Release Notes
TYX TestBase
Version 2.5.3
8 January 2004

1. Overview

This is a maintenance release that corrects several problems.

The release includes the following main items:

- TestBase installation package, including a complete set of sample files, documentation and tutorial slides
- Redistributables, including:
  - TYX License Manager
  - Third-party run-time engines
  - Instrument drivers needed for running some of the samples
- Installation Guide

2. Detailed Description

2.1. Critical items

2.1.1. Configuration of Functional Test Controller (FTC) execution reports

When generating FTC Execution Reports that include Test Information and Parameter Values, it is strongly recommended to use separate files, instead of appending information to the same file. If information is appended to the same file, this file will eventually become very large and this will increase the execution time. Use the FTC menu **Tools | Options | UI_FT Specific** to configure the generation of execution reports.

2.1.2. Security setting for Visio 2002

Visio 2002 displays a warning message each time it opens a flowchart that contains macros. Because the TestBase IDE makes use of Visio macros, this message will appear during IDE operation. To disable the display of the warning message, open the Visio application, select the menu entry **Tools | Macros | Security** and choose the security level **Low**.

2.1.3. Environment variable setting for LabVIEW Adapter

The LabVIEW Adapter loads the LabVIEW engine when it is loaded by TestBase, and unloads the engine when it is unloaded. It appears that the repeated unloading and reloading of the LabVIEW engine causes unreliable operation. Thus, for a reliable operation of the LabVIEW test procedures it is recommended to set the value of the environment variable **TB_TCA_UNLOAD** to 0 (zero). See the Help System for details.

2.1.4. Error when generating Diagnostic Procedure HTML Reports

When generating an HTML report for a diagnostic strategy, you may get the error message "Unable to export CFD". The report files containing the Visio diagram are not generated. The report file containing the text description of the diagnostic strategy is generated, but its hyperlinks will not operate.

This problem occurs if Visio Professional was installed on the system but its HTML export feature was never used. To fix the problem, open an existing flowchart in Visio or create a new one (menu **File | New**), then export it as HTML (menu **File | Save As**, select file type "HTML Files", select a temporary folder and press **OK**).

2.1.5. Operating system compatibility

The product works properly with Windows NT 4.0 SP 6, Windows 2000 SP 2 and Windows XP.

2.1.6. Internet Explorer compatibility

The product works properly with Microsoft Internet Explorer 5.5 and 6.0. **It does not work with Internet Explorer 5.0**.

2.1.7. Microsoft Office compatibility

The display of Excel reports works properly with Microsoft Office 2000 Service Release 1 and Microsoft Office XP.

2.1.8. Microsoft Visio compatibility

The product works properly with Microsoft Visio 2000 and Visio 2002.

2.1.9. Oracle compatibility

The MTI Controller distributed with the current version of the product was tested with the following combinations of versions :

- Oracle 8.1.5 and 8.1.7, under Windows NT 4.0
- Oracle 8.1.5 and 8.1.7, under Windows 2000
- Oracle 9.0.1.1, under Windows NT 4.0.

A known compatibility problem between the Oracle software and the Microsoft libraries used internally by TestBase may be fixed, for Oracle 8 versions, by configuring the system registry as described in the document Connectivity Issue with MDAC and Oracle8i.pdf. For Oracle 9, use the information provided for version 8.1 in the above document, performing the following replacements in the strings to be entered in the registry:

1. replace oraclient8.dll with oraclient9.dll
2. replace orasql8.dll with orasql9.dll

2.1.10. LabWindows/CVI compatibility

The Adapter for LabWindows/CVI works properly with the following versions: 5.0, 5.5 and 6.0. Minor changes to the sample test code must be performed in order to compile it under version 5.0. The TestBase installation package redistributes version 6.0 of the LabWindows/CVI Run Time Engine. You may obtain other versions of the Run Time Engine from National Instruments.

2.1.11. Case sensitivity of Outcome comparisons

When a Test block is followed by decision blocks, the compiler compares the outcome values tested by the decision blocks against the set of possible outcomes returned by the called Test procedure. In previous releases, the above comparison was case insensitive. For consistency with other comparisons in TestBase, the comparison algorithm was changed to case sensitive, starting with release 2.4.0. This change does not affect the operation of existing projects and does not require their recompilation.

2.1.12. Rebuilding samples of Custom Data Type Editors and samples using Custom Data Type Editors

When rebuilding the Custom Data Type (CDT) Editor samples (or a modified version of these samples), overwrite the existing .OCX file (also included in the distribution). Do not delete the existing file before generating the new one. Overwriting the original file preserves the version used by Visual Basic to record binary compatibility. If you generate a new .OCX file (without overwriting), you will need to re-build all test procedures using that CDT Editor.

To rebuild the LabWindows/CVI test procedure samples that use Custom Data Type (CDT) Editors (or a modified version of these samples), you must first re-generate the wrapper files for the CDT Editor components. When re-generating, overwrite the original files.

2.2. Known Limitations

2.2.1. TestBase IDE

The "Undo" command used during flowchart editing does not operate properly in some situations. It is recommended to avoid using it.

The "Undo" command for the addition of an Off-Page Reference block does not delete the "pair" block that was automatically appended. Workaround: delete manually the "pair" block.

When running a test strategy in debug mode, with execution stopped between steps, if the user clicks on a different Control Flow Diagram (flowchart) then clicks the **Run**, **Step**, **SimulatedStep** or **SimulatedStepWithUI** buttons, an error message may be displayed: "Automation Error. Illegal to call out while inside message filter". This is normal. Press **OK**, click on the Control Flow Diagram that is currently debugged and continue using the IDE. The error indicated before has no impact on subsequent operation.

In some situations accelerator keys (Ctrl-O for **File | Open**, F5 for **Debug | Go**, etc.) do not work. Use menus or toolbar buttons instead.

A General Protection Fault occurs when closing the IDE or a Diagnostic Controller user interface after an Abort operation. Because it occurs only when the application is terminated, this behavior does not have harmful effects.

2.2.2. LabWindows/CVI Adapter

A limited subset of scalar data types is currently supported in the TestBase Support Library for LabWindows/CVI. Use direct access to VARIANT fields for other data types.

The execution of samples that use Custom Data Type Editors as ActiveX controls requires a LabWindows/CVI installation (the run-time engine is not sufficient).

2.2.3. PAWS Adapter

Array parameters are not supported.

Debugging of ATLAS TPSs launched from TestBase is not supported. Workaround: debug TPSs in standalone execution mode.

2.2.4. <u>LabVIEW Adapter</u>

The documentation for developing test procedures with LabVIEW does not cover the use of Custom Data Type Editors as ActiveX controls on LabVIEW panels.

2.2.5. <u>VEE Adapter</u>

The documentation for developing test procedures with VEE does not cover the use of Custom Data Type Editors.

2.2.6. <u>Permissions for installation</u>

The user performing the installation must have Administrator permissions for the Windows operating system. **After installing TestBase, this user must start the IDE once, to perform a Registry initialization.** After that, users with more restricted permissions can use TestBase modules. The user performing security administration from the IDE must have also Administrator permissions for the Windows operating system.

2.2.7. <u>MTI error on Global Parameters Values with the same name</u>

If Global Parameter Values with identical names are defined at multiple level of the Project tree and the storage of test results in MTI databases is enabled, an error will be generated at run-time. Workaround: avoid using identical names, until this problem is fixed.

2.2.8. <u>Custom Data Types</u>

The use of CDT Editors is currently not supported in test procedures developed with ATLAS and Agilent VEE.

The generic reports included with TestBase display the values of CDTs as follows:

- The Diagnostic Procedure report displays the generic text "<Value for Custom Data Type Editor with ProgID: ", followed by the ProgID of the CDT Editor.
- The sample Excel report "Sample_MTI_offline.xlt" displays the generic text "<Value for Custom Data Type Editor with ProgID: ", followed by the ProgID of the CDT Editor.

The values of Global Parameters with Custom Data Types are not stored in Oracle and Access MTI databases. You may retrieve these values from the Input Parameters to whom the Global Parameters are assigned.

2.2.9. <u>Diagnostic Procedure HTML Reports</u>

The internal links of text-based reports will not operate if the set of files generated for a report is moved to a different location in the file system or on a web server.

Graphical reports are not generated if Visio 2000 Standard is installed. This is due to a functional limitation of the Standard edition of Visio. The text-based reports are generated but their internal links will not operate.

Graphical reports are not generated if Visio 2002 is installed. This functionality will be added in a future release of TestBase. The text-based reports are generated but their internal links will not operate.

2.2.10. Generic Procedures

The release contains a Generic Procedure that is not described in the Help System. Provisional documentation is provided in Section 2.5.1 below.

2.2.11. Inconsistent behavior when compiling Diagnostic Procedures

An inconsistent behavior exists when compiling Diagnostic Procedures (DPs) in IDE. For details, see notes for Problem Report 03-126, in Section 2.4.

2.3 Enhancements

none

2.4 Problem Reports

This section addresses problem reports identified for release 2.5.2.

Increase in memory usage when running VEE test procedures

The amount of memory allocated to the VEE Adapter process increases when the environment variable **TB_TCA_UNLOAD** is 0 (zero) and the Diagnostic Controller is used for long periods of time without unloading the Project.

The integration between the VEE adapter and the VEE run-time engine was corrected, to properly free memory after execution.

IDE error when importing a Diagnostic Procedure that was not compiled

An error occurs when importing a Diagnostic Procedure where a Test block has no Outcome Value assigned.

The importer was enhanced, to be more tolerant to incomplete diagnostic procedures (i.e., diagnostic procedures that would generate compilation errors, if compiled). In a few situations when this was not possible due to functional constraints, the exporter requests the user to complete the design before performing the export. With these modifications, all Diagnostic Procedures and Test Procedures that are exported can be re-imported without errors. If a procedure that generates compilation errors is exported, the re-imported procedure will generate exactly the same errors.

Improper version of MDAC package distributes

Version 2.0 of MDAC is distributed with TestBase 2.5.2 (previous versions included MDAC 2.5). While this does not compromise TestBase operation, it is suspected to cause problems with NT 4.0 installations.

The current version re-distributes MDAC 2.5.

2.5 Additional Documentation

2.5.1 Generic Procedure “EvaluateExpression”

_EvaluateExpression – calculates the result of an expression

Input Parameters

- Arg0…Arg9 – expression arguments; have the type “double”
- Expression – string representing the expression to be evaluated; details are provided in the following

Expression syntax

The expression string can contain:

- immediate numeric constants, as described below
- operators, as described below
- functions, as described below
- the arguments Arg0 ... Arg9

Spaces can be freely introduced to visually separate expression elements. Parentheses '(' and ')' can be used to override operator precedence as necessary.

Numeric constants

You can specify constant numbers exactly like in programming languages, using the "e" or the "E" to separate mantissa from exponent and the point "." as decimal separator. Do not use the comma "," as decimal separator as it's used as parameter separator. Do not use the space " " as decimal separator as it could represent implied multiplication. Examples:

Correct	Incorrect
1	1,000,000
1.0	1,5
1e5	1e+0.5
1000000	1 000 000
1e+5	.05

Operators

Operator	Name	Description
& AND	logical AND operators	Combine multiple conditions formed using relational or equality expressions. Return the value 1 if both operands are nonzero; otherwise, return 0.
| OR	logical OR operators	Combine multiple conditions formed using relational or equality expressions. Return the value 1 if both operands are nonzero; otherwise, return 0.
NOT	logical negation	Returns the value 0 if its operand is nonzero; returns

		the value 1 if its operand is 0.
> < = >= <=	binary relational and equality operators	Compare their first operand to their second operand to test the validity of the specified relationship. Return 1 if the tested relationship is true and 0 if it is false.
<> >< !=	binary inequality operators	Compare their first operand to their second operand to test whether they are different. Return 1 if they are not equal, 0 if they are equal.
+	binary addition operator	Causes its two operands to be added.
-	binary subtraction operator	Subtracts the second operand from the first.
–	unary minus operator	Causes the following operand to undergo a change in sign.
*	binary multiplication operator	Causes its two operands to be multiplied.
/	binary division operator	Causes the first operand to be divided by the second.
^	binary power operator	Calculates the first operand raised to the power of the second operand.
!	unary factorial operator	Calculates the factorial of the operand.

The table below contains all the operators, in order of increasing evaluation precedence:

&, AND, |, OR
NOT
>=, <=, >, =, <, <>, ><, !=
+, -
-
*, /
^
!

Functions

Function	Description
ABS(x)	Returns the absolute value of x. Example: ABS(-5) returns 5.
ACOS(x)	Returns the arc-cosine of x. The return value is an angle specified in radians.
ASIN(x)	Returns the arc-sine of x. The return value is an angle specified in radians.
ATAN(x)	Returns the arc-tangent of x. The return value is an angle specified in radians.
CEIL(x)	Returns a floating-point value representing the smallest integer that is greater than or equal to x. Example: CEIL(4.3) returns 5.
COS(x)	Returns the cosine of x. The x value is an angle expressed in radians.
COSH(x)	Returns the hyperbolic cosine of x. The x value is an angle expressed in radians.
DIV(x,y)	Returns the quotient of the integer division between x and y. The x and y values must be integer values, otherwise unpredictable results will be generated.
MOD(x,y)	Returns the remainder of the integer division between x and y. The x and y values must be integer values, otherwise unpredictable results will be generated.
EXP(x,y)	Returns the exponential of x, that is to say the neperian constant *e* raised to the x-th power.
FLOOR(x,y)	Returns a floating-point value representing the largest integer that is less than or

	equal to x. Example: FLOOR(6.7) returns 6.
HYPOT(x,y)	Returns the hypotenuse of a right triangle with the given cateti x and y. Example: HYPOT(3,4) returns 5.
LN(x)	Returns the natural logarithm of x. If x is negative, returns an indefinite. If x is 0, returns INF (infinity).
LOG(x)	Returns the base-10 logarithm of x. If x is negative, returns an indefinite. If x is 0, returns INF (infinity).
MAX(x,y)	Returns the maximum of x and y.
MIN(x,y)	Returns the minimum of x and y.
RAND(x)	Returns a pseudorandom integer in the range 0 to x.
SIN(x)	Returns the sine of x. The x value is an angle expressed in radians.
SINH(x)	Returns the hyperbolic sine of x. The x value is an angle expressed in radians.
SQRT(x)	Returns the square root of x.
TAN(x)	Returns the tangent of x. The x value is an angle expressed in radians.
TANH(x)	Returns the hyperbolic tangent of x. The x value is an angle expressed in radians.

Internal number representation

The evaluation algorithm makes uses the type *double* (8 bytes) for internal number representation. The range for this data type is +/-1.7E308, with at least 15 digits of precision.

Examples

Arg1*(Arg1-Arg3)

1.2*SIN(Arg3)

<u>Output Parameters</u>

- Result – contains the expression evaluation result; has the type "double"

<u>Outcome Values</u>

- PASS – returned when the expression was successfully evaluated
- FAIL – returned when expression evaluation is impossible, although the expression's syntax is correct; this may occur, for example, in the case of a division by zero

<u>Error conditions</u>

An error condition is returned if the expression has an incorrect syntax. The following error messages may be returned:

Error Message	**Description**
Empty expression	This error arises when the expression string is empty, or composed of white spaces only
Wrong number of arguments	This error arises when you call a function with a bad number of arguments, for example when you try to evaluate the expression: sin(45, 2), when the sin() is known to accept one only argument.
Wrong use of reserved word	This error occur when you try to use a name of a function as if it were a variable or a constant or something else. In practice this is true whenever the parentheses which indicate a function call are not used. For example the following expressions generate this type of error:

	1 + sin (COS + TAN) sin 45
An argument was expected	This error usually occurs when an expression is truncated or when you forget an operand or an argument; this error is also generated when you forget a parenthesis at the end of a correct expression, rendering it an unusable sub-expression. For example the following expressions generate this sort of error: Sin((1+5)/(4+) (1+5)/(4+5
Undefined function	This error is reported when you use an identifier name (a name that can be used to name a variable) immediately before an open parenthesis: it seems to be a function name but it is not (probably because you misspelled the name, or perhaps because you intended to multiply a variable by something that is inside the parenthesis). Examples: x(5+4)+y - you intended: x*(5+4)+y sine(3.14) - you intended: sin(3.14)
Undefined identifier	This error is reported when you use an identifier name (a name that can be used to name a variable) different from Arg0 ... Arg9, probably because you misspelled the name.