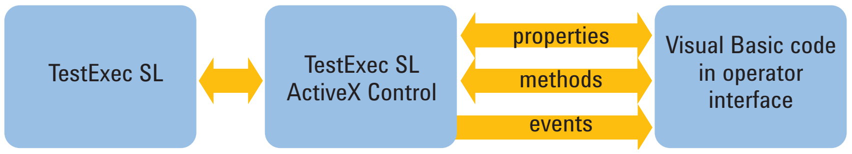

TestExec SL
TestExec SL ActiveX Control
properties
methods
events
Visual Basic code in operator interface

Figure 1. TestExec SL ActiveX control in Visual Basic

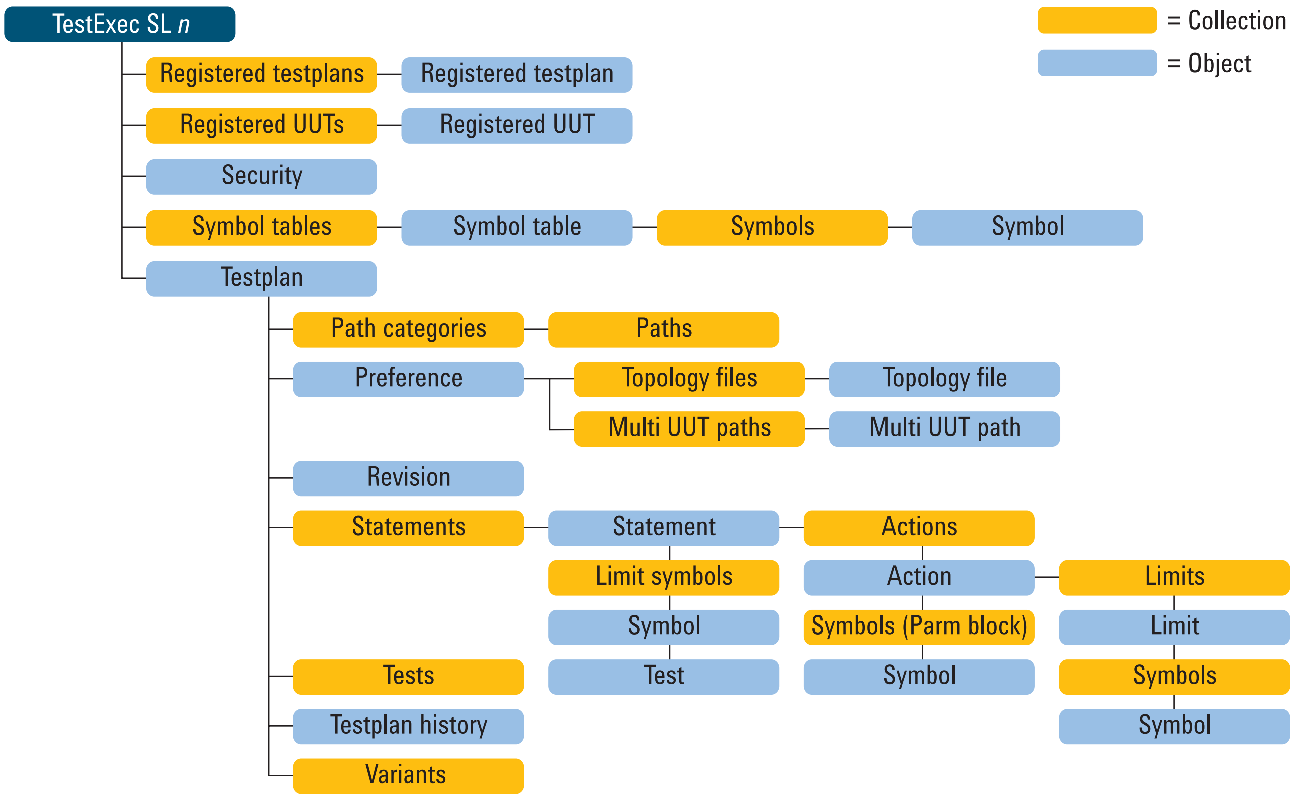


Figure 2. Hierarchy of TestExec SL ActiveX control collections and objects

Designing the Operator Interface Layout

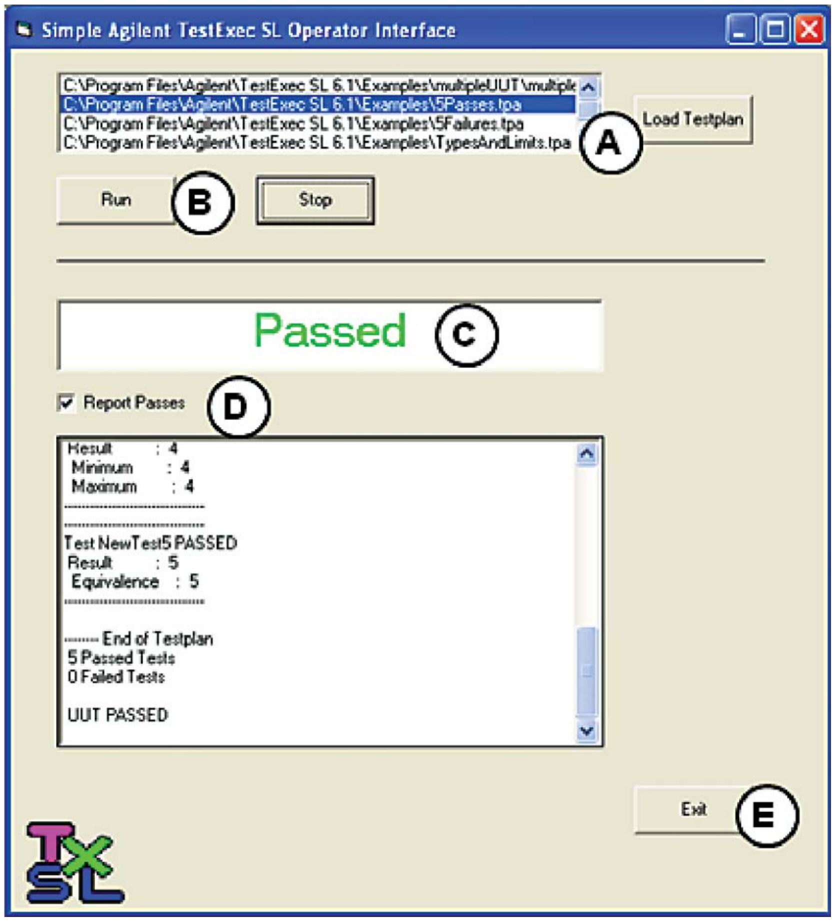
The first step in customizing an operator interface using Visual Basic is to design its interface layout. To design an effective operator interface, other than knowing the needs of your audience, you also have to consider other design practices. A good interface design layout would have the following characteristics:

- Simple appearance with adequate information
- Logical layout with clear flow of tasks

A simple operator interface should include the basic features of (1) selecting and loading a testplan to test a specific product, (2) running and stopping the testplan, (3) showing the test status of either pass or fail, (4) optionally reporting the test results, and (5) exiting the operator interface. Below is an example showing the layout of a simple operator interface.

This operator interface will populate a list box with the paths of registered testplans. It will then allow the running and stopping of the testplans. The state of testplan execution is maintained in a text box. The testplan report information is maintained in a rich text box. The user is given one aspect of control of the testplan, and that is to enable or disable the reporting of passed tests.

It is important that you define the physical appearance and names of all the controls at this stage prior to coding. Use the following properties window to set the properties of controls in Visual Basic environment.



```
C:\Program Files\Agilent\TestExec SL 6.1\Examples\multipleUUT\multiple
C:\Program Files\Agilent\TestExec SL 6.1\Examples\5Passes.tpa
C:\Program Files\Agilent\TestExec SL 6.1\Examples\5Failures.tpa
C:\Program Files\Agilent\TestExec SL 6.1\Examples\TypesAndLimits.tpa
```

```
 Result     : 4
 Minimum     : 4
 Maximum     : 4
...............................

...............................
Test NewTest5 PASSED
 Result     : 5
 Equivalence  : 5
...............................

....... End of Testplan
5 Passed Tests
0 Failed Tests

UUT PASSED
```

Figure 3. Simple Agilent TestExec SL operator interface

Control Type	Name	Properties
Form	frmSimple	Caption = Simple Agilent TestExec SL operator interface
Listbox	lstTestplans	N/A
Command button	cmdLoad	Caption = Load testplan
Command button	cmdRun	Caption = Run
Command button	cmdStop	Caption = Stop
Command button	cmdExit	Caption = Exit
Textbox	txtTxSLState	Font = 24 point, Text = State, Alignment = Center
Checkbox	chkReportPasses	Caption = Report passes
Rich-textbox	rtbReport	Scroll bars = Both
Txsl	testexecsl1	Assume all defaults

Table 1. Properties of controls

Coding the Event Handlers for Your Operator Interface

Upon completion of the interface layout design, the appropriate event handlers have to be created for different controls in the operator interface.

A. Loading registered testplan

A list box is added in the interface to display a list of registered testplan for user's selection. All testplans are registered in the *preferences.upf* file prior to running the operator interface. This allows the developer to control and ensure that only designated testplans are accessible by the operators.

On the initial load of this interface, the coding will check the *RegisteredTestplan* collection of TestExec SL, and use the results to populate the list box *lstTestplans*.

```
Private Sub Form_Load()
    Dim RegisteredTestplanInstance As Object
    Set RegisteredTestplanInstance = TestExecSL1.
RegisteredTestplans(1)
    For Each RegisteredTestplanInstance In TestExecSL1.
RegisteredTestplans
            lstTestplans.AddItem
RegisteredTestplanInstance.Path
    Next RegisteredTestplanInstance
End Sub

Private Sub TestExecSL1_AdviseClearReport()
    rtbReport.Text = ""
End Sub
```

A *LoadTestplan* button is added to allow users to control the testplan loading. The return value of the *LoadTestplan* method can be used to update the interface based on *BeforeTestplanLoad* and *AfterTestplanLoad* events.

```
Private Sub cmdLoadTestplan_Click()
    TestExecSL1.LoadTestplan lstTestplans.Text
End Sub
```

After the testplan is loaded, the *chkReportPasses* checkbox status and testplan state information should be updated on the interface to reflect the correct status of the just loaded testplan.

```
Private Sub TestExecSL1_AfterTestplanLoad(ByVal Path As
String)
    txtTxSLState.Text = "Loaded"
    If TestExecSL1.Testplan.Preference.ReportPass Then
            chkReportPasses.Value = vbChecked
    Else
            chkReportPasses.Value = vbUnchecked
    End If
        chkReportPasses.Enabled = True

    cmdRun.Enabled = True
    cmdRun.SetFocus
End Sub
```

B. Running and stopping testplan

In all operator interfaces, after selecting and loading the testplan, *Run* and *Stop* buttons are the necessary functions to control the states of testplan execution. Below are the source codes assigned to the *Run* and *Stop* buttons.

```
Private Sub cmdRun_Click()
    TestExecSL1.Testplan.Run
End Sub

Private Sub cmdStop_Click()
    TestExecSL1.Testplan.Stop
End Sub
```

For actual applications, users may consider adding coding to check the testplan state (e.g. running or not running) before executing the .Run and .Stop TestExec SL methods. Furthermore, users may also add the *Abort* button to immediately terminate a running testplan under certain urgent situations. Aborting a testplan is different from stopping a testplan where all hardware and software states will still remained active as when it is terminated. However, the latter can be coded with proper test procedures to end a testplan.

C. Displaying the state of testplan execution

The next important task is to display the information of the testplan state (e.g. testing, passed, failed, etc.) on the operator interface.

Users can acquire the state information directly through TestExec SL state model and can customize the display information such as the display text, its properties and any animation effects. These can be coded for *BeforeTestplanBegin* and *AfterTestplanStop* events in TestExec SL. We use the return parameter Reason in the latter to set the text and color of the *txtTxSLState* text box.

```
Private Sub TestExecSL1_BeforeTestplanBegin(ByVal Count
As Long)
    txtTxSLState.Text = "Testing"
End Sub

Private Sub TestExecSL1_AfterTestplanStop(ByVal Reason
As HPTestExecSL.TestplanState)
    Select Case Reason
        Case HPTestExecSL.TestplanPassed
            txtTxSLState.ForeColor = vbGreen
            txtTxSLState.Text = "Passed"
        Case HPTestExecSL.TestplanFailed
            txtTxSLState.ForeColor = vbRed
            txtTxSLState.Text = "Failed"
        Case HPTestExecSL.TestplanStopped
            txtTxSLState.ForeColor = vbRed
            txtTxSLState.Text = "Stopped"
        Case Else
            txtTxSLState.ForeColor = vbRed
            txtTxSLState.Text = "Unknown Exit"
    End Select
End Sub
```

D. Showing the testplan report

Typically, to display testplan report information, a rich text box will be used instead of just a standard text box. This is because a standard text box has limited size, which is not appropriate for long reports. A checkbox is included in the interface to offer the user the option to display the detailed report information.

```
Private Sub chkReportPasses_Click()
    If chkReportPasses.Value = vbChecked Then
        TestExecSL1.Testplan.Preference.ReportPass = True
    Else
        TestExecSL1.Testplan.Preference.ReportPass = False
    End If
End Sub

Private Sub TestExecSL1_ReportMessage(ByVal Message As
String)
    rtbReport.SelLength = 1000000
    rtbReport.SelText = Message
End Sub
```

E. Exiting the operator interface

Last but not least, a complete operator interface should have an exit button to unload the application.

```
Private Sub cmdExit_Click()
    Unload Me
End Sub
```

Additional Enhancement Features for Operator Interface

What constitutes a basic operator interface is usually not sufficient for an actual manufacturing environment due to limited functionality. More event management coding is needed to make the interface more robust. For example, the *LoadTestplan* button should only be enabled when a testplan is selected.

More often than not, an operator interface also requires some error trapping routines and needs to be more informative and interactive. Additional information (e.g. company logo, progress bar, and test yield information) will certainly improve the usability of the interface.

Besides the appearance and layout, useful prompts and status information are also vital to an operator interface for improving the user interactivity. Direct and precise messages are essential to guide the operators on their tasks. In some cases, keyboard shortcuts are also practical to have on top of mouse navigation features.

One of the main reasons to create a customized interface is to control the user access to the test system. The level of access is very much dependent on the user types — operator, supervisor, developer and administrator. TestExec SL allows you to create a single interface with multiple logins for different level of access.

Accessing External Hardware Resources

Other than communicating with TestExec SL controls, the operator interface can also be customized to access external hardware resources for partially or fully automated manufacturing lines. You may want to control switches and sensors using a digital I/O card or a bar code reader through RS-232 serial interface. In the *TypicalOpUI* example, you will learn how to access the external bar code reader from the operator interface. It synchronizes the testplan with bar code reader events.

There are two methods in writing Visual Basic codes to interact with the external hardware resources from operator interface. The first method is to access the hardware directly via an I/O interface and the second method is to interact with TestExec SL, which in turn controls hardware via its standard I/O interface.

Conclusion

This application note explains the basic features of a typical operator interface and serves as a good introduction to learning the development of an operator interface for beginners. For more advanced learning and customization of the operator interface, we recommend you to refer the *TypicalOpUI* sample operator interface which comes together with TestExec SL installation. It includes several layers of error checking, extensive management of the state of controls and also support of external hardware for manufacturing automation. You can use the sample source code almost immediately for your project with minimal modification.

References and Appendices

Testplan-level events	
AdviseClearReport	Indicates that report output is being cleared at the beginning of a run of the testplan. Typically triggers once at the beginning of a run of the testplan, even if the testplan will loop.
AfterTestplanLoad	Indicates that a new testplan has successfully been loaded.
AfterTestplanPause	Indicates that TestExec SL has entered a paused state, and returns the reason why the testplan paused.
AfterTestplanStop	Indicates that TestExec SL has halted, and returns the reason why the testplan stopped.
AfterTestplanUnload	Indicates the testplan has been unloaded.
BeforeTestplanBegin	Indicates that a pass through the testplan sequence is about to begin. Also occurs when testplan execution resumes via calling the Continue method.
BeforeTestplanLoad	Triggered in response to a LoadTestplan method.
ReportMessage	Indicates a new "block" of report output has arrived.
Test-level events	
AfterTestDone	Triggered after the current test finishes executing.
BeforeTestBegin	Triggered before the next test in the testplan begins executing.
ReportMessage	Indicates a new "block" of report output has arrived.

Miscellaneous events	
AdviseUpdate	Triggers to let the operator interface update its display in a way that does not interrupt the critical timing of a testplan.
UserDefinedMessage	Triggers to notify the operator interface that a user-defined message has arrived.

For more information and other literature, please go to **www.agilent.com/find/testexec** and **www.agilent.com/find/ftforum**.

www.agilent.com
www.agilent.com/find/testexec
www.agilent.com/find/ftforum

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

Agilent Direct

www.agilent.com/find/agilentdirect
Quickly choose and use your test equipment solutions with confidence.

LXI

www.lxistandard.org
LXI is the LAN-based successor to GPIB, providing faster, more efficient connectivity. Agilent is a founding member of the LXI consortium.

Remove all doubt

Our repair and calibration services will get your equipment back to you, performing like new, when promised. You will get full value out of your Agilent equipment through-out its lifetime. Your equipment will be serviced by Agilent-trained technicians using the latest factory calibration procedures, automated repair diagnostics and genuine parts. You will always have the utmost confidence in your measurements. For information regarding self maintenance of this product, please contact your Agilent office.

Agilent offers a wide range of additional expert test and measurement services for your equipment, including initial start-up assistance, onsite education and training, as well as design, system integration, and project management.

For more information on repair and calibration services, go to:

www.agilent.com/find/removealldoubt

Product specifications and descriptions in this document subject to change without notice.

Microsoft is a U.S. registered trademark of Microsoft Corporation.

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:

www.agilent.com/find/contactus

Americas	
Canada	(877) 894-4414
Latin America	305 269 7500
United States	(800) 829-4444

Asia Pacific	
Australia	1 800 629 485
China	800 810 0189
Hong Kong	800 938 693
India	1 800 112 929
Japan	0120 (421) 345
Korea	080 769 0800
Malaysia	1 800 888 848
Singapore	1 800 375 8100
Taiwan	0800 047 866
Thailand	1 800 226 008

Europe & Middle East	
Austria	01 36027 71571
Belgium	32 (0) 2 404 93 40
Denmark	45 70 13 15 15
Finland	358 (0) 10 855 2100
France	0825 010 700*
	*0.125 €/minute
Germany	07031 464 6333
Ireland	1890 924 204
Israel	972-3-9288-504/544
Italy	39 02 92 60 8484
Netherlands	31 (0) 20 547 2111
Spain	34 (91) 631 3300
Sweden	0200-88 22 55
Switzerland	0800 80 53 53
United Kingdom	44 (0) 118 9276201

Other European Countries:
www.agilent.com/find/contactus

Revised: March 24, 2009

Agilent Technologies