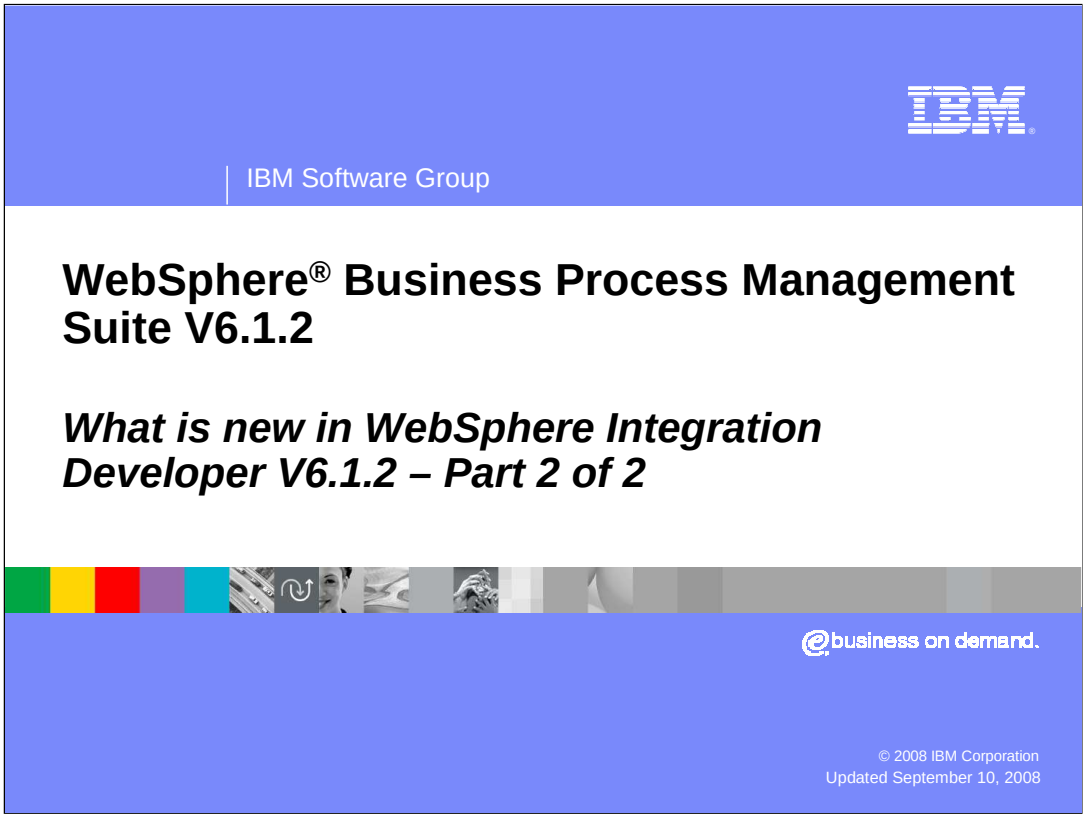


This presentation is the second part of a two part series to cover what is new in WebSphere Integration Developer version 6.1.2.

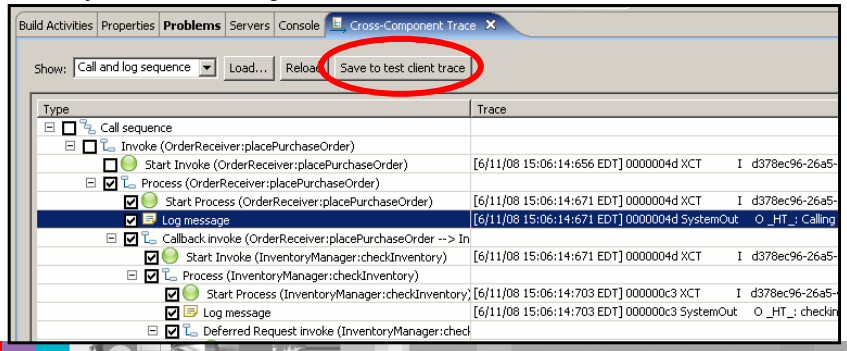
Agenda

- Problem determination enhancements
  - Troubleshooting enhancements
    - Cross-Component Trace - XCT
    - Fine-Grained Trace
    - Local debugging of XML Maps
    - TCP/IP monitor view
    - More quick fix items
  - Testing enhancements
    - Test maps from XML map editor
    - Wait option for test client
    - Test client automation
    - Emulate human tasks in test client
    - Data pool enhancements for test client
    - Test through SCA and Web service binding exports in test client
    - Testing XML Maps with test client
    - Component Test Explorer

This presentation introduces the second theme for WebSphere Integration Developer version 6.1.2 enhancements, problem determination enhancements. You will start with the troubleshooting enhancements sub-theme. First, you are introduced to Cross-Component Trace, referred to as XCT, followed by Fine-Grained Trace. You will finish out the troubleshooting sub-theme with learning about local debugging of XML Maps, easier access to the TCP/IP monitor view, and the addition of more quick fix items. The second sub-theme deals with testing business integration applications in WebSphere Integration Developer. You can now test XML maps directly from XML map editor. The integrated test client has added a wait option, can now be automated with scripts, can emulate human tasks, and has a more powerful data pool story. The test client can also test through SCA and Web service binding exports and test XML maps. Finally, you will be introduced to the Component Test Explorer which allows you to run and schedule tests and emulate SCA components and human tasks.

Troubleshooting enhancements

- Cross-Component Trace - XCT
  - Runtime enhancement for additional trace options
    - emits correlation information to logs
  - Cross-Component Trace view
    - GUI turns cross-component trace log into usable information
    - Ability to see server log correlated with SCA trace information in one view

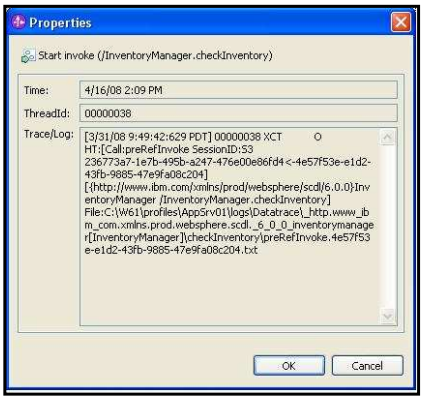


Cross component trace, or XCT, is really a runtime enhancement that provides additional trace options to the server log emitting correlation information in addition to input data. XCT logs make it possible to correlate or follow a session from entry into the system across components and across modules if wired with SCA bindings. Log information correlated to SCA flow across multiple synchronous, asynchronous, or long running instances/processes, multiple log files, and multiple threads. What this presentation covers is the Cross-Component Trace View in WebSphere Integration Developer that is the GUI to view these logs in an easy to read view.

In previous versions, finding a complex multi-component application or test case failure was time-consuming. The timestamp and thread ID was used for problem determination to correlate errors across components and the BPC Explorer was used to view flows and state. In addition, the problem would have to be re-run on the server to attempt reproduction of the issue. With the new cross-component trace view, you can examine and analyze cross-component traces from the server directly in WebSphere Integration Developer. You can perform analysis on snapshots of requests/response and SCA call data to see the error/exception and which component in SCA are they related to.

Troubleshooting enhancements

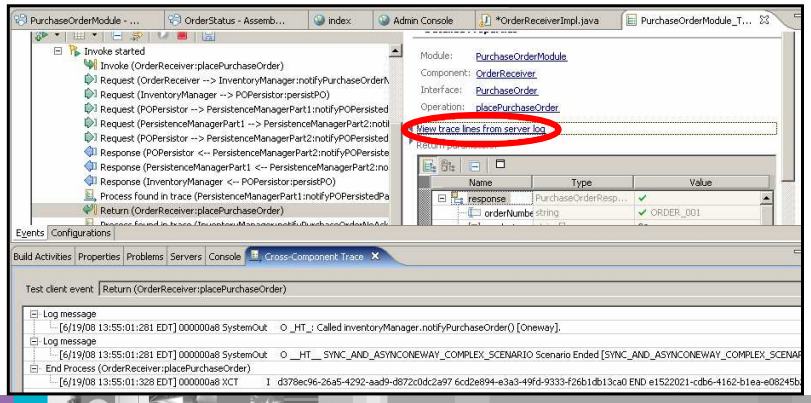
- Cross-Component Trace - XCT
  - Displays four pieces of information
    - Type of component and name
    - Timestamp
    - Thread-id
    - Entire trace/log message
  - Limitation
    - Cannot correlate across multiple calls to same BPEL instance
  - Ability to save and run in test client
    - Selections in XCT view automatically cascaded up to the parent invoke
      - Test client needs invoke to run
    - Select only certain sections of the logs to test
      - Focus on a part of process causing problems



The XCT view displays four pieces of information on trace/element record; type of component and name, timestamp, thread-ID and trace/log message. You can also right click on an entry to get a properties dialog, as shown in the screen capture on this slide. The current restriction is the cross-component trace log does not correlate across multiple calls to same BPEL instance. One of the most usable points for cross-component trace is the ability to save and run tests using the integrated test client. When selecting which components are causing the problem in the XCT view, the selection of check boxes are automatically cascaded up to the parent invoke because the test client needs an invoke to run a test. Without this automatic selection, you can save a cross-component trace to use in the integrated test client without the correct invokes, therefore causing errors once the test is run. Another nice point is you can select only certain sections of the logs to test so you can focus on a part of process causing problems and not be inundated with testing the entire process in the cross-component log.

Troubleshooting enhancements

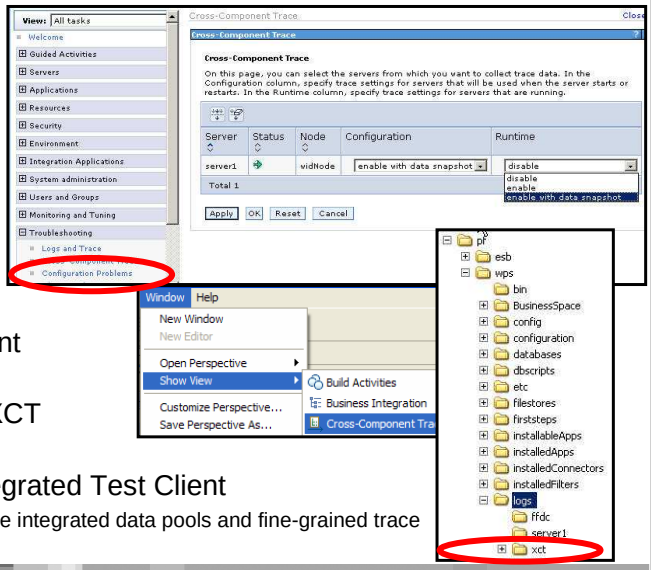
- Cross-Component Trace - XCT
  - Test client integration
    - Import XCT log into integrated test client
      - For more detailed examination
      - Logs can be reconstructed into re-runnable test scenarios



You can import the XCT log into the integrated test client for more detailed examination and the XCT logs can be reconstructed into re-runnable test scenarios. This saves you from having to re-deploy and re-run the application to try and solve the problem. Open test client and re-run with the same data, or change the input data on the fly. This slide shows a screen capture of a cross-component test log being run in the integrated test client and points out the ability to see trace lines in the server log directly from the integrated test client, in addition to all the request/response data and a view of the SCA flow.

Troubleshooting enhancements

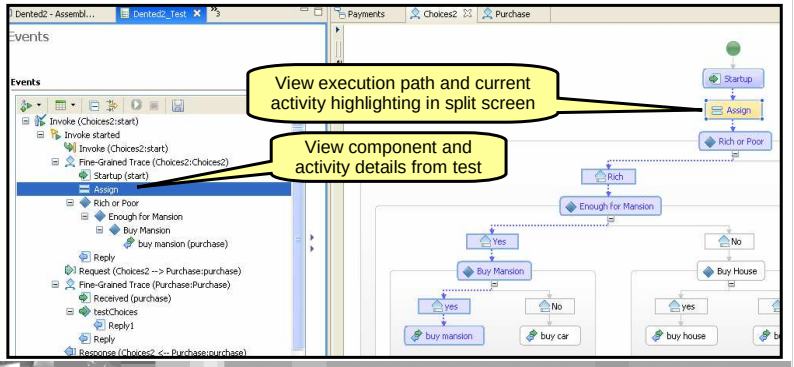
- Cross Component Trace - XCT
  - Used in development or test timeframe
    - Adds performance load to runtime
- To use
  - 1. Turn on XCT in Admin Console
  - 2. Run through test to re-create problem
  - 3. Open Cross-Component Trace View
  - 4. Load server logs into XCT View
  - 5. Save and re-run in Integrated Test Client
    - Tip: Use new test features like integrated data pools and fine-grained trace



Cross-Component Trace is meant to be used in development or test timeframe given that turning on the XCT logging does add a performance load to the runtime. In order to use XCT, first turn on XCT in the WebSphere Portal Server administration console. Select Troubleshooting -> Cross-Component Trace. Select the server to modify and choose from three options: Disable, Enable, Enable with data snapshot. Cross-Component Trace is disabled by default. Save the setting at next server start or immediate change to the runtime. This needs to be set for each server you want to turn XCT on for. Once you have XCT turned on for the server, run through your application to re-create the problem. Once the problem happens, open the Cross-Component Trace View in WebSphere Integration Developer and load the server logs into XCT View using the load button. Search local or remote machines for the server logs and XCT logs. For the WTE in WebSphere Integration Developer, you can find the server logs under WebSphere Integration Developer install directory/pf/profile name/logs/server1 and the XCT logs under WebSphere Integration Developer install directory/pf/profile name/logs/xct. Once you have the logs loaded into the XCT view, you can select which part of the log is causing the problem and click the “save to test client trace” button. In the integrated test client, you can save and re-run the tests and change the input on the fly. Tip. Use the new test client features like the integrated data pools and fine-grained trace.

Troubleshooting enhancements

- Fine-Grained Trace
  - Ability to see process details and execution path from test
    - BPEL activities
    - input and output variables
    - business state machine states
    - mediation flow component primitives
  - Split screen feature
    - Highlight execution path and activity

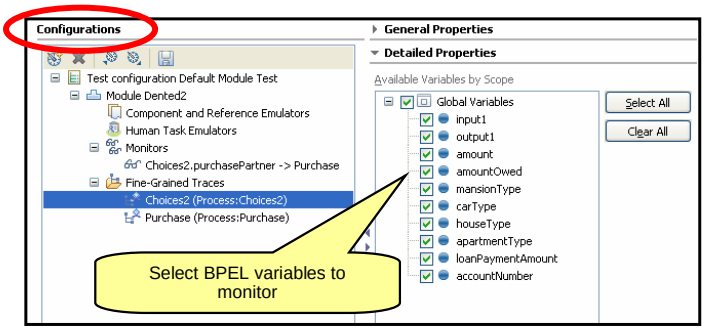


In previous versions, you used the integrated test client to track SCA events and restart server in debug mode to re-invoke test and step through application until problem was found. Fine-Grained Trace combines the two concepts of the test client and the debugger to overlay the appropriate editor with a visualization to view more process details and see the execution path. The benefit is the ability to immediately see where the exception happened in the flow in the development environment. Fine-Grained Trace allows you to quickly visualize detailed events performed for a single execution for a component in the test client. Currently, you can view BPEL activities, input and output variables, business state machine states, and mediation flow component primitives.

In addition, Fine-Grained Trace has a split screen feature that highlights the execution path and activities, as in the case of testing BPEL. In the screen capture, clicking on the assign activity in the events section of the test client highlights the assign activity in the BPEL editor in yellow. Notice the blue highlighting through the BPEL activities is the actual execution path of the BPEL when it was run. The path of events is highlighted in the business process editor, the mediation flow editor, and the business state machine editor. The appropriate editors are auto-opened by clicking on event in the test client.

Troubleshooting enhancements

- Fine-Grained Trace
  - Preferences
    - Highlighting
    - Split editors
    - Disabling/enabling
  - To use
    - Open test case
    - Click configurations tab in test client
    - Select Add button and then select Fine-Grained Trace
      - Select module and components to trace
    - Optionally select variables to trace in detailed properties panel
    - Run test and view activity / variable information by clicking on activity/mediation
      - Primitives and SMO values for mediations
      - Activities and variables for BPEL

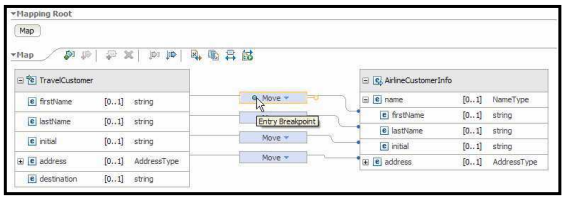


Preferences for highlighting, split-editors, and disabling fine-grained trace are located at Window > Preferences > Business Integration > Integration Test Client > Fine-Grained Trace.

To use Fine-Grained Trace, create a test case or open a previous test/execution trace and open the configurations tab. Click the add button in top right corner or right click in editor area and select Fine-Grained Trace, the module, and components to trace. Once created, you can select variables to track, as shown in the smaller screen capture in this slide.

Troubleshooting enhancements

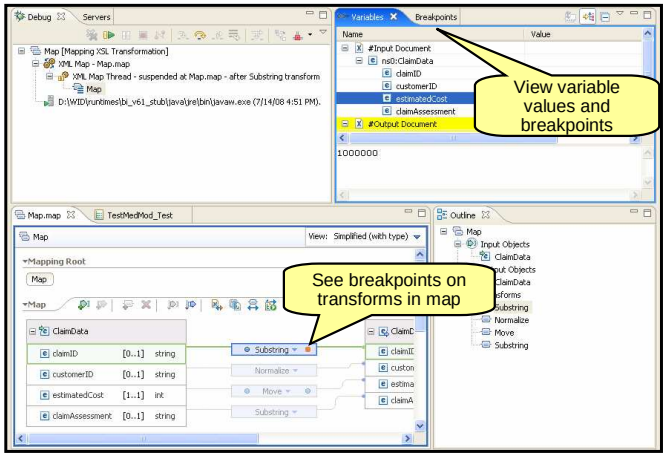
- Local debugging of XML Maps
  - Debug XML maps with local XSLT engine
  - No need to start runtime in debug mode
    - Speeds troubleshooting of incorrect output from xml map
  - View output of transform as it is being constructed
    - Output variables displayed in variables view
    - Includes in-line maps and sub maps
  - Set/unset entry or exit breakpoints on transforms
    - View XML before transform has been executed
    - View map result after completing transform



In previous versions, you had to start the server in debug mode and step through the mediation to prove that output of the XML map primitive is wrong. Now you can locally debug XML maps using a built-in XSLT engine without having to start the runtime in debug mode. This speeds troubleshooting of incorrect output from XML map by being able to debug locally and not using the runtime. Similar to debugging, you can step through transforms, in-line maps, and sub maps to see XML content as it is being constructed. The variables view allows you to see how the XML variables change as the map is being run. In addition, you can set local breakpoints before or after the transform to stop the debugger at a certain point so you can view XML before transform has been ran or view the map result after completing transform.

Troubleshooting enhancements

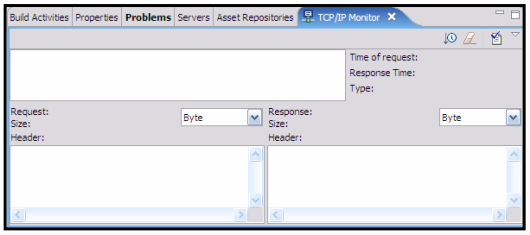
- **Local debugging of XML Maps**
  - To use
  - Right click on transform in map to set/unset breakpoints
  - Invoke test client
  - Choose debug instead of run
  - Opens debug perspective automatically
  - Options
    - Step from transform to transform
    - Step-into an inline map or sub map
    - Step-out of sub maps and inline maps
  - Breakpoints will not pause execution on server
    - Only when testing map locally



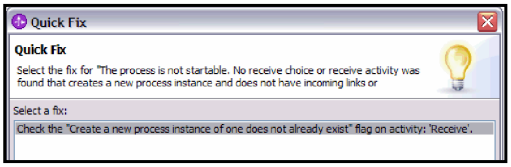
In order to use the local debugging, set or unset any breakpoints in the XML map. Then invoke the test client and choose debug instead of run. This action will open the debug perspective automatically and allow you to step from transform to transform, step-into an inline map or sub map, and step-out of sub maps and inline maps. One thing to note here is that breakpoints will not pause execution on server since this is testing the map with a local XSLT engine and not the runtime.

Troubleshooting enhancements

- **TCPIP Monitor**
  - Easier access to view
    - For monitoring Web service information
    - Window > Show View> TCP/IP Monitor
  - Can be used with component test
    - For testing Web services



- **More Quick Fix items added**
  - More BPEL problems easily fixed
  - Look for red X icon with light bulb icon on top

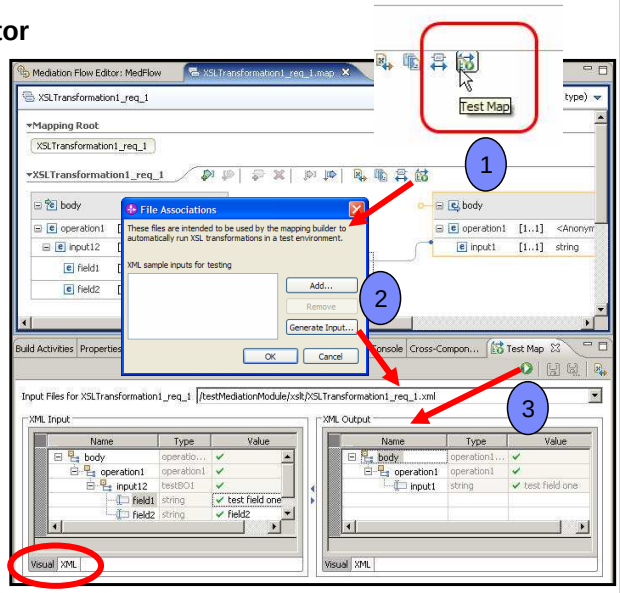


The TCP/IP Monitor is not a specific WebSphere Business Process Management application, but a simple server that monitors all the requests and the responses between a Web browser and server. To start monitoring TCP/IP, you will fill out the local monitoring port, host name, port of the server, choose HTTP or TCP/IP, and click to start the monitor. For version 6.1.2 of WebSphere Integration Developer, you can now access the TCP/IP monitor view easier by selecting Window > Show View > TCP/IP Monitor. This view can be used to help with Web service debugging along with component test on Web service exports.

Several quick fixes have been added to WebSphere Integration Developer V6.1.2 for BPEL validation problems. Quick fixes can be seen in the problems view or when viewing a red X icon in WebSphere Integration Developer. A quick fix is available for an error if the red X icon has a light bulb icon added on top of it. The quick fix will provide suggestions for problem resolution, allow you to choose which solution fixes the problem, and will automatically solve the problem with a click of the OK button.

Testing enhancements

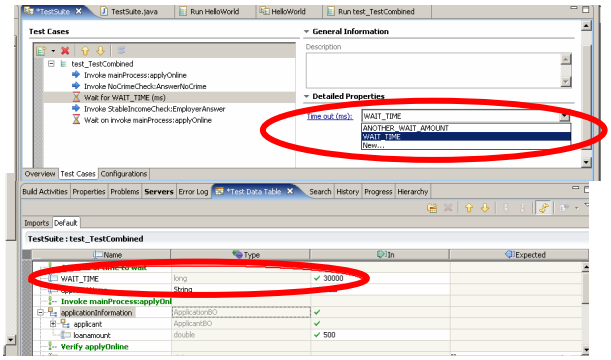
- **Test maps from XML map editor**
  - Iteratively test XML maps while developing applications
  - 2 new buttons on map editor
    - Test Map
    - Generate XSLT script
  - Generate or add input file to run data through map
  - Test Map View
    - Input file used for test
    - input XML
    - output XML
    - run button – run
  - 2 ways to view XML input and output
    - Visual and XML tabs
  - Save input XML
    - for later use and editing
  - Test is local, runtime not used



In version 6.1.2, you can now test maps from within the XML map editor. This enhancement allows you to iteratively test XML maps while developing applications. Two new buttons on map editor have been added to gain access to this new feature; the test map button and the generate XSLT script button. Either supply an input XML file with data to run through the test map facility or use the tools to generate your own input file. Input XML files are used by the mapping builder to automatically run the XSL transformations. Once the input file is set, the new test map view opens to display which input file is being used for the test, an area to show the actual input XML, and another area to show the output XML from running the map. The way to run the map is by using the run button, or green circle with white triangle, in the test map view. There are two ways to view XML input and output. The visual and XML tabs on each area will display the content differently. The visual tab, as shown in the screen capture, will display the XML data in GUI table. The XML view displays the content in its original form. Changes under one tab are reflected in the other and visa versa. A nice feature here is the ability to save the input XML for later use by you or a team of developers. Another nice feature is the fact the test is local. This allows you to quickly test changes to XML maps or different input content without having to build, deploy, and run the application to test.

Testing enhancements

- Wait option for test client
  - Previous versions
    - No way to pause testing
  - Now pause for a certain amount of time before continuing execution
    - Force a time-out
    - Cause a human task to escalate
    - Wait for an asynchronous process to finish
    - Reflect a real-life service level agreement in the test environment



  - Add “wait for time” to test client
    - Adds variable to hold a milliseconds value of type long
    - Represents how long to wait before continuing execution

In previous versions, the test client cannot wait or pause for a specified amount of time before proceeding to the next statement. In version 6.1.2 a wait option has been added to allow test cases to pause for a certain amount of time before continuing execution. The typical use case is to add a wait to force a time-out, cause a human task to escalate, wait for an asynchronous process to finish, or reflect a real life service level agreement in the test environment. To use the new wait feature, right click or select the add button in the test cases editor and choose “wait for time”. This action creates a variable to hold a milliseconds value of type long to represent how long to wait before continuing execution. The wait can be defined to be before or after an invoke.

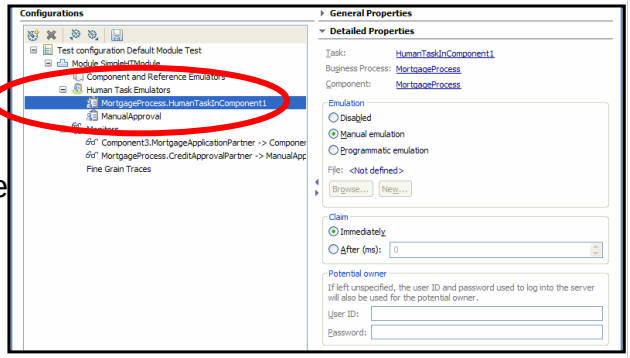
Testing enhancements

- **Test client automation**
  - Automatically run test case from a command prompt
  - Tests driven using ANT scripts
  - ANT scripts call Servlet to remotely run test cases in test project
    - Retrieve, build, deploy, and run test cases
  - Run independently or in headless mode
    - Can run ANT scripts in headless mode with batch/shell scripts
    - Allows tests to be run as part of a build script
    - Can run nightly or during low usage times
  - XML file describing results is produced at end
    - Can use to display custom build/test results
  - Limitations
    - Creation of the scripts are by the user
    - Automation assumes the test cases do not change
      - Edit scripts appropriately
  - Example
    - Integration with source control systems like CVS using ANT

Test automation has been added in WebSphere Integration Developer version 6.1.2. This enhancement automatically will run a test case from a command prompt using ANT scripts. The ANT scripts call Servlet to remotely run test cases in test project in order to retrieve, build, deploy, and run test cases. You can run the ANT scripts independently or in headless mode with batch scripts on Windows operating systems or shell scripts on Linux operating systems. This allows tests to be run as part of a build script and can be used as part of a nightly build or during low usage times. At the end of the test an XML file describing results is produced. This resulting XML file is used to display custom build/test results. The current limitations as of version 6.1.2 are the creation of the scripts is by you and the automation assumes the test cases do not change. If the test cases have been changed in WebSphere Integration Developer, update the scripts appropriately. The example used in the screen capture on this slide is using integration with source control systems like CVS using Ant scripting. You will need to code ant scripts to work with CVS. However, the point here is you can pull your module and test case projects from CVS, and then deploy and run the test cases against those projects using a combination of ANT and CVS scripting.

Testing enhancements

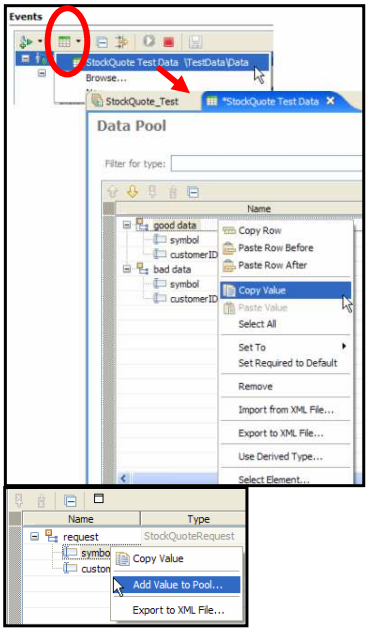
- Emulate Human Tasks in test client
  - Claim, enter data, and complete humans tasks
  - No need to claim tasks manually with a client
    - like BPC explorer
  - For in-line or stand-alone human tasks
  - Claim immediately or delay
    - Force escalation or replicate real life scenario
  - Limitation
    - must write code to programmatically emulate human tasks for component test
    - can use manual or programmatic in test client



In previous versions of WebSphere Integration Developer you had to create UI for Human Task (or use BPC Explorer) to claim, enter data and complete tasks for testing. Using the component test and test client features, you can now emulate and claim human tasks that are in the testing module. In the test client, you can either manually or programmatically emulate a human task. However, in the component test feature, you must write code to programmatically emulate a human task. This feature can be used with both in-line or stand-alone human tasks. To use, open the test client and click on configurations tab similar to setting fine-grained trace. Here you can add, define, and edit human task emulators. Options include disabling the human task emulator or choosing between manual or programmatic emulation. There is a file option for providing the code for programmatic emulation. In addition, there is a claim option which allows you to claim the task immediately or wait for a certain amount of milliseconds before continuing execution. Finally, you can set the potential owner of the human task in the test client to replicate a real life scenario.

Testing enhancements

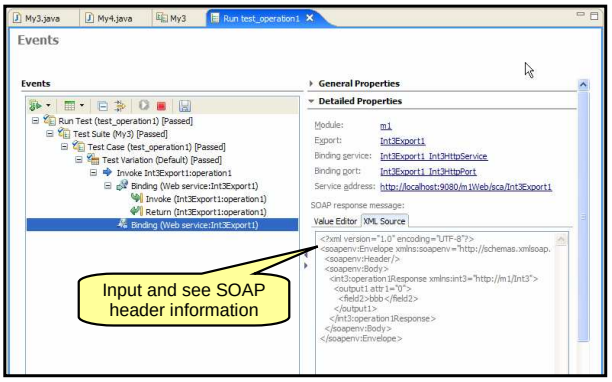
- Data Pool enhancements for test client
  - Data pools now saved in workspace
    - Instead of one data pool being saved in metadata
    - Old data pools migrated into workspace on first touch
      - Migration wizard automatically opens
    - Data pools imported into a test project
    - Can export data pool with project
    - Can check data pool into source control
    - Participates in refactoring
    - Can have multiple data pools
  - First class data pool editor
    - Multiple data pools can be open
    - Values, rows can copied or imported
  - Best Practice
    - Place test artifacts in test projects
  - Access
    - Data pool button at top of events tab of test client



In previous versions of WebSphere Integration Developer, you only had one data pool that you had to find in the comp test plug-in metadata in the workspace and cannot easily share between a team. The integration test client now supports the use of multiple data pools and has a first class editor. You are no longer limited to one data pool that is in a hidden in the workspace metadata. Data pools are now workspace artifacts, which means that you can use them in business object refactoring, export them with a project, or check them into a source code repository. In addition to being able to add values to, or use values from, one or more data pools, you can also use the new data pool editor to edit saved values, rearrange values, or copy multiple values between different data pools. Now teams of developers can share and update data pools easier between the team. To use a team member will save their input data to a file in the test pool project. Another team member will retrieve data to start a test case and use a source control system like CVS or Clear Case to open the Test Pool project and use the data pool. For those who used data pools previous to WebSphere Integration Developer version 6.1.2, you are prompted when first opening the workspace with the old data pool that the old data pool will now be imported into the workspace. Preferences for data pools are held in Window > Preferences > Business Integration > Integration Test Client. To access, click the data pool button at the top of the events tab in the test client.

Testing enhancements

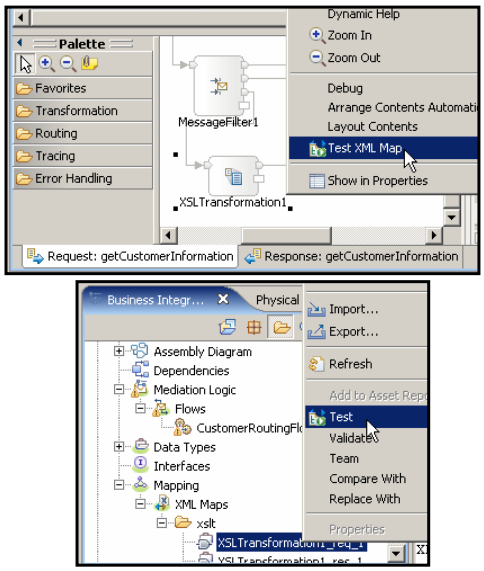
- Test through SCA and Web service exports in test client
  - Component test cases can start with a Web service or SCA export
    - Invoke Web service binding exports and SCA binding exports
  - Save export invocations in test client as test case
  - Able to input and see SOAP header information for Web service binding
  - Able to input and see interface/business object information for SCA binding



Before version 6.1.0 you cannot use the test client to test Web service or SCA exports. You would have to turn on debug and then call the module with your own client. As of version 6.1.0 you can test a Web service export with the test client by supplying the whole SOAP Envelope as input data. In addition, version 6.1.0 also introduced Component Test, the ability to test SCA components with test cases and test suites. Unfortunately, you cannot test a Web service export with test cases. This means if you start by testing through the Web service edge with the test client, you cannot right click to turn that into a test case. It also means you can't write test cases for modules dependent on soap headers. In version 6.1.2, test cases can start with a Web service or an SCA export. That is, component testing now enables you to invoke Web service binding exports and SCA binding exports and save export invocations in test client as a test case. For SCA binding exports, you would enter the parameter values in the variables specified in the parameters section. For Web service binding exports, enter the parameter values in the SOAP message. The input, output, and fault parameters are wrapped in a SOAP message. SOAP type variables are in each parameter type section regardless of how many parameters are defined in the operation. The SOAP parameter sections are request, response, and exception. When you select a Web services binding export in the test suite editor, the test data table is populated with a serviceAddress variable. This enables you to change the service address used to invoke the Web service and allows you to route the invocation to a different monitoring tool, such as the TCP/IP Monitor.

Testing enhancements

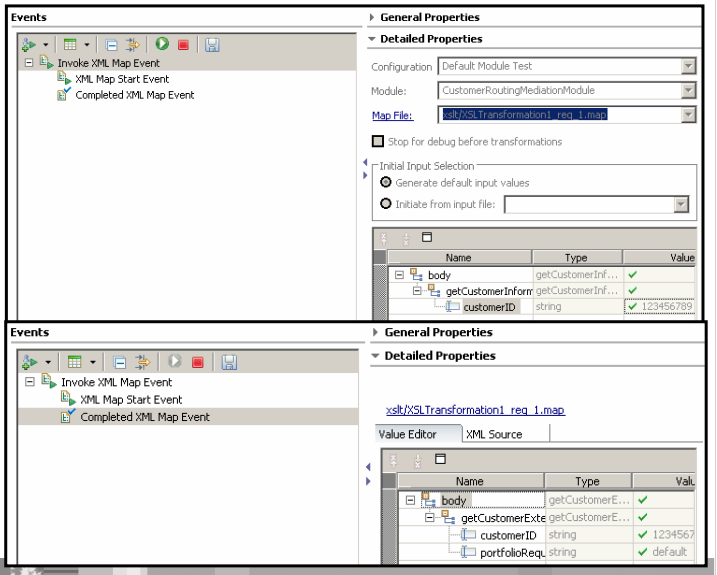
- **Testing XML Maps with Test Client**
  - Option for testing XML maps from XML Map editor
  - Access
    - Right click XML map in mediation flow editor
      - Test XML Map
    - Right click XML map in business integration view
      - Test
    - Right click .map file in physical resources view



In addition to testing a map from within the XML map editor, you can use the test client to test transformations on XML maps. Previously you needed to test the whole mediation flow component on the server using the test client to accomplish this task. To open the test client for an XML map, right click XML map in mediation flow editor and select “Test XML Map”. You can also right click the XML map in business integration view or right click a .map file in physical resources view and select “Test”. Again, this enhancement is for developers actively developing and changing maps while wanting to test XML maps using several different data inputs.

Testing enhancements

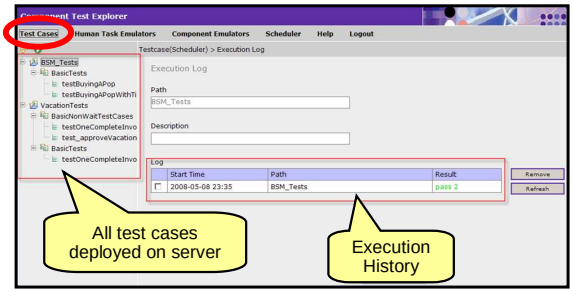
- **Testing XML maps with test client**
- Internal xslt engine used
  - No server needed
  - Quick tests
- Load input file
  - Able to edit, resave
- 2 ways to view output
  - Value editor and XML source
- Organize saved tests in component test project

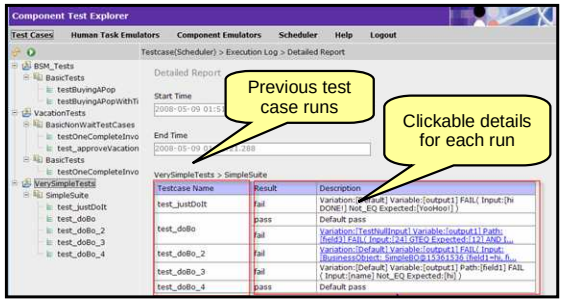


Though this feature is in the test client, the server is not used to run the test. An internal XSLT engine is used instead in order to speed testing without having to start the server. In addition, you can rerun the test sessions and save the data in a test project to reuse for later testing. The advantage to using the test client to test XML maps over running tests from the XML map editor and test map view is the ability to keep tests organized in a component test project.

Testing enhancements

- **Component test explorer**
  - A Web application
  - Allows testers to define, manage, and run test cases already deployed without WebSphere integration developer
  - Deployed test cases can be queried, executed, and scheduled
    - View results of previous runs, re-run test
  - Define and manage global emulators
- **Test cases tab**
  - Easily view component test projects, test suites, test cases on server
  - Able to select and run test cases
  - View execution history results
    - Pass, fail, exception
    - Result text is clickable
      - Opens detailed report of run
    - Detailed report
      - Clickable details for more information

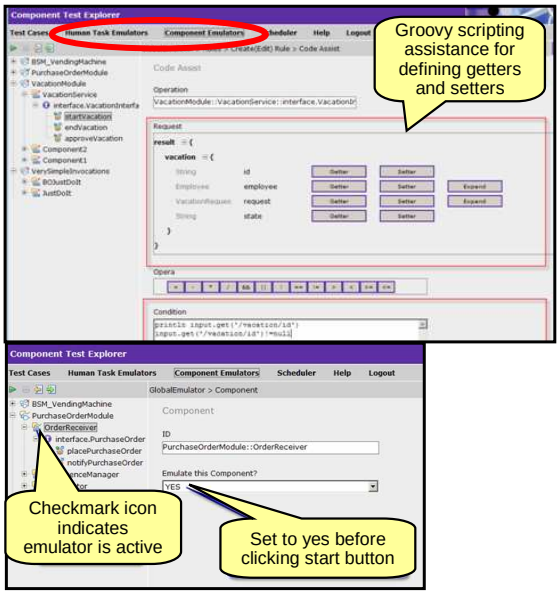




In previous versions of WebSphere Integration Developer, testers used WebSphere Integration Developer to load all module, library and test case projects to run on a test server and needed WebSphere Integration Developer to test the applications once deployed to the test server. Testers are now able to define, manage, and run test cases deployed on a test server without WebSphere Integration Developer. The component test explorer is a Web application, or .ear file, that runs on the server where component test is located. You can use normal methods or ANT tasks to deploy all module, library and test cases to a test server. Deployed test cases can be queried, executed, and scheduled from the component test explorer. Detailed results from previous test runs can be viewed and re-run. In addition, you can define and manage global emulators. From the test cases tab you can easily view component test projects, test suites and test cases currently deployed on the server. This gives you the ability to select and run test cases and view execution history results. The results of pass, fail, or exception are displayed in colored text and are clickable to open a detailed report of the test run. The detailed report even has more clickable details for more information. The limitation is that all modules, libraries and test cases need to be deployed to the test server before any testing work can be done and the component test explorer application needs to be deployed on the testing server.

Testing enhancements

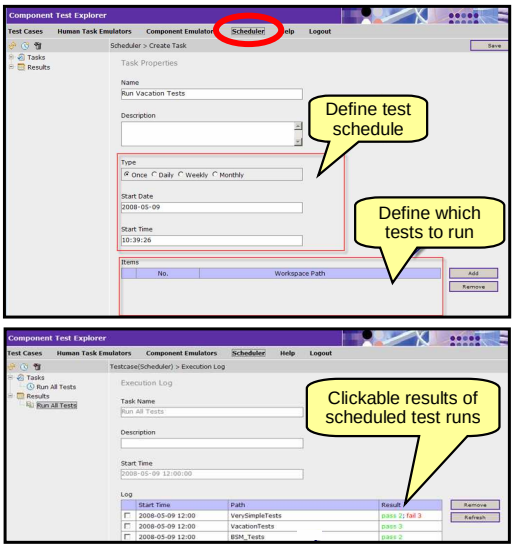
- **Component test explorer**
  - **Emulators tabs**
  - Define and manage two types of emulators
  - SCA component
    - Emulate any SCA component
  - Human task
    - Emulate stand-alone or in-line human tasks
    - Automatically claim global human tasks
  - Groovy assistance to emulate interface information
    - Request parameters
    - Response parameters
  - Checkmark icon
    - Indicates emulator is active
    - Must set emulator to yes beforehand
    - Then click start button
  - Start and stop buttons
    - Start button
      - All emulators set to yes are activated
    - Stop button
      - Stops all emulators
  - Import and export buttons
    - Import or export xml



You can define and manage two types of emulators in the component test explorer Web application; SCA component and human task. This means that you can emulate any SCA component or human task. For human tasks, you can emulate stand-alone or in-line human tasks and automatically claim any global human tasks. Groovy scripting assistance is provided in the Web application to emulate interface information for request parameters and response parameters. This makes it easier to define parameters to emulate and it comes with its own validation. To see if an emulator is active, check the left list of test artifacts for the checkmark icon. This indicates the emulator is active. To make the emulator active you must set emulator to yes beforehand, as denoted in the bottom screen capture, then click the start button. The start and stop buttons are located at the top-left of the page. The start button will activate all emulators set to yes, and the stop button stops all emulators. In addition to the start and stop buttons, there are the import and export buttons. Use these buttons to import or export XML.

Testing enhancements

- **Component test explorer**
  - Scheduler tab
    - Accessible from scheduler tab
    - Schedule test cases for execution
      - Define tests to run
        - Add and remove tests deployed to server
      - Define when to run tests
        - Once, daily, weekly, monthly
        - Start and end date
  - Two ways to access
    - Launch from servers view
      - Right click servers view > launch > component test explorer
      - Automatically installs Web application if not installed
      - Recommended for first time use
    - Launch Web browser and use URL
      - https://<host name>:<port>/ComponentTestExplorer/
      - For use without WebSphere Integration Developer



The ability to schedule tests to run is accessed from the scheduler tab in the component test explorer. On this page you define which tests to run by adding and removing tests that are deployed to the server. You will also need to define when to run tests. Options are immediately, once, daily, weekly, and monthly. In addition, you will need to provide a start and end date.

There are two ways to access the component test explorer. First, you can launch from the servers view inside WebSphere Integration Developer by right clicking the servers view > Launch > Component Test Explorer. This action automatically installs Web application if not installed and is recommended for first time use. Afterwards you can bypass WebSphere Integration Developer and launch the Web browser and use the Web application's URL. If installed by WebSphere Integration Developer, the component test explorer application is placed at https://<host name>:<port>/ComponentTestExplorer/.

Summary

- Problem determination enhancements
  - Troubleshooting enhancements
    - Cross-Component Trace - XCT
    - Fine-Grained Trace
    - Local debugging of XML Maps
    - TCP/IP monitor view
    - More quick fix items
  - Testing enhancements
    - Test maps from XML map editor
    - Wait option for test client
    - Test client automation
    - Emulate human tasks in test client
    - Data pool enhancements for test client
    - Test through SCA and Web service binding exports in test client
    - Testing XML Maps with test client
    - Component Test Explorer

This presentation introduced the second theme for WebSphere Integration Developer version 6.1.2 enhancements, problem determination enhancements. You started with the troubleshooting enhancements sub-theme by being introduced to Cross-Component Trace, referred to as XCT, followed by Fine-Grained Trace. You finished out the troubleshooting sub-theme with learning about local debugging of XML Maps, easier access to the TCP/IP monitor view, and the addition of more quick fix items. The second sub-theme dealt with testing business integration applications in WebSphere Integration Developer. You can now test XML maps directly from XML map editor. The integrated test client has added a wait option, can now be automated with scripts, can emulate human tasks, and has a more powerful data pool story. The test client can also test through SCA and Web service binding exports and test XML maps. Finally, you were introduced to the Component Test Explorer which allows you to run and schedule tests in addition to emulate SCA components and human tasks.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WPIv612_WIDWhatsNew_2.ppt

This module is also available in PDF format at: ../WPIv612_WIDWhatsNew_2.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.