Contents

Preface .......................................................................................................................... 7
- Documentation Conventions .................................................................................. 8
- Related Publications .............................................................................................. 8
- Customer Support .................................................................................................. 8
- Help Us to Serve You Better .................................................................................. 9
- User Feedback ...................................................................................................... 11
- Information Builders Consulting and Training ...................................................... 11

1. Defining Troubleshooting and Debugging Strategies ............................................. 13
- Troubleshooting and Debugging Overview ............................................................ 13

2. Using iWay Service Manager Diagnostics, Tracing, and Logging Facilities .......... 15
- Running in a Command Shell ................................................................................ 15
- Diagnostic Commands .......................................................................................... 17
- Troubleshooting on Windows ................................................................................. 19
- Performing Diagnostic Functions ......................................................................... 20
  - Log Settings ...................................................................................................... 20
  - Trace Settings ................................................................................................. 23
- Measurements and Statistics .............................................................................. 28
  - Memory ........................................................................................................... 28
  - Deadlocks ...................................................................................................... 30
  - Statistics ........................................................................................................ 30
  - Emitted Statistics Information ......................................................................... 32
  - Tips .................................................................................................................. 34
- Using the Log Viewer ............................................................................................ 35
- Creating a Diagnostic Zip ...................................................................................... 36

3. Identifying Available Services for Troubleshooting and Debugging ..................... 39
- Activity Log Entry Service (com.ibi.agents.XDXALogEvent) .............................. 39
- Catch Service (com.bi.agents.XDCatchAgent) .................................................... 41
- Fail Service (com.ibi.agents.XDFailAgent) .......................................................... 43
- QA Service (com.ibi.agents.XDQAAgent) ............................................................. 44
- Trace Message Writer Service (com.ibi.agents.XDTraceAgent) ......................... 46
4. Identifying Available Commands and Functions for Troubleshooting and Debugging ................................................................. 49
   Using the Testfuncs Tool ...................................................................... 49
   Using the Testxpath Tool .................................................................... 50
   Using the Flow Command ................................................................... 51
   Using the Line Command .................................................................... 53
   Using the _eval() Function .................................................................. 54

5. Creating and Using a Remote Command Console ....................................... 55
   Remote Command Console Overview .................................................. 55
   Creating a Remote Command Console ................................................. 56
   Connecting to a Remote Command Console ......................................... 63
   Using a Telnet Client ......................................................................... 64
   Remote Only Commands ..................................................................... 66
   Telnet Scripting Example .................................................................... 66

6. Using Event and Startup Process Flows ................................................ 69
   Event Process Flows ......................................................................... 69
   Server Startup .................................................................................. 70
   iWay Business Activity Monitor Database Loss of Access .................... 71
   Channel Startup Failure ................................................................... 71
   Retry Expired .................................................................................. 73
   Failed ReplyTo ............................................................................... 74
   Send to Dead Letter ......................................................................... 76
   Parse Failure ................................................................................... 78
   Startup Process Flow ......................................................................... 79

7. Recommended Third-Party Tools for Troubleshooting and Debugging ............ 81
   JConsole .......................................................................................... 81
   JVM Startup Options .......................................................................... 82
   SoapUI ............................................................................................. 83
   KeyTool IUI ...................................................................................... 83
   Operating System Commands ............................................................ 84
   Wireshark .......................................................................................... 84
Preface

This documentation describes the iWay Service Manager (iSM) tools and facilities currently available for troubleshooting and debugging purposes. It is intended for developers who must integrate new XML-based applications seamlessly with existing enterprise transactions, procedures, and application packages.

Note: This Release 7.0.x content is currently being updated to support iWay Release 8.0.x software. In the meantime, it can serve as a reference for your use of iWay Release 8. If you have any questions, please contact Customer_Success@ibi.com.

How This Manual Is Organized

This manual includes the following chapters:

<table>
<thead>
<tr>
<th>Chapter/Appendix</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Defining Troubleshooting and Debugging Strategies</td>
</tr>
<tr>
<td></td>
<td>Provides an introduction to iWay Service Manager (iSM) troubleshooting and debugging facilities, and outlines key strategies that can be used.</td>
</tr>
<tr>
<td>2</td>
<td>Using iWay Service Manager Diagnostics, Tracing, and Logging Facilities</td>
</tr>
<tr>
<td></td>
<td>Describes how to use the iWay Service Manager Administration Console to perform diagnostic functions.</td>
</tr>
<tr>
<td>3</td>
<td>Identifying Available Services for Troubleshooting and Debugging</td>
</tr>
<tr>
<td></td>
<td>Identifies the services (agents) that are provided by iWay Service Manager (iSM) for troubleshooting and debugging purposes.</td>
</tr>
<tr>
<td>4</td>
<td>Identifying Available Commands and Functions for Troubleshooting and Debugging</td>
</tr>
<tr>
<td></td>
<td>Identifies the commands and functions that are provided by iWay Service Manager (iSM) for troubleshooting and debugging purposes.</td>
</tr>
<tr>
<td>5</td>
<td>Creating and Using a Remote Command Console</td>
</tr>
<tr>
<td></td>
<td>Describes how to create and use a remote command console in iWay Service Manager (iSM).</td>
</tr>
<tr>
<td>6</td>
<td>Using Event and Startup Process Flows</td>
</tr>
<tr>
<td></td>
<td>Describes how iWay Service Manager (iSM) Event and Startup process flows can be used for troubleshooting and debugging purposes.</td>
</tr>
<tr>
<td>7</td>
<td>Recommended Third-Party Tools for Troubleshooting and Debugging</td>
</tr>
<tr>
<td></td>
<td>Provides information on iWay-recommended third-party tools if the troubleshooting or debugging level scope falls outside of the iWay framework.</td>
</tr>
</tbody>
</table>
## Documentation Conventions

The following table describes the documentation conventions that are used in this manual.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>THIS TYPEFACE or</td>
<td>Denotes syntax that you must enter exactly as shown.</td>
</tr>
<tr>
<td>this typeface</td>
<td></td>
</tr>
<tr>
<td>this typeface</td>
<td>Represents a placeholder (or variable), a cross-reference, or an important</td>
</tr>
<tr>
<td></td>
<td>term. It may also indicate a button, menu item, or dialog box option that you</td>
</tr>
<tr>
<td></td>
<td>can click or select.</td>
</tr>
<tr>
<td>underscore</td>
<td>Indicates a default setting.</td>
</tr>
<tr>
<td>Key + Key</td>
<td>Indicates keys that you must press simultaneously.</td>
</tr>
<tr>
<td>{ }</td>
<td>Indicates two or three choices. Type one of them, not the braces.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>symbol.</td>
</tr>
<tr>
<td>...</td>
<td>Indicates that you can enter a parameter multiple times. Type only the</td>
</tr>
<tr>
<td></td>
<td>parameter, not the ellipsis (...).</td>
</tr>
<tr>
<td>.</td>
<td>Indicates that there are (or could be) intervening or additional commands.</td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>

## Related Publications

Visit our Technical Content Library at [http://documentation.informationbuilders.com](http://documentation.informationbuilders.com). You can also contact the Publications Order Department at (800) 969-4636.

## Customer Support

Do you have questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing tips and techniques. Access Focal Point at [http://forums.informationbuilders.com/eve/forums](http://forums.informationbuilders.com/eve/forums).
You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our website, http://www.informationbuilders.com. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of http://www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

Call Information Builders Customer Support Services (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your questions. Information Builders consultants can also give you general guidance regarding product capabilities. Please be ready to provide your six-digit site code number (xxxx.xx) when you call.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Help Us to Serve You Better

To help our consultants answer your questions effectively, be prepared to provide specifications and sample files and to answer questions about errors and problems.

The following tables list the environment information our consultants require.

<table>
<thead>
<tr>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
</tr>
<tr>
<td>OS Version</td>
</tr>
<tr>
<td>JVM Vendor</td>
</tr>
<tr>
<td>JVM Version</td>
</tr>
</tbody>
</table>

The following table lists the deployment information our consultants require.

<table>
<thead>
<tr>
<th>Adapter Deployment</th>
<th>For example, Business Services Provider, iWay Service Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>For example, WebSphere</td>
</tr>
</tbody>
</table>

Troubleshooting and Debugging Best Practices for iWay Service Manager
<table>
<thead>
<tr>
<th>Request/Question</th>
<th>Error/Problem Details or Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Did the problem arise through a service or event?</td>
<td></td>
</tr>
<tr>
<td>Provide usage scenarios or summarize the application that produces the problem.</td>
<td></td>
</tr>
<tr>
<td>When did the problem start?</td>
<td></td>
</tr>
<tr>
<td>Can you reproduce this problem consistently?</td>
<td></td>
</tr>
<tr>
<td>Describe the problem.</td>
<td></td>
</tr>
<tr>
<td>Describe the steps to reproduce the problem.</td>
<td></td>
</tr>
<tr>
<td>Specify the error message(s).</td>
<td></td>
</tr>
</tbody>
</table>
### Request/Question vs. Error/Problem Details or Information

<table>
<thead>
<tr>
<th>Request/Question</th>
<th>Error/Problem Details or Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any change in the application environment: software configuration, EIS/database configuration, application, and so forth?</td>
<td></td>
</tr>
<tr>
<td>Under what circumstance does the problem <em>not</em> occur?</td>
<td></td>
</tr>
</tbody>
</table>

The following is a list of error/problem files that might be applicable.

- Input documents (XML instance, XML schema, non-XML documents)
- Transformation files
- Error screen shots
- Error output files
- Trace files
- Service Manager package to reproduce problem
- Custom functions and agents in use
- Diagnostic Zip
- Transaction log

For information on tracing, see the *iWay Service Manager User’s Guide*.

### User Feedback

In an effort to produce effective documentation, the Technical Content Management staff welcomes your opinions regarding this document. You can contact us through our website, [http://documentation.informationbuilders.com/connections.asp](http://documentation.informationbuilders.com/connections.asp).

Thank you, in advance, for your comments.

### Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.
For information on course descriptions, locations, and dates, or to register for classes, visit our website (http://education.informationbuilders.com) or call (800) 969-INFO to speak to an Education Representative.
Defining Troubleshooting and Debugging Strategies

This section provides an introduction to iWay Service Manager (iSM) troubleshooting and debugging facilities, and outlines key strategies that can be used.

In this chapter:

- Troubleshooting and Debugging Overview

Troubleshooting and Debugging Overview

iWay Service Manager (iSM) provides a collection of tools and facilities that are designed for troubleshooting and debugging purposes. One of the key challenges in any debugging scenario is to identify the level (or area) where the error is occurring. As a starting point, asking the following questions can help during the early stages of an investigation:

- Is the application generating an error at the server level?
- Is the application generating an error at the channel? Should I be debugging within a channel?
- Is there an issue with the process flow(s) being used by the application? Should I be debugging within a specific process flow?
- Should I be debugging at a thread level?
- Is the error occurring during application design time or runtime?

In actual scenarios, issues may not occur at a specific level, but between levels. For example, as a message or transaction passes between the process flow and channel levels.
Based on the answers to these questions, the appropriate troubleshooting or debugging tool can be identified. The following diagram illustrates this logic and the various levels that a typical application may traverse.

The primary goals of this best practices guide is to:

1. Identify the troubleshooting and debugging facilities that are provided by iSM.
2. Identify iWay-recommended third-party tools if the troubleshooting or debugging level scope falls outside of the iWay framework.
3. Identify scenarios where the appropriate tool or facility can be best used to troubleshoot and debug the issues as they occur.
Using iWay Service Manager
Diagnostics, Tracing, and Logging Facilities

This section describes how to use the iWay Service Manager Administration Console to perform diagnostic functions. Information on how to run iWay Service Manager (iSM) in a command shell to enter commands that can assist you during troubleshooting is provided. It also describes how to use the Windows Event Viewer for troubleshooting purposes.

The following topics include instructions on how to configure logging and debugging properties to view the resulting log files in the console. Instructions on how to enable the tracing features of iSM are also provided, in addition to information about the iSM test tools that assist in debugging.

In this chapter:

- Running in a Command Shell
- Diagnostic Commands
- Troubleshooting on Windows
- Performing Diagnostic Functions
- Measurements and Statistics
- Using the Log Viewer
- Creating a Diagnostic Zip

Running in a Command Shell

You can run iWay Service Manager (iSM) in a command shell to debug and troubleshoot any errors that may occur.

On Windows, the iwsrv command starts iSM in a command window for debugging purposes. For reference, the following topic includes the full syntax of the iwsrv command.

Syntax: How to Start iWay Service Manager in a Command Window (Windows)

Navigate to the iWay home bin directory. For example, on Windows, if iWay is installed in C:\Program Files\iWay61, go to
The syntax for the `iwsrv` command, which starts iWay Service Manager in a command window on Windows, is:

```
iwsrv [configuration] [-s service] [-l launch] [options]
```

where:

**configuration**
- Is the name of the server configuration that is loaded for this instance. The default value is `base`.

**service**
- Is the name of the service that is executed. Valid values are:
  - **start**: Starts the server configuration (default).
  - **stop**: Stops the server configuration.
  - **install**: Installs the server configuration.
  - **remove**: Removes the server configuration.
  - **query**: Queries the server configuration.

**launch**
- Specifies the launch method. Valid methods are:
  - **java**: Loads Java in a separate process and uses the JVM options, NT dependencies, and other preferences found within the iSM configuration that are configured through the console. For example:
    
    ```
iwsrv.exe base -s start -l java
    ```

  - **script file**: Specifies a script file that defines the run-time preferences. This script file must be located in the iWay Service Manager installation directory. For example:
    
    ```
iwsrv.exe base -s start -l iWay61.cmd
    ```
    
    Both of the above uses of `-l` will force the service to load Java in a separate process. When the service is stopped, both `iwsrv.exe` and `java.exe` are terminated.

**options**
- Specifies tracing or server back-up information. Valid values include:
  - **-b**: Indicates that Service Manager is a back-up server, for example:
iwsrv.exe base -s start -b

-c. Turns tracing on. In this mode, you can display useful error messages on the console. For example, you can display a message that says the Java Runtime Environment (JRE) is not properly installed. For example:

iwsrv.exe base -s start -c

-d. Limits tracing to debug only, for example:

iwsrv.exe base -s start -d

-f. [PATH] filters the system path when invoking JAVA. [RESTART] suppresses the JVM fault restart capability.

-h. iWay61 home directory.

-t. The amount of time (in seconds) to process service shutdown.

Example: Starting a Server Configuration With Traces Enabled

The following command starts a server configuration named test and sends traces to the command window as print lines:

iwsrv test -c

Diagnostic Commands

When iWay Service Manager (iSM) is running in a command shell, you can control it by typing commands in addition to using the console. These commands are designed to assist you in resolving issues.

Several of the key commands are listed and described in the following table. To see a full list of available commands, type Help at the Command Prompt on Windows after using the iwsrv command to start iSM.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diagzip</td>
<td>Creates a diagnostic information file for use by iWay Support. For example, you can enter the following command: diagzip c:\temp\Diag_from_base</td>
</tr>
<tr>
<td>errors</td>
<td>Displays the last ten errors reported by the server.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>exits</td>
<td>Displays loaded exits, such as Activity Log and Correlation Manager.</td>
</tr>
<tr>
<td>func</td>
<td>Displays the list of iWay Functional Language (iFL) functions, or the parameters of that function.</td>
</tr>
<tr>
<td>gc</td>
<td>Runs the Java garbage collector.</td>
</tr>
<tr>
<td>help</td>
<td>Displays help for the diagnostic commands. Type the following to display help for a specific diagnostic command: help command_name</td>
</tr>
<tr>
<td>info</td>
<td>Displays channel information.</td>
</tr>
<tr>
<td>license</td>
<td>Displays available iWay license codes.</td>
</tr>
<tr>
<td>line</td>
<td>Prints one or more lines on the command window or the trace log to improve the readability.</td>
</tr>
<tr>
<td>memory</td>
<td>Lists the amount of memory that is currently in use and the amount of free memory that is available.</td>
</tr>
<tr>
<td>pools</td>
<td>Lists resource pools.</td>
</tr>
<tr>
<td>providers</td>
<td>Displays providers currently in use.</td>
</tr>
<tr>
<td>pull</td>
<td>Load information from another configuration or installation.</td>
</tr>
<tr>
<td>quit</td>
<td>Exits the server. All listeners must be stopped.</td>
</tr>
<tr>
<td>refresh</td>
<td>Restarts a specified listener with an updated local configuration.</td>
</tr>
<tr>
<td>run</td>
<td>Runs a command file.</td>
</tr>
<tr>
<td>set</td>
<td>Sets a parameter. Usually used to set the tracing level, for example: set trace on</td>
</tr>
<tr>
<td>shell</td>
<td>Attempts to run an operating system command.</td>
</tr>
<tr>
<td>show</td>
<td>Displays server information.</td>
</tr>
<tr>
<td>sregs</td>
<td>Displays globally available special registers.</td>
</tr>
</tbody>
</table>
### Command Table

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>Starts one or more channels.</td>
</tr>
<tr>
<td>stats</td>
<td>Run statistics on the current instance or listener.</td>
</tr>
<tr>
<td>stop</td>
<td>Stops one or more channels.</td>
</tr>
<tr>
<td>threads</td>
<td>Lists execution threads currently controlled by the server. Does not necessarily include threads started by auxiliary packages, such as third-party interfaces. Useful after a Stop command to determine what is still running.</td>
</tr>
<tr>
<td>time</td>
<td>Prints the GMT time on the console.</td>
</tr>
<tr>
<td>tool</td>
<td>Runs a named tool, such as <code>testfuncs</code>.</td>
</tr>
<tr>
<td>type</td>
<td>Type or display the contents of a text file.</td>
</tr>
<tr>
<td>version</td>
<td>Displays the product version and all later versions of .JAR files.</td>
</tr>
</tbody>
</table>

### Troubleshooting on Windows

Information, warning, and error messages are logged in the Windows Event Log system. When a problem occurs, the Windows Event Log is the first place to look for information.

**Procedure:** How to Display Messages in the Windows Event Viewer

1. Access the Event Viewer from Administrative Tools, which can be found in the Windows Control Panel.
2. From the left pane of the Event Viewer, click *Application* to view iWay Service Manager entries.
The following image shows entries in columns that indicate the type of message, the date and time, the source (for example, iWay Service Manager), the category, the event, and the user, when applicable.

<table>
<thead>
<tr>
<th>Type</th>
<th>Date</th>
<th>Time</th>
<th>Source</th>
<th>Category</th>
<th>Event</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>3/5/2007</td>
<td>7:02:29 PM</td>
<td>iWay Service Manager</td>
<td>None</td>
<td>33</td>
<td>N/A</td>
</tr>
<tr>
<td>Error</td>
<td>3/5/2007</td>
<td>7:03:18 PM</td>
<td>iWay Service Manager</td>
<td>None</td>
<td>6</td>
<td>N/A</td>
</tr>
<tr>
<td>Error</td>
<td>3/5/2007</td>
<td>7:03:18 PM</td>
<td>iWay Service Manager</td>
<td>None</td>
<td>6</td>
<td>N/A</td>
</tr>
<tr>
<td>Information</td>
<td>3/5/2007</td>
<td>6:57:06 PM</td>
<td>ESENT</td>
<td>General</td>
<td>101</td>
<td>N/A</td>
</tr>
<tr>
<td>Information</td>
<td>3/5/2007</td>
<td>6:57:06 PM</td>
<td>ESENT</td>
<td>General</td>
<td>103</td>
<td>N/A</td>
</tr>
<tr>
<td>Information</td>
<td>3/5/2007</td>
<td>6:55:15 PM</td>
<td>iWay Service Manager</td>
<td>None</td>
<td>32</td>
<td>N/A</td>
</tr>
<tr>
<td>Information</td>
<td>3/5/2007</td>
<td>6:53:44 PM</td>
<td>iWay Service Manager</td>
<td>None</td>
<td>6</td>
<td>N/A</td>
</tr>
</tbody>
</table>

3. To view its contents, double-click an entry.

If you are having difficulty starting a service for iWay Service Manager, which cannot be resolved using information from the Event Viewer, start the service in console mode.

**Performing Diagnostic Functions**

The iWay Service Manager Administration Console enables you to configure diagnostic properties for logging and tracing. After logging and tracing properties are enabled, you can view the resulting log files in the console.

**Log Settings**

The Trace Log is used to record the diagnostic information that is generated by the run-time components of iWay Service Manager. The Transaction Log is used to maintain a record of every document received and processed by iWay Service Manager. The following procedure describes how to configure log settings that are defined in the base configuration of iWay Service Manager.
Procedure: How to Configure Log Settings

Settings
- General Settings
- Java Settings
- Register Settings
- Trace Settings

Log Settings
- Path Settings
- Data Settings
- Backup Settings

1. In the left console pane of the Server menu, select Log Settings.
The Log Settings pane displays, as shown in the following image.

**Log Settings**

The trace log is used to record the diagnostic information that is generated by the runtime components of iWay Service Manager. The transaction log is used to maintain a record of every document received and processed by iWay Service Manager. Listed below are the settings defined in the base configuration of this server.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logging</strong></td>
<td>This setting is used to turn trace logging on or off. When trace logging is turned on, all traces will be persisted to a log file as defined by the log file location. Any changes you make with respect to the size setting will not take effect until you restart/deploy the server.</td>
</tr>
<tr>
<td><strong>Logfiles Location</strong></td>
<td>This setting specifies the directory used to save trace log files. The name of each trace log file varies, but is based on the name iWay###.log. Any changes you make to the location setting will not take effect until you restart/deploy the server.</td>
</tr>
<tr>
<td><strong>Logfile Size Limit</strong></td>
<td>This setting is used to limit the size of each trace log file. The value represents the log size in KBytes and is used as a parameter in the implementation of log file rotation. Any changes you make with respect to the size setting will not take effect until you restart/deploy the server.</td>
</tr>
<tr>
<td><strong>Logfiles in Rotation</strong></td>
<td>This setting is used to limit the number of trace log files. The value represents the maximum number of trace log files that are kept before rotating rotation to the beginning. Any changes you make with respect to the log file rotation will not take effect until you restart/deploy the server.</td>
</tr>
<tr>
<td><strong>Message Size Limit</strong></td>
<td>This setting is used to limit the maximum size of data messages placed in the log. The value is specified as a size in KBytes. Tracing large messages severely affects system performance.</td>
</tr>
</tbody>
</table>

2. Change the default values.

For more information, see *Log Setting Properties* on page 22.

3. Click *Update*.

4. For your changes to take effect, restart iWay Service Manager.

**Reference:** Log Setting Properties

The following table lists and describes the log setting properties.
### Property Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logging</td>
<td>Turns logging on or off. Required if you want to log to a file, use a diagnostic activity log, or view the log online.</td>
</tr>
<tr>
<td>Logfiles Location</td>
<td>Directory where the trace log root resides. To create the directory if it does not exist, select the check box.</td>
</tr>
<tr>
<td>(Directory field)</td>
<td></td>
</tr>
<tr>
<td>Logfile Size Limit</td>
<td>Maximum allowed for each file size in kilobytes (used for log rotation). iWay recommends a minimum of one megabyte.</td>
</tr>
<tr>
<td>(Number field)</td>
<td></td>
</tr>
<tr>
<td>Logfiles in Rotation</td>
<td>Maximum number of files to keep (used for log rotation).</td>
</tr>
<tr>
<td>(Number field)</td>
<td></td>
</tr>
<tr>
<td>Message Size Limit</td>
<td>Maximum size of the data message in a log file measured in kilobytes. Large trace messages affect system performance.</td>
</tr>
<tr>
<td>(Number field)</td>
<td></td>
</tr>
</tbody>
</table>

### Trace Settings

Tracing is key to diagnosing problems and thus to application reliability. iWay Service Manager provides a full complement of tracing services, oriented to diagnostic analysis of the running system. Tracing provides a step-by-step explanation of the internal activity of the server.

It is important to note that tracing can affect system performance. The iWay Service Manager Administration Console enables you to select the levels of traces that you want to generate. Unless you are diagnosing a problem, you should limit tracing to error-level only.

A separate category called JLINK debug masks trace messages originating in the iWay JDBC driver that is used to access the main data server. You can specify actual tracing levels for all instances of the driver in the driver settings of the Data Server Properties configuration window. For more information, see *How to Activate JLINK Tracing* on page 27.

The following procedure describes how to control the amount of detail that is produced by the diagnostic components embedded within iSM. Traces produced during run time are displayed or logged based on settings in the run-time environment.
**Procedure: How to Select Trace Levels**

**Settings**

- General Settings
- Java Settings
- Register Settings
- **Trace Settings**
- Log Settings
- Path Settings
- Data Settings
- Backup Settings

1. In the left console pane of the Server menu, select **Trace Settings**.
   
   The Trace Settings pane displays, as shown in the following image.

   ![Trace Settings table](image)

   Trace Settings allow you to control the amount of detail that is produced by the diagnostic components embedded within IWay Service Manager. Traces produced during runtime are either displayed or logged based on settings in the runtime environment. Listed below are the trace settings that are defined and active in the base configuration of this server.

<table>
<thead>
<tr>
<th>Trace Level</th>
<th>Description / Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Displays error messages. Set by default.</td>
</tr>
<tr>
<td>Warning</td>
<td>Displays warning messages. Set by default.</td>
</tr>
<tr>
<td>Info</td>
<td>Displays informational messages. Set by default.</td>
</tr>
<tr>
<td>Debug</td>
<td>Displays extensive trace messages.</td>
</tr>
<tr>
<td>Deep</td>
<td>Displays even more extensive trace messages. Tracing at this level can impact system performance.</td>
</tr>
<tr>
<td>Tree</td>
<td>Displays the document tree as a document is parsed. Tracing at this level can impact system performance.</td>
</tr>
<tr>
<td>Data</td>
<td>Displays data entering/exiting the system. Tracing at this level can seriously impact system performance.</td>
</tr>
<tr>
<td>Validation Rules</td>
<td>Displays trace messages about validation rules. Tracing at this level can seriously impact system performance.</td>
</tr>
<tr>
<td>External</td>
<td>Displays trace messages about external components. Tracing at this level can seriously impact system performance.</td>
</tr>
</tbody>
</table>

   - Click **Update**.

2. If other than the default trace levels (Info and Error) are required, select the desired trace level check box.

   For more information, see [Trace Setting Properties](#) on page 25.

3. Click **Update**.
Reference: Trace Setting Properties

The following table lists and describes the trace setting properties.

<table>
<thead>
<tr>
<th>Trace Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Displays error messages. This trace level is set by default.</td>
</tr>
<tr>
<td>Warning</td>
<td>Displays warning messages. This trace level is set by default.</td>
</tr>
<tr>
<td>Info</td>
<td>Displays informational messages. This trace level is set by default.</td>
</tr>
<tr>
<td>Debug</td>
<td>Reports data that is helpful for debugging situations. Shows logic that</td>
</tr>
<tr>
<td></td>
<td>tracks the path of a document.</td>
</tr>
<tr>
<td>Deep</td>
<td>Used for detailed logic tracing. Stack traces are reported by the system in</td>
</tr>
<tr>
<td></td>
<td>deep debug level. Use only if instructed to do so by iWay Support.</td>
</tr>
<tr>
<td></td>
<td><strong>Caution:</strong> Tracing at this level can impact system performance.</td>
</tr>
<tr>
<td>Tree</td>
<td>Displays the document as it enters and leaves the system in XML form. This</td>
</tr>
<tr>
<td></td>
<td>is a level at which intermediate processing as a document evolves is done.</td>
</tr>
<tr>
<td></td>
<td><strong>Caution:</strong> Tracing at this level can impact system performance.</td>
</tr>
<tr>
<td>Data</td>
<td>Displays the incoming and outgoing documents as they pass to and from the</td>
</tr>
<tr>
<td></td>
<td>protocol channel.</td>
</tr>
<tr>
<td></td>
<td><strong>Caution:</strong> Tracing at this level can impact system performance.</td>
</tr>
<tr>
<td>Validation Rules</td>
<td>Displays trace messages about validation rules.</td>
</tr>
<tr>
<td></td>
<td><strong>Caution:</strong> Tracing at this level can impact system performance.</td>
</tr>
<tr>
<td>External</td>
<td>Displays trace messages about external components.</td>
</tr>
<tr>
<td></td>
<td><strong>Caution:</strong> Tracing at this level can impact system performance.</td>
</tr>
</tbody>
</table>

Procedure: How to Log Traces to a File

If tracing is turned on without logging, the tracing information appears only in the debug window and is not saved to a file.
To log traces to a file:

1. In the left console pane of the Server menu, select Log Settings.

   The Log Settings pane opens.

2. In the Logfiles Location section, specify the path to the directory used to save log files.

3. Click Update.

4. For your changes to take effect, restart iWay Service Manager.

Traces are available in several levels and controlled independently. For more information, see Trace Setting Properties on page 25.

**Note:** Trace settings for managed configurations must be set for each configuration independently.

All levels can be masked, so that the log contains only brief informational and error messages.

Unlike most design time settings, changing trace levels takes immediate effect in the run-time system. Changing the log file location does not take effect until iWay Service Manager is restarted.
Procedure: How to Activate JLINK Tracing

Settings
General Settings
Java Settings
Register Settings
Trace Settings
Log Settings
Path Settings
Data Settings
Backup Settings

1. In the left console pane of the Server menu, select Data Settings.

The Data Settings pane opens, as shown in the following image.

```
Data Settings:
JLINK is a technology that can be used to access information hosted by iWay, WebFOCUS and BDA data servers. Listed below are the general settings for the JLINK in the base configuration of this server.

- JLINK

  Encoding - Identifies the default codepage to be used with the JLINK Data provider.

  Codepage:
  - U.S. English (Default)
  - Select a predefined codepage

  Encryption - Determines whether JLINK traffic should be encrypted over the wire.

  Encryption:
  - On

  Diagnostics - Determines whether JLINK tracing is enabled.

  Diagnostics:
  - On

  Trace Levels - Sets the trace levels of the JLINK Data provider.

  Trace Levels:
  - api
  - io
  - logic
  - debug

  Trace File - File name and location of the file on the server to store JLINK trace information.

  Trace File

  [Update Clear Reset]
```

a. Select the Diagnostics check box.

b. Specify trace levels for specific instances of the driver.

The trace levels are:

- **api**: Provides entry and exit tracing as the application steps through JDBC calls.
io. Traces data in and out of the system.

logic. Traces the internal activity of the driver. This is equivalent to the Debug trace level of the server.

debug. Traces internal operations of the driver. This is equivalent to the Deep Debug trace level of the server.

c. Type the name of the trace file in the Trace File field.

The trace file specification enables you to route traces from the iWay JDBC driver to a specific file. If you do not specify a trace file, the traces (in most cases) appear in the standard server trace. Certain generalized services that use the iWay JDBC driver do not pass traces through the server. In these cases, specification of the external trace file enables the traces to be captured. You may be prompted to send this file to iWay Support as part of the problem resolution process.

2. Click Update.

Measurements and Statistics

The Measurements package allows you to analyze the behavior of iWay Service Manager.

The Statistics package provides the following functionality:

- Reports heap memory usage as an extension to the existing memory command.
- Searches for and detects deadlocked workers as part of the extended threads command.
- Reports CPU and user time expended by masters as part of the extended stats command.

Usage statistics can also be sent to an external monitoring facility for more detailed analysis.

In some situations, the Measurements package can add significant overhead to the operation of iWay Service Manager. Therefore, do not use the Measurements package in a production environment unless that environment is undergoing analysis.

The information and formats described in this topic are release-dependent, and subject to change.

Memory

The memory command displays the amount of memory in use at the time that the command is issued. When the Statistics package is in use, the standard display is augmented by an additional line that starts with the word Heap.
The Java Virtual Machine has a heap, which is the run-time data area from which all required memory is allocated. The heap is created at the startup of the Java Virtual Machine. Heap memory for objects is reclaimed by an automatic memory management system, which is known as a garbage collector. Although the garbage collector runs automatically, you can issue the `gc` command to force it to run for analytic purposes.

The heap memory display has four fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>Represents the initial amount of memory (in bytes) that the Java Virtual Machine requests from the operating system for memory management during startup. The Java Virtual Machine may request additional memory from the operating system, and may also release memory to the system over time. The value of <code>init</code> may be undefined (0) on some platforms.</td>
</tr>
<tr>
<td>committed</td>
<td>Represents the amount of memory (in bytes) that is guaranteed to be available for use by the Java Virtual Machine. The amount of committed memory may change over time (it may increase or decrease). The Java Virtual Machine may release memory to the system, and the value of <code>committed</code> could be less than the value of <code>init</code>. The value of <code>committed</code> is always greater than or equal to the value of <code>used</code>.</td>
</tr>
<tr>
<td>max</td>
<td>Represents the maximum amount of memory (in bytes) that can be used for memory management. Its value may be undefined. If its value is defined, the maximum amount of memory may change over time. If <code>max</code> is defined, the amount of <code>used</code> and <code>committed</code> memory is always less than or equal to the value of <code>max</code>. A memory allocation may fail if it attempts to increase the <code>used</code> memory, such that the value of <code>used</code> is greater than the value of <code>committed</code>, even if the value of <code>used</code> is less than or equal to the value of <code>max</code> (for example, when the system is low on virtual memory).</td>
</tr>
<tr>
<td>used</td>
<td>Represents the amount of memory currently used.</td>
</tr>
</tbody>
</table>

The heap display is more accurate than the memory display issued without the Measurements package installed. The original (standard) information is displayed, in addition to the new heap information.
Enter command:>memory
STR00X35: memory used 8244K, free 591K,
    nodes: cache 1001 allocated 8607, reclaimed 116, destroyed 116
namespace 0 namespace reclaim 0
Heap: init=0K committed=8244K max=65088K used=7662K

Enter command:

The key value is used, which indicates how much memory is currently allocated. As the value of used approaches the value of max, the garbage collector may start, and performance may be eroded.

Deadlocks

Deadlocks

The deadlock detector finds cycles of threads that are in deadlock, waiting to acquire locks. Deadlocked threads are blocked, waiting to enter a synchronization block, or waiting to reenter a synchronization block after a wait call, in which each thread owns one lock while trying to obtain another lock already held by another thread.

A thread is deadlocked if it is part of a cycle in the relation is waiting for lock owned by. In the simplest case, thread A is blocked, waiting for a lock owned by thread B, and thread B is blocked, waiting for a lock owned by thread A.

This is an expensive operation. Use it only in cases in which you suspect that messages are locked up in the system.

To enable monitoring, use the command thread monitor on. To disable monitoring, use the command thread monitor off. While the monitor is enabled, entering the threads command displays information regarding deadlocks, such as the thread name or names, and the lock name or names. The thread names indicate the components that are deadlocked.

Statistics

Statistics

When iWay Service Manager is running without the Measurements package, some statistics are generated with wall clock times. With the Measurements package, the CPU and user state times are also generated. On the summary page, all values for time are reported in seconds, with a precision of four places. It is possible to develop a report with a greater precision for time.

In some cases, wall clock times show useful information. These times provide a measure of performance for a single message as experienced by the sender. They do not provide any information regarding the throughput capacity of iWay Service Manager.
CPU and user times describe the actual execution time expended on messages. Implementation of these measurements depends on the platform and the Java Virtual Machine (JVM). In many cases, the CPU and user times are the same, as the JVM may not discriminate between the two. For those platforms on which the JVM does discriminate, expect the CPU time to be greater than the user time.

User time is the CPU time that the current thread has executed in user mode, that is, the time spent executing iWay Service Manager instructions.

CPU time is the sum of user time and system time. It includes the time spent setting up for JVM services such as locks, network operations, I/O operations, and other services.

Enter command:>stats

In seconds

<table>
<thead>
<tr>
<th>name</th>
<th>count</th>
<th>low</th>
<th>high</th>
<th>mean</th>
<th>variance</th>
<th>std.dev.</th>
<th>ehr</th>
<th>num/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>mq1a</td>
<td>2</td>
<td>0.0470</td>
<td>0.1560</td>
<td>0.1015</td>
<td>0.0030</td>
<td>0.0545</td>
<td>-</td>
<td>9.85</td>
</tr>
<tr>
<td>cpu</td>
<td>2</td>
<td>0.0312</td>
<td>0.0625</td>
<td>0.0469</td>
<td>0.0002</td>
<td>0.0156</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>user</td>
<td>2</td>
<td>0.0312</td>
<td>0.0625</td>
<td>0.0469</td>
<td>0.0002</td>
<td>0.0156</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

The stats command displays a summary of statistics gathered up to that point. To reset the values to zero, use the stats reset command. iWay recommends that you do not rely on statistics until several messages have been handled to completion, as iWay Service Manager front-loads initialization. Once the system is in a steady state, reset the statistics to zero.

The numbers displayed on the summary page are approximate and are intended for general guidance only. Brief descriptions of the displayed fields are provided in the following table. A fuller understanding of the message processing distribution described here requires some knowledge of statistics and probability, as they apply to queuing.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>The number of messages that have been handled, for which statistics have been gathered.</td>
</tr>
<tr>
<td>low</td>
<td>The lowest time recorded for the handling of a message.</td>
</tr>
<tr>
<td>high</td>
<td>The highest time recorded for the handling of a message.</td>
</tr>
<tr>
<td>mean</td>
<td>The numeric mean of the times recorded. This value is the sum of the times divided by the number of messages handled. This value is frequently called the average.</td>
</tr>
<tr>
<td>variance</td>
<td>The statistical variance of the times recorded. Variance is a measure of how numbers disburse around the mean.</td>
</tr>
</tbody>
</table>
### Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>std.dev.</td>
<td>The statistical standard deviation of the times recorded. Standard deviation is a measure of how numbers disburse around the mean.</td>
</tr>
<tr>
<td>ehr</td>
<td>The Ehrlang Density Coefficient, which provides evidence of the randomness of the time distribution. If there are too few values to compute the coefficient, a hyphen (-) is displayed. If the coefficient is sufficiently close to constant, the term <code>const</code> is displayed. This value is an approximation. A value of 1.0 indicates a Poisson distribution, which is the design point of iWay Service Manager. A very low value can indicate that the individual times recorded are skewed and therefore less usable for predicting behavior.</td>
</tr>
<tr>
<td>num/sec</td>
<td>The reciprocal of the mean, providing the number of messages handled per second. This value is displayed for the wall time. It is not a direct measure of the throughput capacity of iWay Service Manager.</td>
</tr>
</tbody>
</table>

The iWay Service Manager Administration Console also displays a summary of statistics. The Listener Statistics pane displays a table similar to the following.

![Table](image)

**Emitted Statistics Information**

iWay Service Manager can emit statistics as each measurement is generated. Statistics records are included in a comma-delimited file of alphanumeric characters.
The following table describes the fields in the file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>The value 1, which is the record type. Other record types may be added in a future release.</td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>The generator (worker) ID.</td>
</tr>
<tr>
<td>tid</td>
<td>String</td>
<td>The transaction ID.</td>
</tr>
<tr>
<td>msglen</td>
<td>Integer</td>
<td>The message length (non-streaming). If the length cannot be determined, the value -1 is specified.</td>
</tr>
<tr>
<td>complexity</td>
<td>Integer</td>
<td>A measure of the complexity of the message. The higher the number, the greater the complexity. This value is generally a measure of the number of nodes in the XML tree. A value of -1 means unknown. For most purposes, the number of digits that this integer has is a good value to analyze.</td>
</tr>
<tr>
<td>timestamp</td>
<td>Integer</td>
<td>The current time, in milliseconds. This value is the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC (Universal Time, Coordinated). The value is a timestamp for the record.</td>
</tr>
<tr>
<td>gregtime</td>
<td>String</td>
<td>The timestamp in GMT (Greenwich Mean Time). The format is: yyyy-mm-ddTth:mm:ss:mmmmZ</td>
</tr>
<tr>
<td>walltime</td>
<td>Float</td>
<td>The wall clock time expended, in milliseconds.</td>
</tr>
<tr>
<td>usertime</td>
<td>Float</td>
<td>The user time expended, in milliseconds.</td>
</tr>
<tr>
<td>cputime</td>
<td>Float</td>
<td>The CPU time expended, in milliseconds.</td>
</tr>
<tr>
<td>usedmem</td>
<td>Integer</td>
<td>The used memory, in kilobytes (K). See Memory on page 28.</td>
</tr>
<tr>
<td>committed</td>
<td>Integer</td>
<td>The committed memory, in kilobytes (K). See Memory on page 28.</td>
</tr>
</tbody>
</table>
The following is a sample record in the file:

```
```

To enable iWay Service Manager to emit these statistics, define the following to the JVM properties

```
-Dstaturl=host:port
```

where host and port are a UDP receiver.

You can specify the host and port of an iWay Service Manager UDP listener that has a process that is defined to handle incoming messages. Use the iWay Service Manager Administration Console to help define Java system properties.

The iwmeasure.jar extension provides a simple StatsGather agent that appends each record to a named file.

Do not run the statistics gathering component on a machine that is being measured. The process of receiving the statistics will be measured, creating a loop. You must use two configurations, preferably on separate machines.

**Tips**

- When you work with the complexity of a document (a number greater than -1 in the complexity field), a good guideline is to use the number of digits in the field. For example, a message that consists of 172 nodes would get a complexity measure of 3, while a message that consists of 1459 nodes would get a measure of 4. For most analytic purposes, this provides a reasonable value.

- Traces use the bulk of time and memory in iWay Service Manager. For valid statistics, turn off all traces. You can use the `set trace off` command to do this, or you can use the iWay Service Manager Administration Console.

Many books are available on queuing theory, the use of available statistics, and the interpretation of displayed fields.

Kushner, Harold J.; *Heavy Traffic Analysis of Controlled Queuing and Communications Networks*. New York, Springer; (June 8, 2001).
Using the Log Viewer

The Log Viewer manages the display properties of system debugging information when the logging and tracing functions are activated. It filters and displays debugging information as each transaction is received and processed. The Log Viewer also displays the date/time range, type, source, and message of every trace entry.

**Note:** In order to display traces of a specific level, you must have previously enabled them to be written to the log file. For more information, see *Log Settings* on page 20 and *Trace Settings* on page 23.

**Procedure: How to Use the Log Viewer**

1. Click **Tools** in the top pane and select **Log Viewer** from the Diagnostics section in the left pane.

   The Log Viewer pane opens, as shown in the following image.

   ![Log Viewer Pane](image)

2. Select a specific log file to view from the Log File drop-down list.

   **Note:** Log file names are reused in a circular queue so that they will not proliferate and consume too much disk space. The date and time stamp is shown in the drop-down list in order to show the correct sequence of the files.
The Log Viewer pane is automatically refreshed and shows the log file you selected.

3. Select the source component, level, date and time range, and number of lines to display and click Refresh.

The contents of the log file, as filtered by your criteria, are displayed.

Tip: Multiple trace sources can be selected by pressing Ctrl and clicking the trace source.

Creating a Diagnostic Zip

The Create Diagnostic Zip option provides a quick way to collect the current configuration and log files. An iWay Software support representative may ask you to create a diagnostic zip for problem analysis.

You can use this function to add any relevant comments to the file. The file is labeled with a timestamp in your configuration directory.

Note: Remove previous trace files prior to running a diagnostic zip.
Procedure: How to Create a Diagnostic Zip

1. Click Tools in the menu bar, which is located in the top pane.

2. In the left pane, select Diagnostic Zip.
   The Diagnostic Zip pane opens, as shown in the following image.

3. Type your comments in the space that is provided.
4. Click Create Diagnostic Zip.
   In this example, if you are using the base configuration, the file is saved to the location shown in the following image.

   Information successfully saved into the file:
   C:\PROGRA~1\iWay60\config\base\DIAGNOSTICS-2010-01-08-20-19-16.zip
Creating a Diagnostic Zip
Identifying Available Services for Troubleshooting and Debugging

This section identifies the services (agents) that are provided by iWay Service Manager (iSM) for troubleshooting and debugging purposes.

In this chapter:

- Activity Log Entry Service (com.ibi.agents.XDXALogEvent)
- Catch Service (com.bi.agents.XDCatchAgent)
- Fail Service (com.ibi.agents.XDFailAgent)
- QA Service (com.ibi.agents.XDQAAgent)
- Trace Message Writer Service (com.ibi.agents.XDTraceAgent)

Activity Log Entry Service (com.ibi.agents.XDXALogEvent)

Syntax:

```
com.ibi.agents.XDXALogEvent
```

Description:

This service is used to record events to the system log during a process flow. It can record security events and arbitrary user event codes. Each message that is logged has a type, code, and optional message.

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactional</td>
<td>Determines whether this event record will be included in the log based on the transaction. If set to true, the event is logged only if the entire process flow is successful. For transactional recording, the channel must be declared to control the local transaction.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Event     | The event class to be included in the log. Select one of the following values from the drop-down list:  
- emit {emit}  
- security {security} (default)  
- emit {signature}  
- crypto {crypto}  
- user {user}  
You can also type an arbitrary, user-defined value in the Event field that must be greater than 1000. |
| Code      | A code that further describes the event. Select one of the following values from the drop-down list:  
- start {start} (default)  
- end {end}  
- fail {fail}  
- sign {sign}  
- encrypt {encrypt}  
- decrypt {decrypt}  
- verify {verify}  
You can also type an arbitrary, user-defined value in the Code field that must be greater than 1000. |
| Message   | An arbitrary message that you want to associate with this event record. |

**Available Response Edges for XDXALogEvent**

When you connect the XDXALogEvent object to an End object using the OnCustom build relation in a process flow, the available line edges are provided in the Line Configuration dialog box.
The following table lists and describes the available line edges for the XDXALogEvent object.

<table>
<thead>
<tr>
<th>Line Edge</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnError</td>
<td>Error</td>
</tr>
<tr>
<td>OnSuccess</td>
<td>Success</td>
</tr>
<tr>
<td>OnFailure</td>
<td>Failure</td>
</tr>
</tbody>
</table>

**Catch Service (com.bi.agents.XDCatchAgent)**

**Syntax:**

`com.bi.agents.XDCatchAgent`

**Description:**

Error handling in iWay Service Manager process flows can be accomplished in a number of different ways. The possible methods are:

- Explicitly checking for an error, post-service execution, by conditioning the edge with onError or onFailure.
- Including an outlet conditioned with _iserror().
- Including XDCatchAgent at the beginning of the channel. This channel has two edges on the output side that are used for processing. The first is the onCompletion edge. The second is the onCustom edge, with the onError and onFailure cases selected.

The concept of the XDCatchAgent is similar to a try-catch block in other programming languages.

In other programming languages, a block of code is enclosed between the braces of a try statement. Following the try block is a catch block of code that is enclosed in braces. The code in the catch block has statements that handle any errors that might occur in the try block.

When the thread of execution starts, each line in the try block of code is executed. If each statement is successful, execution continues at the statement following the closing brace of the catch block (assuming that there is not a finally block). If an error occurs within the try block, the thread of execution jumps to the code inside the catch block.
In an iWay Service Manager flow, you can add an XDCatchAgent in front of the services in which an error might occur. There are two edges off this service:

- onCompletion (blue)
- onCustom (brown)

The completion edge is the thread of execution in which everything works in a perfect scenario. All the edges after the service connected by the onCompletion edge are then connected to the onSuccess edge.

The onCustom edge has three selected cases (onError, onFailure, and error_retry). Any errors or failures that occur within the path of the process flow are directed down the onError and onFailure edge. The logic in this branch contains any services necessary to handle errors. The error_retry edge is followed when there is a retry exception. For example, when a SQL Object contains an invalid URL in the process flow, the onCustom/error_retry edge will be followed.

Think of the onCompletion path as the try block and the onCustom edge as the catch block.

You can add multiple XDCatchAgents into a process flow. The error branch is taken off the closest XDCatchAgent previous to where the error occurred. In this manner, you can add multiple error conditions for a given process flow if needed.

**Example:**

In this example, a file is put into a directory after its creation from a previous channel. The sample process flow is responsible for transmitting the file to the customer FTP site.

Since this is an FTP site, it is subject to network and site availability and other possible outside issues. An error handling strategy is required so that none of the documents being processed are lost because of an outside issue.
In process flow, the XDCatchAgent immediately follows the Start block. An onCompletion edge connects the Catch Errors block to the FTP Write block. The FTP Write block is an FTP emitter that is set up to write the file to an FTP site. The service directly following the XDCatchAgent (Catch Errors) must have an onCompletion edge for this to work correctly.

Following the FTP Write block is the End block. The edge connecting these two services is an onSuccess edge. If a different edge were used and an error occurred, the error edge off of Catch Errors may not be executed.

The onCustom edge of Catch Errors has the onError and onFailure cases selected for the properties. This edge leads to a file write service, Write Error, that puts the file into a hold directory for later reprocessing. Following Write Error, there is an End with a Terminate since no further processing is required at this point. In a real world scenario, a requirement might be that an email is sent if the site is down.

When the target FTP site is up and available, the files are written to the FTP site. If the FTP site is down or you cannot connect to it, the FTP write service will generate an error. This error causes the next execution point to be the File Write to save the file for further processing.

**Fail Service (com.ibi.agents.XDFailAgent)**

**Syntax:**

`com.ibi.agents.XDFailAgent`

**Description:**
The failure business service always returns an XDException. If the retry option is selected for the Type of failure parameter, the exception calls for a retry of the input, if possible. This service is useful when debugging rollback logic in a customer business service.

**Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of failure</td>
<td>The type of failure to be thrown. Select one of the following options from the drop-down list:</td>
</tr>
<tr>
<td></td>
<td>- fail</td>
</tr>
<tr>
<td></td>
<td>- retry</td>
</tr>
<tr>
<td>Message</td>
<td>Message to be issued to the user.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Bypass Error Message</td>
<td>Indicates not to trace at the error level when this service terminates the process. This is useful for cases where the trace log is being monitored by an external program for errors. The termination will be traced at the debug level.</td>
</tr>
<tr>
<td>Call at EOS?</td>
<td>In a streaming environment, EOS (End of Stream) is the short message that is sent after the last document, which signifies the EOS. This parameter determines whether this service should be called for the EOS message. The default value is false.</td>
</tr>
</tbody>
</table>

**Example:**

This service can be used in situations where a failure must be reported or simulated (for example, if a certain fatal condition is reported). If you run this service in a process flow using `<test/>` as the incoming message and the retry option is selected for the Type of failure parameter, the following document is returned:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<eda>
  <error code="6" timestamp="2009-06-05T19:56:59Z" source="com.ibi.agents.XDFailAgent" stage="AGENT">XD[RETRY] cause: 0 subcause: 0 message: Retry requested from XDFailAgent</error>
</eda>
```

Otherwise, a retry is silent and the incoming message is retried later.

If the fail option is selected for the Type of failure parameter, the process flow handles the termination as an error condition by searching upward on the execution edge to locate a catch node. If none is found, then the process flow is terminated. If a catch node is found, then standard catch logic is performed.

It is not recommended to design process flows in which failure indications are used to control execution logic (for example, simulating a long jump). Failure indications should only be used in the event of actual failures.

**QA Service (com.ibi.agents.XDQAAgent)**

**Syntax:**

`com.ibi.agents.XDQAAgent`

**Description:**
This service emits a flattened copy of the input document to a file named in the init() parameters. The service outputs the document (XML or flat) in QA, ondebug, or always modes, depending on the configuration setting of the When parameter. If the QA mode is not enabled (in the Diagnostic System Properties Console Configuration page or by using the set command) and the always option is not set, then this service functions as a move service. This service is designed to work as a chained service for debugging purposes. The document and all special registers are included in the output.

The QA mode for iSM can be set by executing the following command to enable the QA mode for the configuration:

```
set qa on [-save]
```

To deactivate the QA mode, execute the following command:

```
set qa off [-save]
```

The QA mode must be enabled for the iSM configuration you are using in order for the QA service to output documents when set to QA mode.

To enable the QA service to output documents on debug, set the iSM debug special register to true. To deactivate the debug mode, set the debug special register to false.

### Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where *</td>
<td>File pattern to receive trace file.</td>
</tr>
</tbody>
</table>
| When        | Determines when to emit the information. Select one of the following options from the drop-down list:  
- qa (default)  
- always  
- ondebug |
| Name        | Identifier name to mark emitted trace document.                             |
| Emit input  | Location (file pattern) to which to emit actual input document. If omitted or empty, the incoming document is not emitted. |
| Base64 Decode | If set to true, the value is assumed to be in base64 notation. Only applicable when a specific write value is specified. |
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting Offset</td>
<td>If set, this value represents the starting offset within the data block to start the dump.</td>
</tr>
<tr>
<td>Maximum Length</td>
<td>If set, this value represents the total number of bytes to dump. If not set, the dump starts from the value specified for the Starting Offset parameter to the end of the buffer.</td>
</tr>
</tbody>
</table>

---

**Trace Message Writer Service (com.ibi.agents.XDTraceAgent)**

**Syntax:**

```java
com.ibi.agents.XDTraceAgent
```

**Description:**

This service writes a message to the trace log of the system. The trace log accumulates messages to record the progress of activity within the server. Usually the trace log is used for debugging purposes.

**Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Level</td>
<td>The trace level at which the message is written to the trace log. Select one of the following trace levels from the drop-down list:</td>
</tr>
<tr>
<td></td>
<td>- Error</td>
</tr>
<tr>
<td></td>
<td>- Warn</td>
</tr>
<tr>
<td></td>
<td>- Debug</td>
</tr>
<tr>
<td>Message</td>
<td>The message to be written to the trace log.</td>
</tr>
<tr>
<td>Call at EOS?</td>
<td>In a streaming environment, EOS (End of Stream) is the short message that is sent after the last document, which signifies the EOS. This parameter determines whether this service should be called for the EOS message.</td>
</tr>
</tbody>
</table>

The edges returned are listed in the following table.
<table>
<thead>
<tr>
<th>Edge</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>success</td>
<td>The line was successfully sent to the trace system.</td>
</tr>
<tr>
<td>fail_parse</td>
<td>iFL used in the parameters was invalid.</td>
</tr>
</tbody>
</table>

The trace message is written to the trace log at the error or debug level. The trace system must be configured to accept the message on the issued level. Users must be made aware that the use of a trace log can adversely affect server performance.

**Caution:** The trace log is not the transaction log, which can hold messages regarding operations within the server. Messages can be written to the transaction log using the Activity Log Entry service (XDXALogEvent) and the Activity Log Business Error Message service (XDXALogBizErr). While trace messages are usually free format and designed to help debug a problem, the XALog (transaction log) is more structured and often has externally imposed security, event code, and format constraints.
Trace Message Writer Service (com.ibi.agents.XDTraceAgent)
Identifying Available Commands and Functions for Troubleshooting and Debugging

This section identifies the commands and functions that are provided by iWay Service Manager (iSM) for troubleshooting and debugging purposes.

**In this chapter:**
- Using the Testfuncs Tool
- Using the Testxpath Tool
- Using the Flow Command
- Using the Line Command
- Using the _eval() Function

**Using the Testfuncs Tool**

The `testfuncs` tool enables you to test an expression in the iWay Functional Language (iFL). It evaluates the expression and returns a result. The `testfuncs` tool is intended for technical users. To use the `testfuncs` tool, type the following command:

```
Tool testfuncs <path to an xml document>
```

This tool supports the `set` subcommand to set a Special Register (SREG) value. The following example shows how to test a SREG with arithmetic:

```
funcs->set aa 1
stored
funcs->sreg(aa)+2
<superroot [baseNode]>
  <arith [funcNodeMath]>+
    <sreg [funcNodeFunctionSreg]>sreg(aa)
    <p-literal [funcNodeLit]>aa</p-literal>
  </sreg>
  <x-literal [funcNodeLit]>2</x-literal>
</arith>
</superroot>
3
funcs->
```

The following example may be more easier to follow:
Using the Testxpath Tool

The testxpath tool enables you to test XPath expressions against a standard document. The XPath expression will be evaluated against the provided document and the result returned. To use the testxpath tool, type the following command:

Tool testxpath <sampledocument>

Assume that the standard document is the same one as shown in the testfuncs tool example:
Enter command: \texttt{tool testxpath \smalldoc.xml}  \\
<?xml version="1.0" encoding="ISO-8859-1"?>  \\
<a>  \\
  <top>  \\
    <b>one</b>  \\
    <b>two</b>  \\
    <x>sreg(iwayhome)</x>  \\
  </top>  \\
</a>  \\
Now enter an XPath expression against the sample document:  \\
\texttt{xpath->/a/top/b[1]}  \\
<superRoot [baseNode]>  \\
  <a [baseNode]/>  \\
  <top [baseNode]/>  \\
  <b [baseNode]>  \\
    <functionPredicate [filterInt]>1</functionPredicate>  \\
  </b>  \\
</superRoot>  \\
values->'one'  \\
tree-><?xml version="1.0" encoding="ISO-8859-1"?>  \\
<xpathresult>  \\
  <b>one</b>  \\
</xpathresult>  \\
list-><b>one [parent=top]  \\
xpath->  \\

Using the Flow Command  

Runs a previously published process flow. The process flow is run under control of the server configuration, rather than under control of a channel. Channel services may not be available within the process flow. The flow command can be used to test process flows, including verifying that the process flow produces the expected result.

To issue a flow command, enter the following:

\texttt{flow <flowname> [\textit{<input>} [-x | -f]] [-c] [-o [@outfile]] [-map pairs...]}  

where:

\texttt{flowname}

Is the name of the flow. The flow must have been published to the system area of the configuration under which it is to be run.
**input**

Is the input to be supplied to the flow. The input can be in flat form (not parsed into XML) or in XML. The -x (default) or -f switches set the type. If the input is flat it will be passed as a string in Unicode, and not in byte form. Input specification is subject to iFL operation. If the input is omitted, a standard signal message will be passed to the flow. You can use the `_file()` iFL function to load the contents of a file to be passed to the flow.

**outfile**

Is the path to a file into which the output document contents are flattened. Use of this feature requires that the -o switch is used.

**-c**

Runs the flow transactionally. If this switch is omitted, the flow is not run under transactional control.

**-o**

Is the output of the flow is displayed in the log. If this is omitted, the output is not displayed. The output is the contents of the documents that are sent to the end nodes of the flow.

**-map pairs**

Adds token=value pairs to the standard signal document if used, as the parameter map. The pairs will also be set as DOC level special registers in the execution environment. This must be the last switch on the line, and all tokens that follow it are considered as token=value pairs. The equal sign (=) and commas are optional.

Example: Run a published flow named status.mail. Pass in the name of the channel to monitor. The flow must look in the standard signal document to get the channel name to monitor. The details of the flow are not shown here.

```
flow status.mail -map channel chan1
```

Because no input was specified in the command, a standard signal document will be passed to the flow. It will look like:

```
<signal type='flow' timestamp='time' version='2' protocol='command'>
  <parms count='1'>
    <parm name='channel'>chan1</parm>
  </parms>
</signal>
```

If the optional -expects switch is used, the flow result is compared on a character-by-character basis with the contents of a named file. If the result of the flow matches the expected result, the following command is emitted to the output trace:
match

If the two do not match, the flow emits the following command:

nomatch

In this case, information showing the location of the mismatch and what was found is traced. For example, consider a regression test of the flow “passthru”.

Enter command:>

Using the Line Command

Prints one or more lines on the command window or the trace log to improve the readability. This is useful as an eye catcher when you are reading a long trace file or command log file.

The line command uses the following format:

\texttt{line \texttt{[\texttt{<count>}] \texttt{-log}}} \\

where:

\texttt{\texttt{<count>}}

Specifies the number of lines to print. The default is one.

\texttt{\texttt{-log}}

Writes the specified number of lines to the trace log instead of the command window.

For example:

Enter command:>

---

---

Troubleshooting and Debugging Best Practices for iWay Service Manager 53
Using the _eval() Function

The _eval() function evaluates a string as an expression of the function. This function uses the following format:

```
_eval(expression [,tracemsg [,level]])
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression</td>
<td>string</td>
<td>The string to be evaluated.</td>
</tr>
<tr>
<td>tracemsg</td>
<td>string</td>
<td>A trace message to be issued when the expression is evaluated.</td>
</tr>
<tr>
<td>level</td>
<td>keyword</td>
<td>The trace level specified for the tracemsg attribute. The following trace levels are supported:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>none.</strong> Does not return any traces.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>error.</strong> This setting provides error level traces.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>debug.</strong> This setting provides debug level traces (default).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>deep.</strong> This setting provides deep level traces.</td>
</tr>
</tbody>
</table>

A common use of the _eval() function is to store a complex expression in a file. The expression can be used by _eval(_file(<path>)). Assume that the file /myfilescfg.txt contains the simple expression _sreg('iway.config','none'). If the _file('/myfilescfg.txt') function is used alone, then the value will be _sreg('iway.config','none'), the value in the file. However, by using the _eval() function, the _sreg() is evaluated and the result is the name of the configuration in which the server is running.

An optional tracing service adds a message at the specified trace level (if enabled) to the current trace log. This is useful for debugging the value to be output by this function. By including the special token (%v) in the trace message, the expression value can be included in the message. For example:

```
_eval('file(/holdif1.txt)','eval got %v','deep')
```
This section describes how to create and use a remote command console in iWay Service Manager (iSM).

In this chapter:

- Remote Command Console Overview
- Creating a Remote Command Console
- Connecting to a Remote Command Console

Remote Command Console Overview

iWay Service Manager (iSM) commands such as `start` or `flow` can be entered at the original command window if iSM (the server) is started as a task with a visible window (for example, starting from a command line such as `iway7.cmd`).

![Command Prompt](image)

Additionally, commands can be entered using a remote command facility using Telnet (with or without Secure Sockets Layer (SSL)) or Secure Shell (SSH). In either case, the full set of iSM commands is available to the user, depending on the security level at which the logged in user has been granted.
A remote command channel is configured by a configuration console user, and need not be part of a deployed iWay Integration Application (iIA) or configuration until it is required.

Usually the remote command channel runs off of the base configuration, and the remote command is used to address other running configurations either on the same or another host. A remote command console can be configured to any configuration that is currently running on a host.

Creating a Remote Command Console

The remote command console is created and managed as a facility in the standard iSM Administration Console. To create a new remote command console, click Command Consoles in the Facilities group on the left pane, as shown in the following image.

![Command Consoles](image)

The Command Consoles pane opens, as shown in the following image.

![Command Consoles](image)

If no remote command consoles have been configured, then the screen will be empty, as currently shown.
If a remote command console has been configured, then it will be listed in the Command Consoles pane (for example, Remote1), as shown in the following image.

![Command Consoles]

**Note:** You can only have a single remote command console configured in any given configuration.

Click New in the Command Consoles pane to configure a remote command console.
The Command Consoles configuration pane opens, as shown in the following image.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command Consoles</strong></td>
<td>Creation and management of Remote Command Console(s). Remote Command Console - Provide remote access via the TCP protocol to the ISM command line console.</td>
</tr>
<tr>
<td><strong>Component Properties</strong></td>
<td></td>
</tr>
<tr>
<td>Name *</td>
<td>Unique name to allow for easy identification of the Remote Console.</td>
</tr>
<tr>
<td>Description</td>
<td>Brief description of this Remote Console. This field will be displayed along with the console name in the console list.</td>
</tr>
<tr>
<td><strong>Configuration Parameters for Command Channel</strong></td>
<td></td>
</tr>
<tr>
<td>Port *</td>
<td>TCP port for receipt of Command Console requests.</td>
</tr>
<tr>
<td>Local Bind Address</td>
<td>Local bind address for multi-homed hosts; usually leave empty.</td>
</tr>
<tr>
<td>Session Timeout *</td>
<td>Max time between commands, in seconds. 0 means no timeout. Max is 10800 seconds.</td>
</tr>
<tr>
<td>Number of Connections</td>
<td>Reject new connections after this many connections are active. Must be between 1 and 20.</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td></td>
</tr>
<tr>
<td>Allowable Clients</td>
<td>If supplied, only messages from this list of fully qualified host names and/or IP addresses are accepted. Enter as comma-separated list or use FILED.</td>
</tr>
<tr>
<td>Security Type</td>
<td>Selected security type: 'none' - implies that the connection and command stream are encrypted, 'ss1' - wraps the connection and command stream in an encrypted Secure Socket Layer, 'ssh' or Secure Shell Handler provides a secure shell encryption and packet handling.</td>
</tr>
<tr>
<td>Client Authentication</td>
<td>When 'ssl' is enabled, the client's certificate must be trusted by the trusted server for a connection to be created. Not used when 'none' or 'ssh' is enabled.</td>
</tr>
<tr>
<td>Authentication Realm</td>
<td>When Security Type is 'none' or 'ssl', the name of a configured authentication realm to validate login. For full access to management commands, the user must be assigned the &quot;admin&quot; role. If not supplied, logins will be delegated to the web console's user database. Not used when Security Type is 'ssh'. For ssh console, authentication options are configured in the SSH provider.</td>
</tr>
<tr>
<td>Security Provider</td>
<td>Required if security is enabled (Security Type of either 'ssl' or 'ssh'). This Security Provider will be used to secure the channel. When Security Type is 'ssh', specify the name of an SSL Context Provider. For 'ssl', specify an SSH Provider.</td>
</tr>
<tr>
<td><strong>Events</strong></td>
<td></td>
</tr>
<tr>
<td>Channel Failure Flow</td>
<td>Name of published process flow to run if the channel cannot start or fails during message use. The server will attempt to call this process flow during channel close down due to an error.</td>
</tr>
<tr>
<td>Channel Startup Flow</td>
<td>Name of published process flow to run prior to starting the channel.</td>
</tr>
<tr>
<td>Channel Shutdown Flow</td>
<td>Name of published process flow to run when the channel is shut down.</td>
</tr>
</tbody>
</table>
The Command Consoles configuration pane contains a table with the following groups of parameters:

- **Component Properties.** Name and description of the listener. This name appears in some logs.

- **Configuration Parameters for Command Console.** Basic parameters including port, sessions, and so on.

- **Security.** Security definitions for the remote command console.

- **Events.** Event-handling parameters that can be configured to run specific process flows when the channel fails, starts, or is shut down.

The first groups (Component Properties and Configuration Parameters for Command Console) define the remote command console and how it will be reached. If no other parameters are configured, then the remote command console will be a standard Telnet command console using the console realm for security.
The Security group can be configured as needed. In this case the remote command console will operate using SSH, with a configured realm (for example, LDAP) and an underlying SSH provider. For more information, see the *iWay Service Manager Security Guide*.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Applies to Telnet?</th>
<th>Applies to Telnet SSL?</th>
<th>Applies to SSH?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowable Clients</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security Type</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Client Authentication</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Authentication Realm</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Security Provider</td>
<td>No</td>
<td>Yes (SSL provider)</td>
<td>Yes (SSH provider)</td>
</tr>
</tbody>
</table>
Events are supported in the Events group, as shown in the following image.

![Events](image.png)

The following table lists and describes each of the available configuration parameters for a remote command console.

**Note:** An asterisk indicates a required parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Component Properties</strong></td>
<td></td>
</tr>
<tr>
<td>Name*</td>
<td>A unique name that will be used to identify the remote command console.</td>
</tr>
<tr>
<td>Description</td>
<td>A brief description for the remote command console, which will also be displayed in the Command Consoles pane.</td>
</tr>
<tr>
<td><strong>Configuration Parameters for Command Console</strong></td>
<td></td>
</tr>
<tr>
<td>Port*</td>
<td>TCP port for receipt of Command Console requests.</td>
</tr>
<tr>
<td>Local Bind Address</td>
<td>Local bind address for multi-homed hosts: usually leave empty</td>
</tr>
<tr>
<td>Session Timeout*</td>
<td>The maximum time between commands, in seconds. A value of zero (0) means no timeout. The highest maximum value that can be entered is 10000 seconds. The default value is 600 seconds.</td>
</tr>
<tr>
<td>Number of Connections</td>
<td>Reject new connections after the specified number of connections are active. A value between 1 and 20 must be entered. The default value is 1 connection.</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Definition</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Allowable Clients</td>
<td>If supplied, only messages from this list of fully qualified host names and/or IP addresses are accepted. Enter as a comma-separated list or use the _file() function.</td>
</tr>
<tr>
<td>Security Type</td>
<td>Select one of the following values from the drop-down list:</td>
</tr>
<tr>
<td></td>
<td>- <strong>none</strong>. Implies that the connection and command stream are not encrypted.</td>
</tr>
<tr>
<td></td>
<td>- <strong>ssl</strong>. Wraps the connection and command stream in an encrypted Secure Socket Layer (SSL).</td>
</tr>
<tr>
<td></td>
<td>- <strong>ssh</strong>. Provides secure shell (SSH) encryption and packet handling.</td>
</tr>
<tr>
<td></td>
<td>The default value selected is <strong>none</strong>.</td>
</tr>
<tr>
<td>Client Authentication</td>
<td>If set to <strong>true</strong> and when the Security Type parameter is set to <strong>ssl</strong>, then the client’s certificate must be trusted by the Telnet server for a connection to be created. Not used when the Security Type parameter is set to <strong>none</strong> or <strong>ssh</strong>.</td>
</tr>
<tr>
<td>Authentication Realm</td>
<td>When the Security Type parameter is set to <strong>none</strong> or <strong>ssl</strong>, the specify the name of a configured authentication realm to validate logins. For full access to management commands, the user must be assigned the <strong>admin</strong> role. If not supplied, logins will be delegated to the web console's user database. Not used when the Security Type parameter is set to <strong>ssh</strong>. For SSH console, authentication options are configured in the SSH provider.</td>
</tr>
<tr>
<td>Security Provider</td>
<td>Required if security is enabled (Security Type parameter value of <strong>ssl</strong> or <strong>ssh</strong>). This security provider will be used to secure the channel. When the Security Type parameter is set to <strong>ssl</strong>, then specify the name of an SSL Context Provider. When the Security Type parameter is set to <strong>ssh</strong>, then specify an SSH Provider.</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>Events</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel Failure Flow</td>
<td>Name of a published process flow to run if this channel cannot start or fails during message use. The server will attempt to call this process flow during channel close down due to the error.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Channel Startup Flow</td>
<td>Name of a published process flow to run prior to starting the channel.</td>
</tr>
<tr>
<td>Channel Shutdown Flow</td>
<td>Name of a published process flow to run when the channel is shut down.</td>
</tr>
</tbody>
</table>

**Connecting to a Remote Command Console**

After you have configured a Telnet remote command console, you can use any command line Telnet client. Consider the following use case scenarios where you need to test iWay Functional Language (iFL) functions or browse help remotely for iWay Service Manager (iSM). The specific use of your Telnet client may vary, and users are referred to their specific Telnet client documentation. The Telnet client is not provided by iWay.

1. Connect to iSM using the command line. For example:

   `telnet csswxzpt3`

2. Enter a user name (for example, iway) and a password (for example, iway).

   ![Telnet Console]

3. Once you are connected and logged in, you can now issue any command to monitor or control your iSM instance.
Using a Telnet Client

In this section, the default Telnet client that is available on Windows is used for demonstration purposes.

Once you start the Telnet client, the following Telnet logon screen is displayed, as shown in the following image.

![Telnet Logon Screen](image)

Provided that the connection meets the selected security criteria you are prompted for a user ID and password. These must be configured in the iSM Administration Console, and may have administrative capabilities or not. Lack of administrative capability means that commands that reconfigure iSM, such as `start`, `stop` and `reinit` are not available.
Once the logon is accepted, you are presented with a standard information screen, as shown in the following image.

At the command line, you can use any authorized command. The help command lists these commands, as shown in the following image.

These are the same commands that can be issued from the standard shell console, plus the showlog and hidelog commands to enable or disable tracing for this Telnet session.
For example, if you enter the `memory` command, the following screen is displayed.

Remote Only Commands

The following iSM commands are available only from remote command consoles:

- **showlog.** Causes the trace log to be sent to the remote console.
- **hidelog.** Causes traces to not be sent to the remote console.

For more information on all of the commands that are supported for iSM, see the *iWay Service Manager Command Reference Guide*.

Telnet Scripting Example

The following is an example of automation or lights out operations that you can achieve after configuring a remote command facility using Telnet. A shell script is created containing the following command:
#!/bin/sh
host=localhost
port=9023
cmd="info"
( echo open ${host} ${port}
sleep 1
echo "iway"
sleep 1
echo "iway"
sleep 1
echo ${cmd}
sleep 1
echo quit ) | telnet > /home/jay/out.txt
echo "
echo "* * * command output start * * *"
cat /home/jay/out.txt
echo "* * * command output end * * *"
echo " 

There are more complex ways of running Telnet on Linux than I/O redirection. For example, the command expect is designed to work with interactive commands.

The following example shows more of the script that can be parameterized as an information-only command, which does not affect the behavior or configuration of the server.
Connecting to a Remote Command Console

```
* * * command output start * * *
telnet> Trying ::1...
Connected to localhost.
Escape character is '^]'.

User: iway
Password: ****
******************************
*
* iWay Secure Message Broker
* Remote Administration Console
*
* protocol: Telnet
* engine: base
* iway.serverip: 127.0.1.1
* locale: en_us
* iwayversion: 7.0.3
* iway.serverhost: UbuntuVM
* iwayworkdir: /iway/prog/7.0.3.36971/config/base
* iwayconfig: base
* console-master-port: 9999
* iway.pid: 3392
* iway.serverfullhost: UbuntuVM
* iwayhome: /iway/prog/7.0.3.36971/
* name: Telnet1
* doclocation: config
*
* you are logged in as iway from localhost (0:0:0:0:0:0:0:1)
*
******************************
Enter command:>info
completed   failed   active   workers   free
SOAP1
    http    -- active --    0   0    0   3    3
    file    -- active --    0   0    0   3    3
Telnet1    -- active --    0   0    1   1    0
Enter command:>quit
goodbye!
* * * command output end * * *
```
Using Event and Startup Process Flows

This section describes how iWay Service Manager (iSM) Event and Startup process flows can be used for troubleshooting and debugging purposes.

In this chapter:

- Event Process Flows
- Startup Process Flow

Event Process Flows

Event process flows can be executed when specific (defined) events occur in iWay Service Manager (iSM) or during message processing. The process flows must be published to the configuration (iWay Integration Application) and must be available for execution at the time that they are called.

The Event process flows can run under the following constraints:

- Communicate with the caller by passing a return code as the name of the End node. This is the same rule as is required for subflows of a regular process.

- Can only return a single document, which may or may not be meaningful to the caller.

- Cannot use Emit nodes, although Emit services are permitted. Emit nodes schedule emits for execution at a later time (asynchronous to the process flow), while Emit services emit directly when they are called.

Other restrictions may apply for individual Event process flows. All Event process flows are conditional, and must be configured for execution if their use is required.

The following Event process flows are described in this section:

- Server Startup
- iWay Business Activity Monitor (BAM) Database Loss of Access
- Channel Startup Failure
- Retry Expired
- Failed ReplyTo
Server Startup

The Server Startup process flow is executed by the iSM initialization routines as iSM starts its execution. This process flow can check for the availability of resources that are required by iSM, and can prevent iSM from starting if the resources are not available. A return of success allows iSM to continue its startup sequence. Otherwise the iSM startup is terminated.

The Server Startup process flow cannot start channels, since iSM is not ready to run channels at this early (startup) stage.

The name of the Server Startup process flow must be entered in the Recovery area of the General Settings page (Process Name field), as shown in the following image.

<table>
<thead>
<tr>
<th>Configuration Backups</th>
<th>Number of automatic backups of the configuration to be maintained. Setting this value to 0 represents 'none'. If the value is greater than 0, then the configuration is backed up after each successful start.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>5</td>
</tr>
<tr>
<td>Configuration Backup Location</td>
<td>The directory where the configuration backups are saved. If the directory is not specified or invalid, the backups are stored in the default directory. If the directory is specified, the backups are stored in that directory. The directory is deleted if it is empty after successful backup.</td>
</tr>
<tr>
<td>Directory</td>
<td>deadletter</td>
</tr>
<tr>
<td>Kill Interval</td>
<td>Frequency (in seconds) that channels are checked for runaway requests that have exceeded their maxlife. The format of the field is expressed as [hh][mm][ss]; for example 04h30m45, which creates a duration of 4 hours, 30 minutes, and 45 seconds.</td>
</tr>
<tr>
<td>Duration</td>
<td>1m0s</td>
</tr>
<tr>
<td>Retry Interval</td>
<td>Frequency (in seconds) that the listener can be retired if it fails for external cause. The format of the field is expressed as [hh][mm][ss]; for example 04h30m45, which creates a duration of 4 hours, 30 minutes, and 45 seconds.</td>
</tr>
<tr>
<td>Duration</td>
<td>2m0s</td>
</tr>
<tr>
<td>Startup Process Flow</td>
<td>If set, this must be the name of a process flow deployed to the system. The flow will be executed when service manager starts, just prior to the initialization of system exits like activity logs and correlation management. If the process does not complete successfully, service manager will not start. To bypass the startup flow, start the server with the -n switch.</td>
</tr>
<tr>
<td>Process Name</td>
<td></td>
</tr>
</tbody>
</table>

Update
The following table lists and describes the possible edges that are returned by the Server Startup process flow.

<table>
<thead>
<tr>
<th>Edge</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>success</td>
<td>Continue with iSM startup.</td>
</tr>
<tr>
<td>&lt;other&gt; or flow fails</td>
<td>Do not continue to start iSM.</td>
</tr>
</tbody>
</table>

**iWay Business Activity Monitor Database Loss of Access**

This process flow is executed when the iWay Business Activity Monitor (BAM) drivers lose connectivity to the BAM database. The process flow can notify an operation area of the problem, and can determine how iSM should continue:

- iSM continues, but BAM update is ignored.
- iSM terminates.
- iSM maintains a local file on disk containing BAM information and attempts to update the database when connectivity is restored.

**Channel Startup Failure**

The Channel Startup Failure process flow applies only to channels that are not started by a specific manual command. This process flow must be published to the system, and is executed whenever the channel cannot initialize. The process flow can be used to send an email to alert an administrator of the issue.

Enter the name of a published process flow to be executed in the Startup Failure Flow field, as shown in the following image. This field is a common channel property that is available for all iSM listeners.

The Channel Startup Failure process flow receives a signal message document for processing. The signal message document uses the following structure and format:

```xml
<channelfail name='channelname' protocol='protocol' state='statecode' statename='name of state' failures='count' version='ism version' time='timestamp' >
     <message>text of message</message>
</channelfail>
```
where:

name
Is the name of the configured channel.

state
Is a specific code describing the current state of the channel. The codes have assigned names, which are available in the statename attribute.

statename
Is the name of the current state, which will usually be one of the following:

- **config.** Cannot start due to a configuration error. The channel is not retried.
- **restart.** iSM will attempt restart.
- **stopped.** iSM will not attempt restart.

protocol
Is the name of the protocol being used by the channel (for example, File).

failures
Is the count of sequential failures (for example, base 1).

version
Is the version of iSM.

time
Provides a timestamp of the failure occurrence.

This process flow can signal iSM to stop retrying the channel by sending a stop message. This is done by naming the End node of the process flow (stop). Termination of the process flow by any other End node will instruct iSM to continue retrying the channel using the standard automatic retry logic.

The information in the signal message document passes information into the process flow concerning the channel and the most likely cause of failure.
In the following simplified example, a failure results in an email being sent to an identified party followed by a check to see if the number of sequential failures exceeds a designated limit (in this example, 3).

![Failure Process Flow Diagram]

Normally this process flow would run during iSM startup or channel restart. To have the process flow run if the start is attempted from an iSM start command whether standalone or in a script, use the `-doflow` switch on the start command. For more information on using the start command, see the iWay Service Manager User’s Guide.

**Retry Expired**

Messages can be queued for retry on channels that support this facility. This includes queue-based channels, the File channel, and the Internal Queue channel. The retries are triggered by logic in the process flow. In this circumstance, the message is re-executed on a periodic basis until expiration has been reached.

At the expiration point, a process flow can be executed to take recovery actions including notification, and optionally, changing the destination address or restarting with a changed (extended) expiration time.

Enter the name of a published process flow to be executed in the Expired Retry Flow field, as shown in the following image. This field is a common channel property that is available for all iSM listeners.
On entry, the process flow receives the document as it exists, at the point at which the process flow is called. The following table lists and describes several special registers that are available in the Retry Expired process flow to assist during the analysis.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iway.eventflow.exitflow</td>
<td>Identifies the purpose of the process flow (for example, expiredRetry)</td>
</tr>
<tr>
<td>iway.eventflow.attempts</td>
<td>Count of the number of retry attempts made before the expiration.</td>
</tr>
<tr>
<td>iway.eventflow.expiredtime</td>
<td>Time of the expiration.</td>
</tr>
</tbody>
</table>

The following table lists and describes the possible edges that are returned by the Retry Expired process flow.

<table>
<thead>
<tr>
<th>Edge</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>success</td>
<td>The process flow overrules the expiration. iSM will attempt to resend, this time with the output of the process flow.</td>
</tr>
<tr>
<td>&lt;other&gt; or flow fails</td>
<td>An error document is sent to the error addresses.</td>
</tr>
</tbody>
</table>

**Failed ReplyTo**

A reply designation associated with a document triggers an emit operation following completion of the process flow. If the emit operation is not successful, the Failed ReplyTo process flow is triggered.

Enter the name of a published process flow to be executed in the Failed ReplyTo Flow field, as shown in the following image. This field is a common channel property that is available for all iSM listeners.
On entry, the process flow receives the document as it exists, at the point at which the process flow is called. The following table lists and describes several special registers that are available in the Failed ReplyTo process flow to assist during the analysis.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iway.eventflow.exitflow</td>
<td>Identifies the purpose of the process flow (for example, failedReply).</td>
</tr>
<tr>
<td>iway.eventflow.replyname</td>
<td>Configured name of the reply or error specification.</td>
</tr>
<tr>
<td>iway.eventflow.destination</td>
<td>The address configured for the emit, as evaluated for use.</td>
</tr>
<tr>
<td>iway.eventflow.errormsg</td>
<td>An error message, if any, describing the cause of the failure that caused this event to be generated.</td>
</tr>
<tr>
<td>iway.eventflow.replyprotocol</td>
<td>Protocol used for the emit attempt (for example, File, MQ, and so on).</td>
</tr>
</tbody>
</table>

The following table lists and describes the possible edges that are returned by the Failed ReplyTo process flow.

<table>
<thead>
<tr>
<th>Edge</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>success</td>
<td>The process flow took responsibility to deliver the message.</td>
</tr>
<tr>
<td>&lt;other&gt; or flow fails</td>
<td>An error document is sent to the error addresses.</td>
</tr>
</tbody>
</table>

Each ReplyTo and ErrorTo is treated separately. If an error occurs for one, an attempt is made to handle the error, and iSM continues with the rest of the list. Error handling, however, differs for ReplyTo versus ErrorTo.

A failed ReplyTo causes the Failed ReplyTo process flow to execute (if present). If the process flow is successful (by terminating at an End node called success), the error is considered to be handled and iSM continues through the rest of the address list. If the process flow is absent, fails, or reaches an End node with a different name, then iSM creates an error document and attempts to send it to the ErrorTo instances recursively. All ErrorTo instances will be called for each ReplyTo that fails.
Document siblings are treated as independent documents. The net effect should be similar to sending the document first, and then each of its siblings one by one. iSM does not expect error documents to contain siblings. However, if present, they too will be sent as top-level documents (which may or may not be in error).

**Send to Dead Letter**

Messages queued for emitting at a later time (using the channel configuration (called ReplyTo and ErrorTo) or the Emit object in a process flow are sent when the outlet of the channel is executed. Messages can also have alternate addresses if required.

If all attempts to emit the message fail, then by default, the message is written to a configured dead letter directory.

If an *emit failed* process flow is configured, then the process flow can examine the message, redirect it, replace it, and potentially notify an appropriate authority. It can then send the message to another channel for a retry attempt or continue to allow the message to be written to the dead letter queue.

Enter the name of a published process flow to be executed in the Dead Letter Flow field, as shown in the following image. This field is a common channel property that is available for all iSM listeners.

On entry, the Send to Dead Letter process flow receives the document as it exists at the point at which the process flow is called. The following table lists and describes several Special Registers (SREGs) that are available in the process flow to assist during the analysis.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iway.eventflow.exitflow</td>
<td>Identifies the purpose of the process flow (for example, <em>deadLetter</em>).</td>
</tr>
<tr>
<td>iway.eventflow.replyname</td>
<td>Configured name of the reply or error specification.</td>
</tr>
<tr>
<td>iway.eventflow.destination</td>
<td>The address configured for the emit, as evaluated for use.</td>
</tr>
<tr>
<td>iway.eventflow.errormsg</td>
<td>An error message, if any, describing the cause of the failure that caused this event to be raised.</td>
</tr>
</tbody>
</table>
The following table lists and describes the possible edges that are returned by the Send to Dead Letter process flow.

<table>
<thead>
<tr>
<th>Edge</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>success</td>
<td>The message was successfully handled.</td>
</tr>
<tr>
<td>&lt;other&gt; or flow fails</td>
<td>The output of the process flow to be written to the dead letter directory, if configured.</td>
</tr>
</tbody>
</table>

Each ReplyTo and ErrorTo is treated separately. If an error occurs for one, an attempt is made to handle the error, and iSM continues with the rest of the list. Error handling, however, differs for ReplyTo versus ErrorTo.

A failed ReplyTo causes the Failed ReplyTo process flow to execute (if present). If the process flow is successful (by terminating at an End node called success), the error is considered to be handled and iSM continues through the rest of the address list. If the process flow is absent, fails, or reaches an End node with a different name, then iSM creates an error document and attempts to send it to the ErrorTo instances recursively. All ErrorTo instances will be called for each ReplyTo that fails. ErrorTo instances are used to communicate errors to administrators who are able to resolve such situations.

A failed ErrorTo causes the Send to Dead Letter process flow to execute (if present). If the process flow returns success, iSM considers the error to be handled and continues with the rest of the address list. If the process flow is absent, fails, or reaches an End node with a different name, then iSM attempts to write a file under the configured dead letter directory.

Sending an error to an empty list of ErrorTo instances is an error. It is handled the same way as a failed ErrorTo.

Notice that only error documents are sent to the configured dead letter directory. If an error cannot be reported (because an ErrorTo fails or there are no ErrorTo instances), then iSM attempts to send the error document to the dead letter directory to keep a record for manual processing. An error document contains a copy of the original document that generated the error.
iSM attempts to avoid sending to a duplicate address within the list if iSM already knows it is a bad address. This could happen when an ErrorTo is also a ReplyTo. A duplicate bad address is treated the same as a regular failed ReplyTo or ErrorTo, except the IO was never attempted.

Document siblings are treated as independent documents. The net effect should be similar to sending the document first, and then each of its siblings one by one. iSM does not expect error documents to contain siblings. However, if present, they too will be sent as top-level documents (which may or may not be in error).

**Parse Failure**

The Parse Failure flow is invoked if an incoming message fails the *parse to XML* operation for a channel. This does not apply to a parse that is handled within a process flow by a service (agent) for that purpose.

The incoming document to the flow contains the message that failed parsing. The standard Special Registers (SREGs) for the protocol are available in the flow. For example, a bad message on a File listener will provide the usual information on the source of the file.

The Parse Failure flow can also be used to send a notification.

The flow can replace the document that could not be parsed. This might be done to *fill in* an element in a large batch managed by a splitting preparser. To replace the message, set the document on output to the message required, and return through an End node named *Replace*. The replaced message will then pass through the normal channel cycle. It may be necessary in your application to set a SREG in order to notify subsequent processes that this is a *placeholder* message. If this technique is used, then remember to set the SREG at the channel level, so as to make it available beyond the scope of the flow.
On entry to the event flow, the SREG `iway.parsefail` will be set to the count of the number of parse failures in this channel for this transaction. This count is useful for batch handling, in which a splitting preparser divides the batch into a sequence of sub-messages. For example, your flow might determine that the count of `placeholder` messages returned to the channel has exceeded a threshold, and so elects to take application action to reject the batch.

![Diagram](image)

**Startup Process Flow**

The Startup Process Flow optionally executes as iSM starts. The name of the process flow is entered in the *Recovery* area of the console. If named and present, the process flow is executed by the server just prior to the installation of system components. For example, if SNMP did not begin, then the process flow itself will not be recorded in the activity logs.

If the process flow ends successfully, the server continues with its startup process. If the process does not end successfully (for example, a fail service is encountered), the server does not start.

The process flow is designed to enable the server to verify the availability of required resources. For example, an SQL service in the process flow may perform a simple select against the Business Activity Monitor (BAM) tables by accessing the jdbc/BAMDBProvider. If the select fails, it can be assumed that the BAM database is not available, and the process flow issues a fail. This would prevent processing if BAM, deemed by the application designer to be a critical resource, is not available. Similarly, if an application required the transfer of data from an Oracle to a DB2 database, the startup process flow could determine that both are available before allowing the server to start. Startup criteria are at the discretion of the application designer.

Once started, the server manages errors and recovery normally.
You cannot control the server from this process flow. For example, you cannot use the control service to start channels because the server has not yet been sufficiently initialized for channels to properly start. Other facilities, including the autostart script, can be used for this purpose.

The following image shows the Recovery pane.

On entry, the input document to the process flow is shown below:

```xml
<startup version=currentversion time=timestamp/>
```

where:

- `currentversion`  
  Is the server version number, such as 7.0.

- `timestamp`  
  Is a standard RFC 3339 (ISO 8601) timestamp.

The output document is ignored.

The startup parameter -r causes iSM to start without calling the startup exit. This allows a buggy startup exit to be bypassed so that iWay tools can be used to correct any problems.

**Note:** This is available under the batch (manual) startup mode. Users are advised to avoid starting as a service until the startup exit is known to be functioning properly.
Chapter 7

Recommended Third-Party Tools for Troubleshooting and Debugging

This section provides information on iWay-recommended third-party tools if the troubleshooting or debugging level scope falls outside of the iWay framework.

In this chapter:

- JConsole
- JVM Startup Options
- SoapUI
- KeyTool IUI
- Operating System Commands
- Wireshark
- Tcpdump

JConsole

The Java Monitoring and Management Console (JConsole) can be used to provide information on iWay Service Manager (iSM) performance and resource consumption running on a Java platform. The JConsole uses Java Management Extension (JMX) technology.

JConsole provides a visual (graphical) representation of the Java Virtual Machine (JVM) environment where iSM is running.

Specifically, JConsole provides time-range charts showing usage for the following JVM components:

- Memory Heap Usage
- Threads
- Classes
- CPU Usage

For more information on configuring and using JConsole, see:

http://docs.oracle.com/javase/6/docs/technote MANAGEMENT/jconsole/index.html
**Procedure:** How to Enable iWay Service Manager for Monitoring Through JConsole

1. Log in to the iWay Service Manager Administration Console.
2. Click **Java Settings** in the left pane.
3. Specify the following Java startup options:
   - `-Dcom.sun.management.jmxremote.port=12356`
   - `-Dcom.sun.management.jmxremote.authenticate=false`
   - `-Dcom.sun.management.jmxremote.ssl=false`
4. Click **Update**.
   **Note:** Use caution when modifying the Java settings. If you make a mistake, then iSM may not start.
5. Stop and start iSM.
6. Using JConsole, access iSM by entering the host name and port.

**JVM Startup Options**

The following JVM startup options can be used to control the Heap size.

**Initial Heap Size:**

- `-Xms256m`

**Maximum Heap Size:**

- `-Xmx256m`

The following JVM startup options can be used for JVM OOME debugging:

- `-XX:-HeapDumpOnOutOfMemoryError`
- `-XX:-HeapDumpPath=./java_pid<pid>.hprof`
SoapUI

SoapUI is a free, open source, and cross-platform functional testing solution. In a single testing environment, SoapUI provides complete test coverage and supports all of the standard protocols and technologies.

SoapUI can be used:

- During the development of web service clients that need to be tested on an interactive level.
- For baseline, load, and soak testing strategies.
- For fixed rate testing strategies.
- For variable load testing strategies.
- During statistics calculation and thread count changes.
- When simultaneously running multiple load tests.

For more information on configuring and using SoapUI, see:

http://www.soapui.org/

KeyTool IUI

KeyTool IUI is a cryptography GUI tool that allows you to configure keys and certificates, including the ability to verify, sign, encrypt, and decrypt the files.

KeyTool IUI provides you with the following functionality:

- Exporting and importing trusted certificates
- Creating DSA and RSA keypairs
- Creating a CA certificate reply to a RSA keypair
- Exporting CSR from a RSA keypair
- Creating empty keystore functions

KeyTool IUI displays detailed information in the GUI about private keys (keypairs) and trusted certificates, regarding their valid date, self-signed, trusted CA, key size, certificate type, certificate signature algorithm, or modified date.

For more information on configuring and using KeyTool IUI, see:

http://code.google.com/p/keytool-iui/
Operating System Commands

Another recommendation is to use operating system commands (for example, Netstat and Traceroute), which are also effective for troubleshooting and debugging purposes.

*Netstat* is a command-line tool that displays network connections (incoming and outgoing), routing tables, and a number of network interface (network interface controller or software-defined network interface) and network protocol statistics. For more information, see:  


*Traceroute* is a network diagnostic tool for displaying the route (path) and measuring transit delays of packets across an Internet Protocol (IP) network. For more information, see:  


**Wireshark**

Wireshark is a free and cross-platform compatible packet analyzer. It is used for network troubleshooting and analysis. Wireshark is similar to *tcpdump*, but has a graphical user interface and some integrated sorting and filtering capabilities.

For more information on configuring and using Wireshark, see:  


**Tcpdump**

A fairly common packet analyzer, tcpdump is also free and cross-platform compatible. Tcpdump analyzes network behavior, performance and applications that generate or receive network traffic. It can also be used for analyzing the network infrastructure itself by determining whether all necessary routing is occurring properly, allowing the user to further isolate the source of a problem.

For more information on configuring and using tcpdump, see:  

Feedback

Customer success is our top priority. Connect with us today!

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also preview new innovations to get an early look at new content products and services. Your participation helps us create great experiences for every customer.

To send us feedback or make a connection, contact Sarah Buccellato, Technical Editor, Technical Content Management at Sarah_Buccellato@ibi.com.

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at Frances_Gambino@ibi.com.