TIBCO iWay® Service Manager

Configuration and Usage Best Practices

Version 8.0 and Higher
March 2021
DN3502293.0321

Contents

Chapter 1

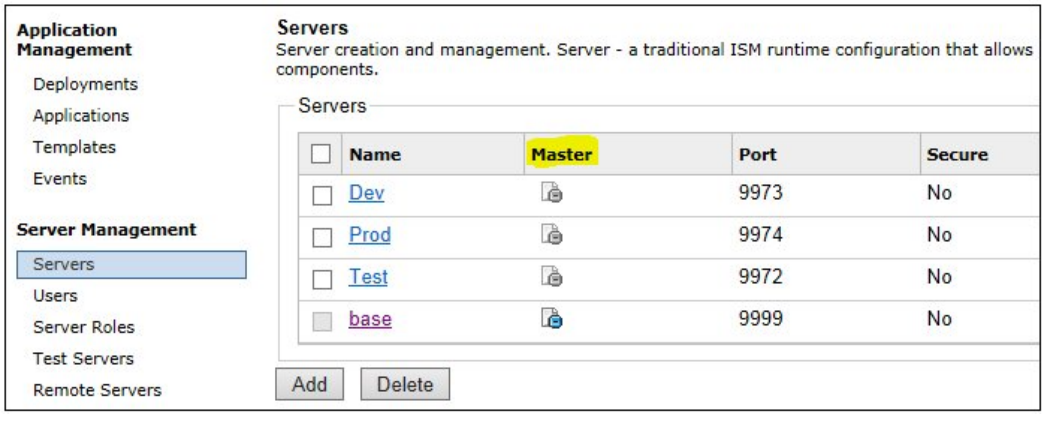
Ensuring Channels are Not Deployed to the iSM Base Configuration

This section describes the importance of not deploying channels to the iSM Base configuration.

In this chapter:

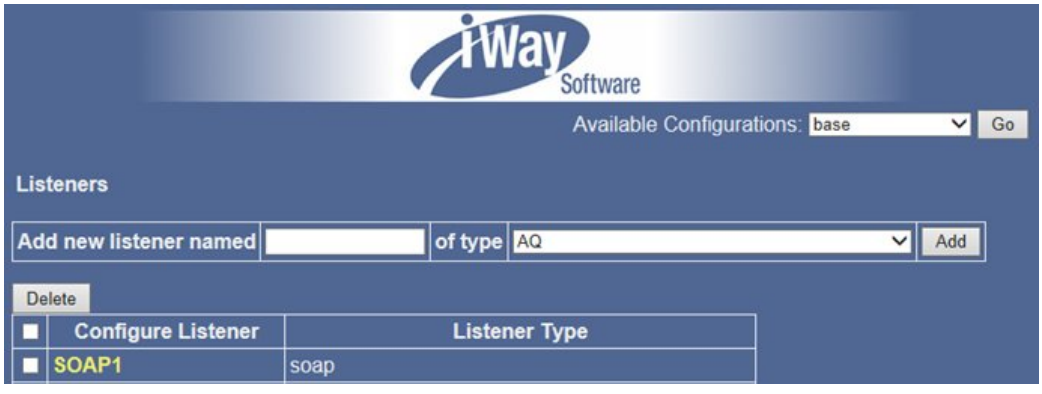
Understanding the Base Configuration

The Base configuration is the master configuration or managed server. The iSM web console default port 9999 is defined within the *base* server, as shown in the following image.



Understanding the SOAP1 Listener

The default SOAP1 listener (listening on port 9000) is also defined within the *base* server. This SOAP listener is hidden by default and can only be viewed on the blue console, as shown in the following image.



You can access the blue console by clicking the iSM version/build hyperlink in the upper-right corner of the iSM console, as shown in the following image.

Chapter 2

Adding an Error-to Agent to the SOAP1 Listener

This section describes how to catch any errors that may occur within the SOAP1 listener.

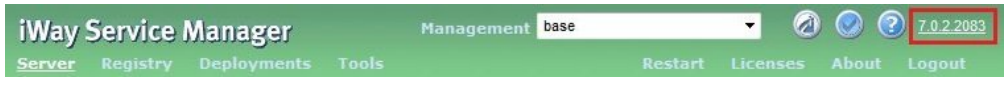
In this chapter:

Adding an Error-to Agent to the SOAP1 Listener

Debugging the SOAP1 listener or at least catching any errors that may occur within the SOAP1 listener is accomplished by configuring an Error-to agent.

Procedure: How to Add an Error-to Agent to the SOAP1 Listener

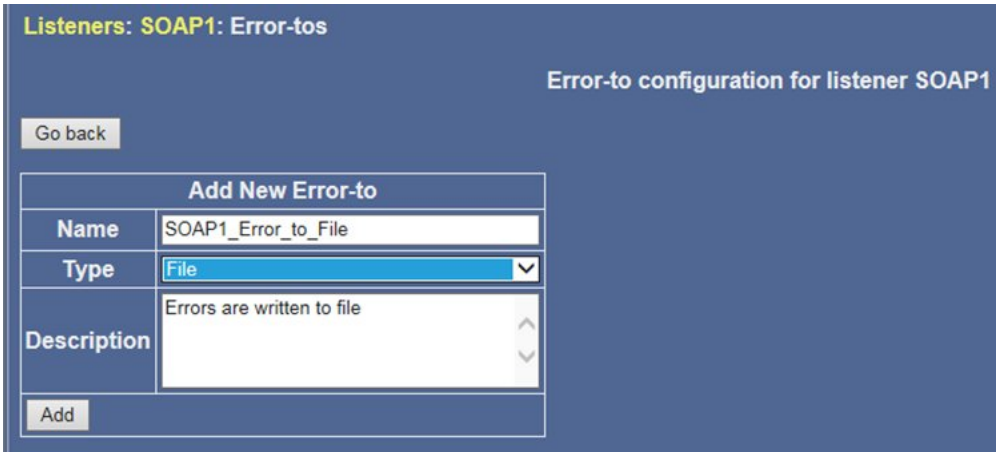
1. Access the blue console by clicking the iSM version/build hyperlink in the upper-right corner of the iSM console, as shown in the following image.



   The blue console opens.

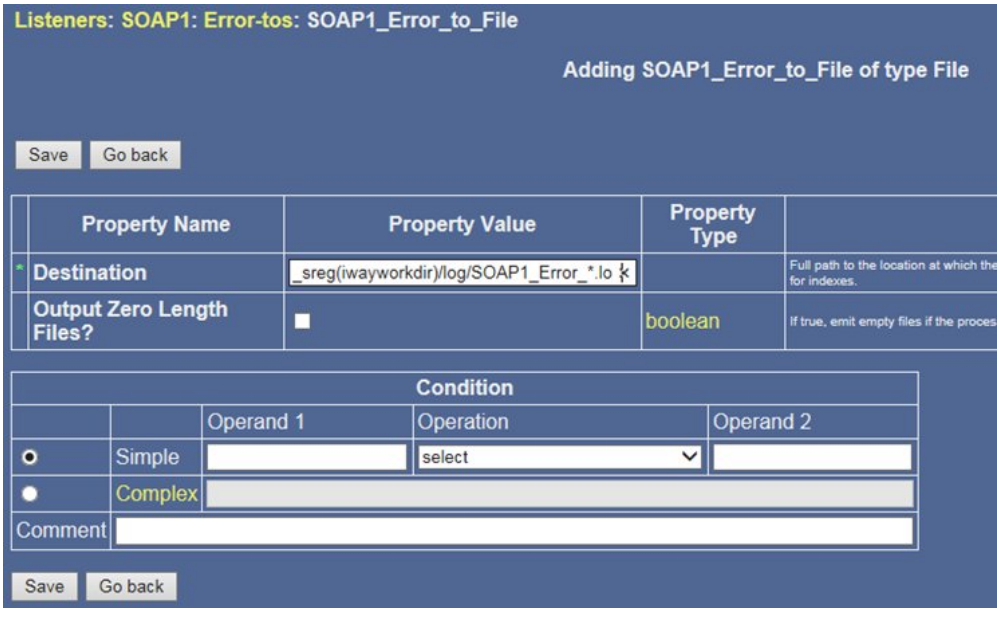
2. In the Configuration for SOAP listener SOAP1 pane, click *Error-to* in the Additional Configuration area, as shown in the following image.

3. Provide a name for the Error-to agent, description (optional), and select File from the Type drop-down list, as shown in the following image.



4. Click *Add*.

   The Adding SOAP1_Error_to_File of type File opens, as shown in the following image.

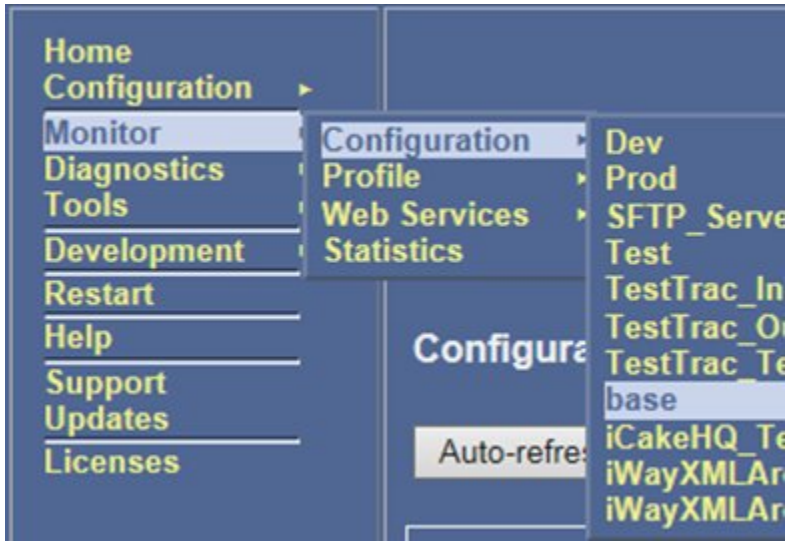


5. In the Destination field, specify a destination directory/file name. For example:

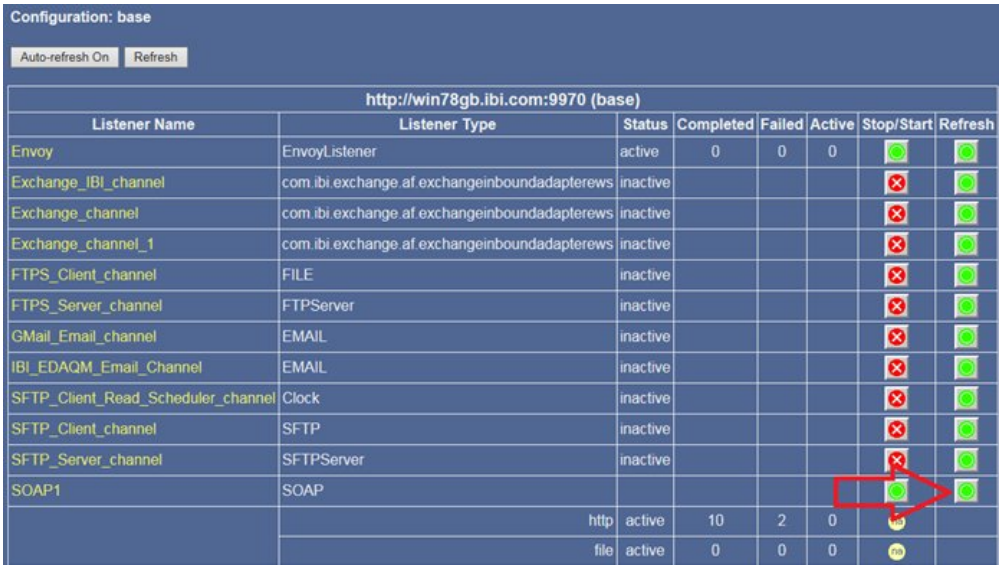
   ```
   sreg(iwayworkdir)/log/SOAP1_Error_*.log
   ```

Note: Including an asterisk character (*) will automatically append a timestamp to the output file name.

6. Click *Save*.
7. To commit this change you must refresh the SOAP1 listener.
8. Click *Monitor* in the left pane, select *Configuration*, and then click *base*, as shown in the following image.



A table that lists all of the available listeners is displayed, as shown in the following image.



9. Click the green icon in the Refresh column for the SOAP1 listener.

Chapter 3

Using iWay Business Activity Monitor (BAM)

This section describes best practices when using iWay Business Activity Monitor (BAM).

In this chapter:

iWay BAM Overview

iWay Business Activity Monitor (BAM) is an extension of iWay Service Manager (iSM) and provides an end-to-end, non-invasive view into transaction life cycles as they span across multiple channels, web services, and iSM servers. This enables you to capture, analyze, resolve, and act upon business transaction events gathered by iWay BAM. The architecture of iWay BAM is based on iSM and uses standard iSM components to provide a seamless integration with an existing application life cycle.

It is important to note, that the *resolve* capability is designed to intercept the message at a particular stage in the process flow, correct and drop it for further processing in the overall application.

iWay BAM Best Practices

The following list provides best practices for iWay Business Activity Monitor (BAM).

- Maintain iWay BAM properly in order to ensure continuous performance when monitoring several channels.
- Archive the iWay BAM database at scheduled intervals.

  The iWay BAM database has to be archived on a regular schedule since the amount of data being processed through the monitored channels will cause the database to continuously grow. Based on customer implementations, it is recommended to archive the database between four and seven days, depending on the data traffic. The more data that exists in the database, as a consequence, the slower the BAM GUI will respond when you refresh the page.

There are two ways to archive the iWay BAM database and reduce its size:

1. You can have a database administrator use standard database tools to archive the tables.
2. You can use the iWay *archive* command. For more information, see the *iWay Business Activity Monitor User's Guide*.

- The *BAMChannel* should be deployed in a different managed server. Deploying *BAMChannel* in a different managed server in iSM will ensure that the display of data in the BAM GUI is not interfering with the overall data processing in iSM.
- Set the *Want Events* parameter to *false*.

  By setting this parameter to false, less information is captured during channel activity and the overall channel activity is monitored. This option is viable in a production system that would not need to monitor the internal steps of each transaction (such as every single step of the process) like event messages on the transaction level. As a result, the size of the database increases at a slower rate. Specifically less inserts of data are made, which results in higher performance.

General Considerations Related to iWay BAM Performance

The following list provides general considerations that are related to iWay Business Activity Monitor (BAM) performance.

- iWay BAM is based on the Activity Logger, which means that all data inserts are performed in parallel to the business process. When a process flow executes, it pushes its information to the logger. The logger runs in a parallel thread writing the collected information to a database. That is why you have special handling under the iWay BAM Activity Logger, which is in the event that the BAM database is not available.
- The configuration of the JDBC Data Provider also has an impact on BAM performance. If the Validate on Return or Validate on Borrow parameters are set to true, then these settings will decrease the performance of the system.
- Correlation being part of an iWay BAM implementation or architecture without iWay BAM in a process flow is a defined as the Correlation Facility, which is used by the Correlation services (agents). These services are called explicitly in the process flow and have an impact on the process flow execution and total amount of calls to the database. This affects the performance as a result, since these services represent another SQL call.

Chapter 4

Using iWay Integration Applications (iIAs)

This section describes best practices when using iWay Integration Applications (iIAs).

In this chapter:

iWay Integration Applications Overview

iWay Integration Applications (iIAs) provide an automated, scriptable, deployment toolset that enables you to build and deploy integration solutions to development, test, and production environments. iIA is an application format that can contain multiple channels and services as well as custom iWay Functional Language (iFL) functions, services (agents) and other dependencies. iWay Integration Tools (iIT), iSM Administration Console, and deployment scripts will use iIAs as containers for pre-built channels and services. iIAs can be source managed and shared among different operational environments.

Various operational environments (such as development, test, or production) are represented by specific server Templates. Templates contain specific environment settings (for example, email addresses, database connection parameters, and so on) in Special Registers (SREGs) or Providers.

Once it is located an iWay server, an iIA can be deployed by using a Template.

iIAs represent an integration solution with a dedicated runtime environment. iIAs can be deployed, started, stopped, and deleted without affecting other iIAs.

iIAs allows you to:

- Build application archives in a development environment and deploy them to test and production environments as required.
- Build application archives from the iWay Registry or an iIT project.
- Manage application deployment from iIT, iSM Administration Console, and deployment scripts.

iWay Integration Applications Best Practices

The following list provides best practices for iWay Integration Applications (iIAs) and their advantages.

- **Transferring an iWay Application From a Development Server to a Production Server**

  On a development server, an application should send email to the developer, while on a production server, an application should send email to production support. Using an iIA, a Production Template can be used to set an email special register to *production.support@ibi.com*, while the Development Template can set this special register to *developer@ibi.com*.

- **Building iWay Integration Applications in iWay Integration Tools**

  In this scenario, a user has an iIT project containing a channel, two process flows, and a Transform component. They are all part of an application, which the user wants to build and deploy to a development server.

  The user can create an iWay Application Archive (iAA) from the iIT project. This creates an application archive file in the iIT project with all of the default deployment properties pre-defined in the project. If there is a name collision or an unresolved dependency, the build will fail.

  Another option for building an iAA is to run an iWay Ant task and point it to the iIT project location. This approach allows building iAAs without any user interaction.

- **Building and Deploying iWay Integration Applications From the iWay Registry Using the iSM Administration Console**

  In this scenario, a user selects existing channels and process flows from the iWay Registry, builds an iWay Application Archive (iAA) and then (optionally) creates a new Template. Finally, the user deploys the iIA by combining an iAA with a Template. The iIA can be started from the command line, just like any other iWay server configuration.

- **Configuring Automatic Deployments on Multiple Remote Servers**

  In this scenario, a pre-built iWay Application Archive (iAA) is stored in an artifact repository or in a source management system. A user wants to deploy this iAA to multiple servers. The shell script for this use case checks the iAA out of the artifact repository, transfers the file to each server, opens a secure shell connection to each remote server and executes *deploy* and *start application* commands using the iWay Ant extension.

Chapter 5

Configuring the Heap (Memory) Size

This section describes best practices when configuring the Heap (memory) size.

In this chapter:

Heap Memory Size Overview

The service for iWay Service Manager (iSM) that runs on either Windows or Linux instantiates a Java Virtual Machine (JVM) upon startup. Left alone, this JVM will try to acquire memory based on free physical memory reported by the platform. At times, this can represent an unnecessarily large amount of memory.

Controlling the amount of memory or Heap used by the iSM service carries several advantages:

- Allowing multiple iSM services to run without using all available system memory.
- Memory size may impact performance.
- Avoiding *Out of Memory* errors and exceptions.

Setting the iSM Service Initial Heap Size

To set the iSM service initial heap size, use the following configuration setting:

```
-Xms256M
```

where:

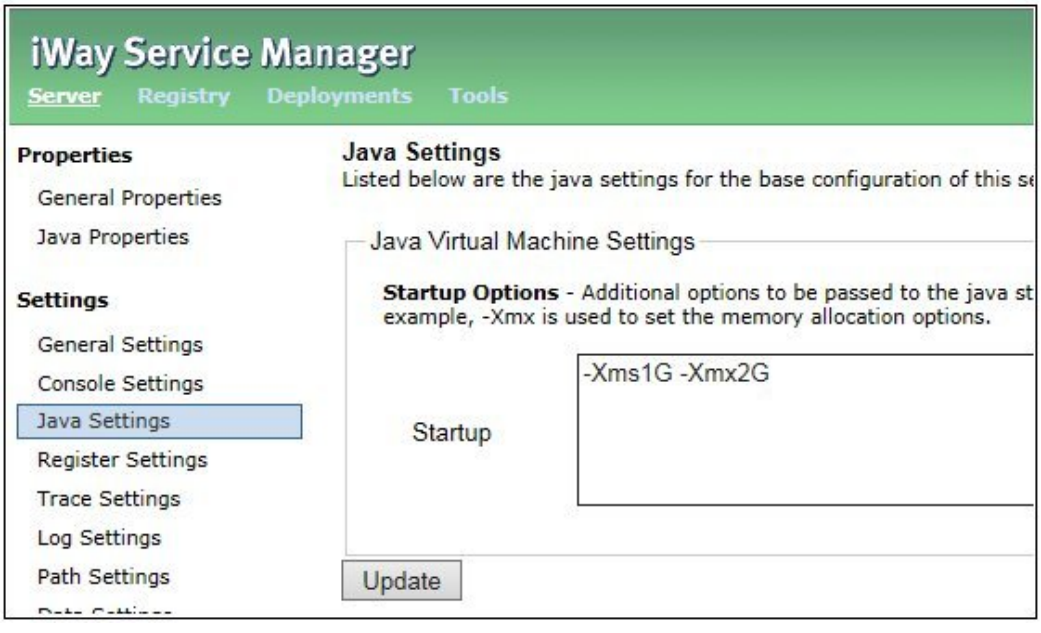
256M

Is the memory size.

To set the maximum heap size, use the following configuration setting:

```
-Xmx256M
```

Note that setting these two configuration settings to an equal size will eliminate the need for memory management, which may increase performance.



Note: The `-Xms1G -Xmx1G` configuration settings set the Heap size to one Gigabyte.

Chapter 6

Benefits of Multiple Cores

This section describes the benefits of using multiple cores in your environment.

In this chapter:

Multiple Cores Overview

The benefit to having multiple cores is that the Garbage Collector (GC) can use a dedicated core in parallel while the application uses other cores and is uninterrupted. This is instead of a *Stop the world* type of collection, which effects the running application.

Java 7 GC makes use of multiple cores using the G1(-XX:+UseG1GC) collector or by the use of the following Java options that will use 16 cores:

```
-XX:+UseParallelGC -XX:ParallelGCThreads=16
```

Some of these settings may be the default in certain environments.

Determining Core Usage on Windows Platforms

To determine how many cores you have available on Windows platforms, use the following resources and facilities:

- *http://www.cpuid.com/cpuz.php*
- msinfo32 at a command prompt (DOS)
- Windows Task Manager

Determining Core Usage on Linux Platforms

To determine how many cores you have available on Linux platforms, use */proc/cpuinfo* or to retrieve a count on logical processors use the following command:

```
grep "^processor" /proc/cpuinfo | wc -l
```

To retrieve a count of physical cores use the following command:

```
grep "cpu cores" /proc/cpuinfo | uniq
```

Chapter 7

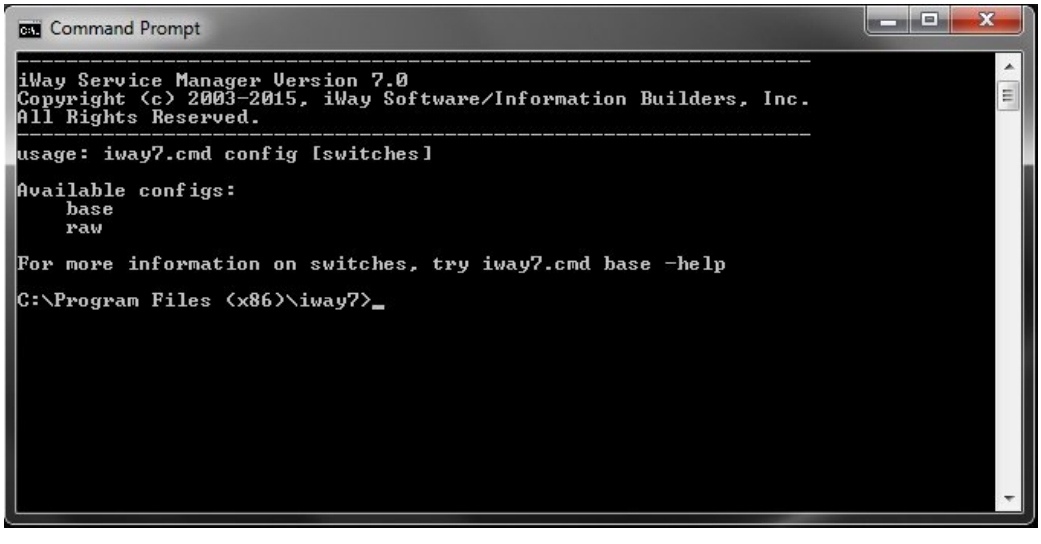
Creating and Using a Remote Command Console

This section describes how to create and use a remote command console in iWay Service Manager (iSM).

In this chapter:

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).



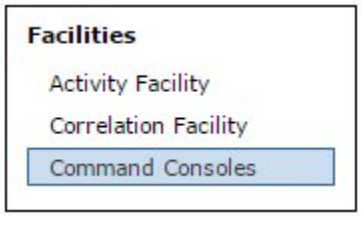
Additionally, commands can be entered using a remote command facility using Telnet (with or without Secure Sockets Layer (SSL)) or Secure Shell (SSH). In either case, the full set of iSM commands is available to the user, depending on the security level at which the logged in user has been granted.

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.

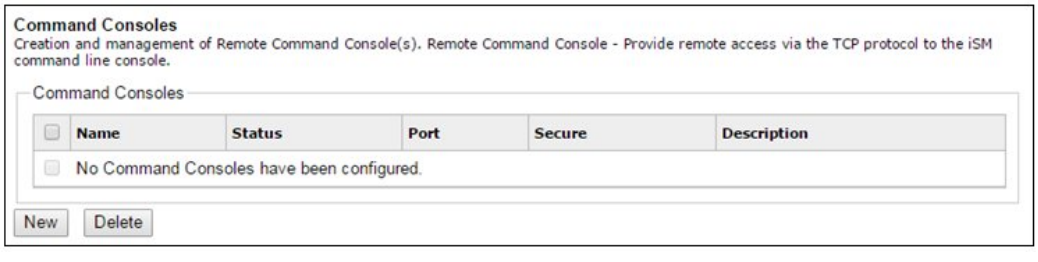
Usually the remote command channel runs off of the base configuration, and the remote command is used to address other running configurations either on the same or another host. A remote command console can be configured to any configuration that is currently running on a host.

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.

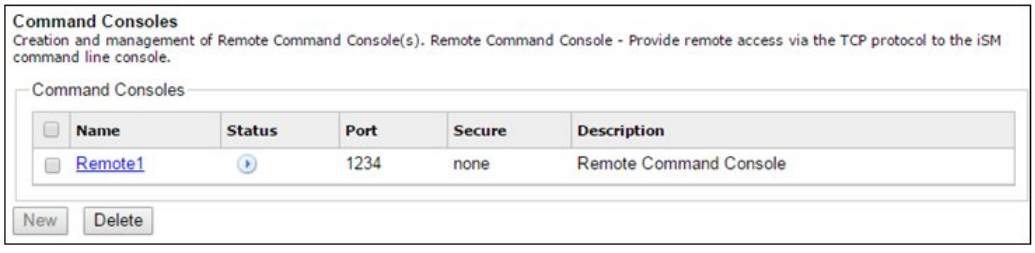


The Command Consoles pane opens, as shown in the following image.



If no remote command consoles have been configured, then the screen will be empty, as currently shown.

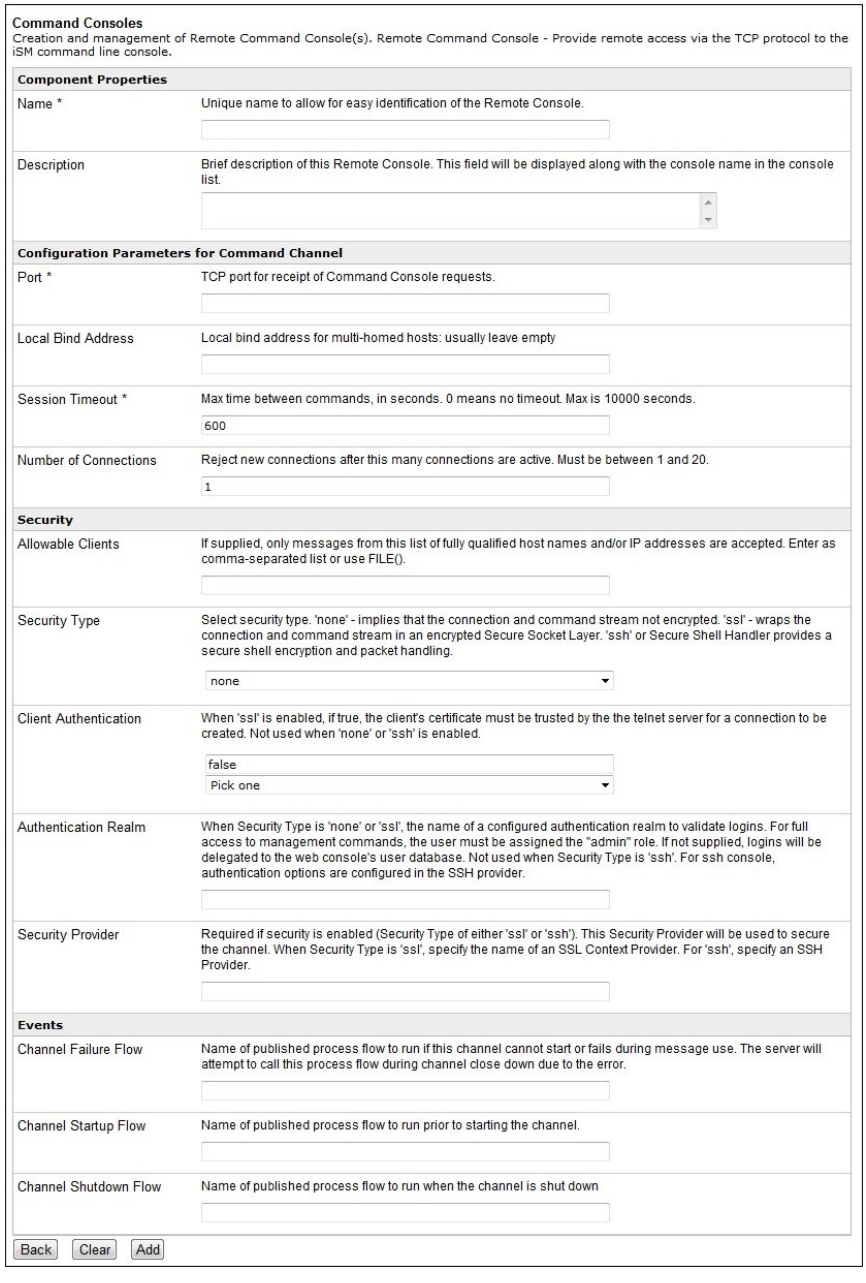
If a remote command console has been configured, then it will be listed in the Command Consoles pane (for example, *Remote1*), as shown in the following image.



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.

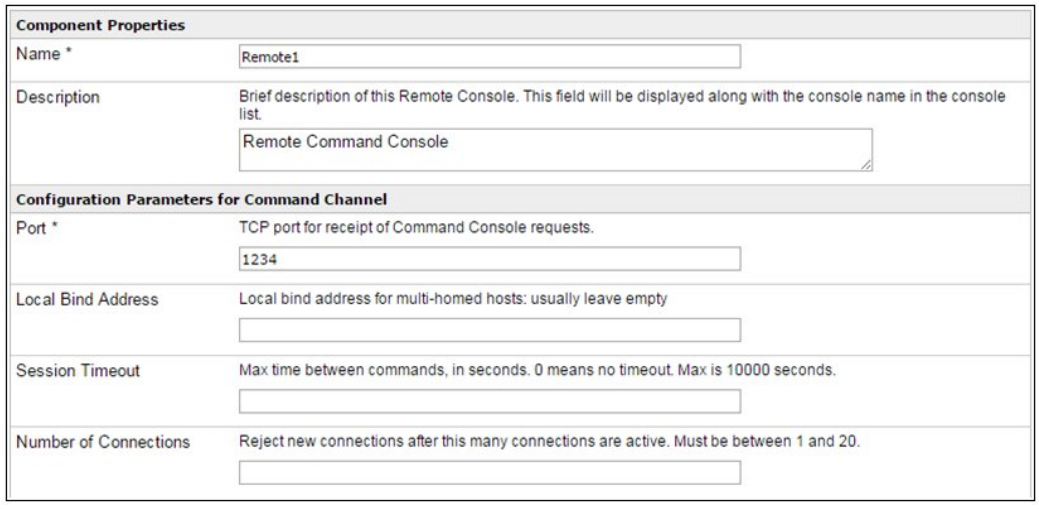
The Command Consoles configuration pane opens, as shown in the following image.



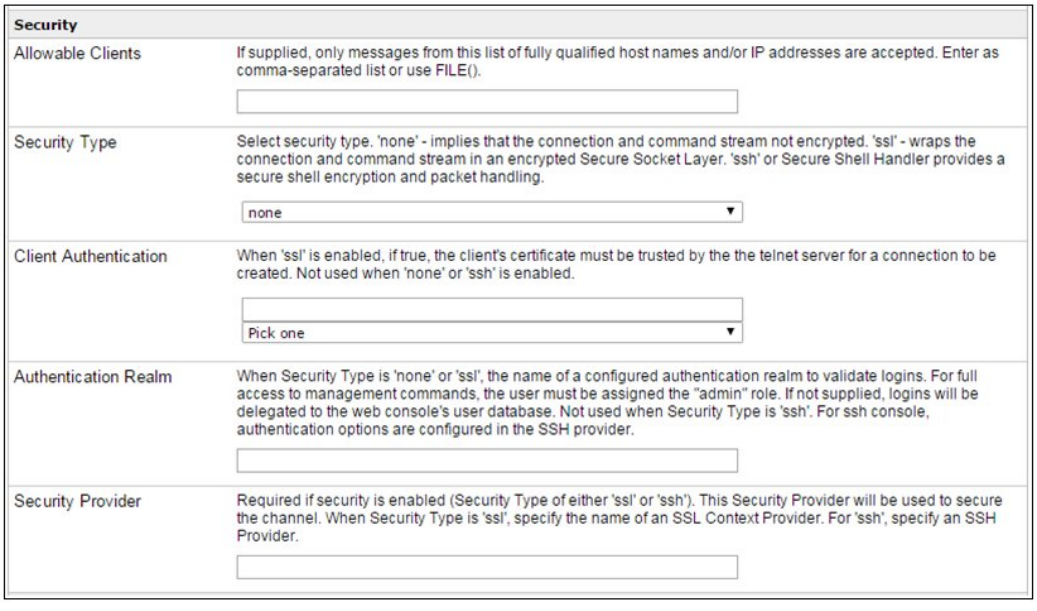
The Command Consoles configuration pane contains a table with the following groups of parameters:

- **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.

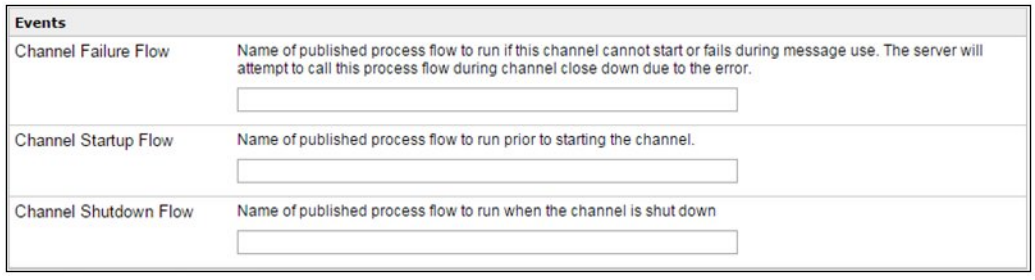


The Security group can be configured as needed. In this case the remote command console will operate using SSH, with a configured realm (for example, LDAP) and an underlying SSH provider. For more information, see the *iWay Service Manager Security Guide*.



Parameter	Applies to Telnet?	Applies to Telnet SSL?	Applies to SSH?
Allowable Clients	Yes	Yes	Yes
Security Type	N/A	N/A	N/A
Client Authentication	No	Yes	No
Authentication Realm	Yes	Yes	No
Security Provider	No	Yes (SSL provider)	Yes (SSH provider)

Events are supported in the Events group, as shown in the following image.



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

Note: An asterisk indicates a required parameter.

Parameter	Definition
Component Properties	
Name*	A unique name that will be used to identify the remote command console.
Description	A brief description for the remote command console, which will also be displayed in the Command Consoles pane.
Configuration Parameters for Command Console	
Port*	TCP port for receipt of Command Console requests.
Local Bind Address	Local bind address for multi-homed hosts: usually leave empty
Session Timeout*	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.
Number of Connections	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.
Security	

Parameter	Definition
Allowable Clients	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.
Security Type	Select one of the following values from the drop-down list: ❏ **none.** Implies that the connection and command stream are not encrypted. ❏ **ssl.** Wraps the connection and command stream in an encrypted Secure Socket Layer (SSL). ❏ **ssh.** Provides secure shell (SSH) encryption and packet handling. The default value selected is *none*.
Client Authentication	If set to *true* and when the Security Type parameter is set to *ssl*, 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 *none* or *ssh*.
Authentication Realm	When the Security Type parameter is set to *none* or *ssl*, the specify the name of a configured authentication realm to validate logins. For full access to management commands, the user must be assigned the *admin* 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 *ssh*. For SSH console, authentication options are configured in the SSH provider.
Security Provider	Required if security is enabled (Security Type parameter value of *ssl* or *ssh*). This security provider will be used to secure the channel. When the Security Type parameter is set to *ssl*, then specify the name of an SSL Context Provider. When the Security Type parameter is set to *ssh*, then specify an SSH Provider.
Events	
Channel Failure Flow	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.

Parameter	Definition
Channel Startup Flow	Name of a published process flow to run prior to starting the channel.
Channel Shutdown Flow	Name of a published process flow to run when the channel is shut down.

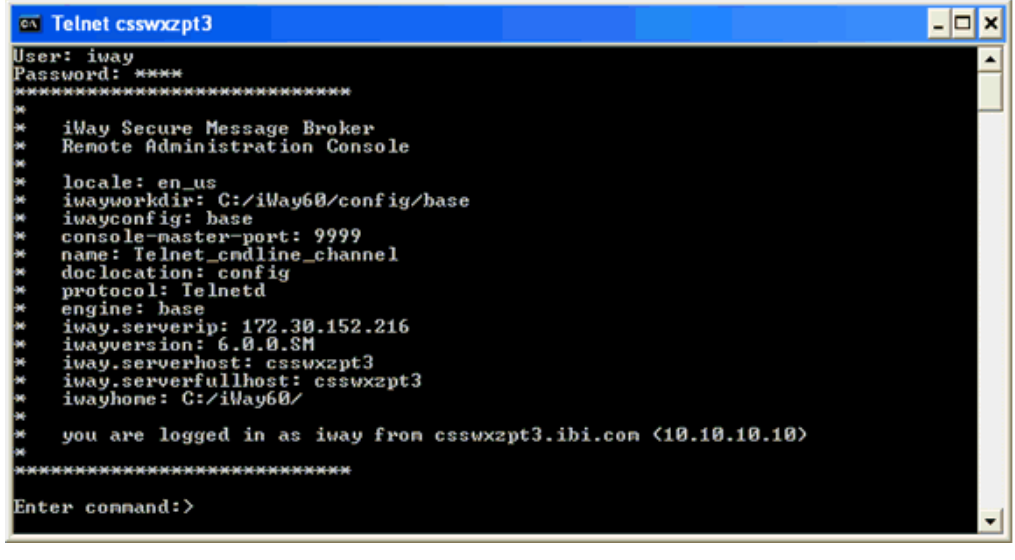
Connecting to a Remote Command Console

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).

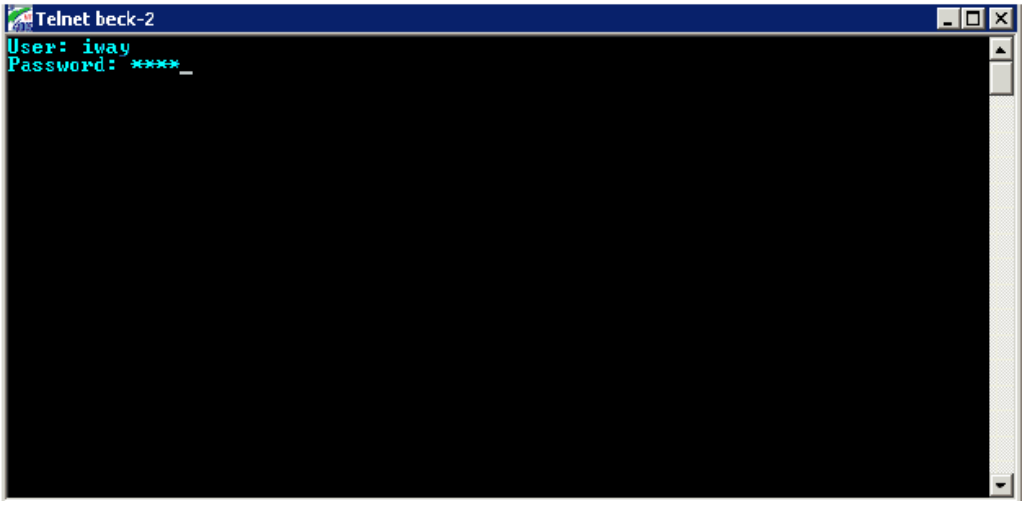


3. Once you are connected and logged in, you can now issue any command to monitor or control your iSM instance.

Using a Telnet Client

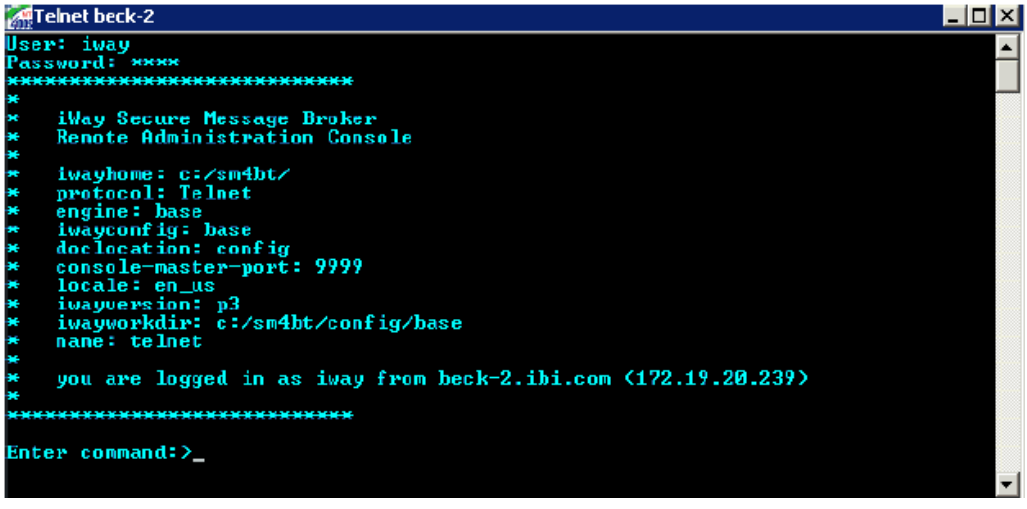
In this section, the default Telnet client that is available on Windows is used for demonstration purposes.

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

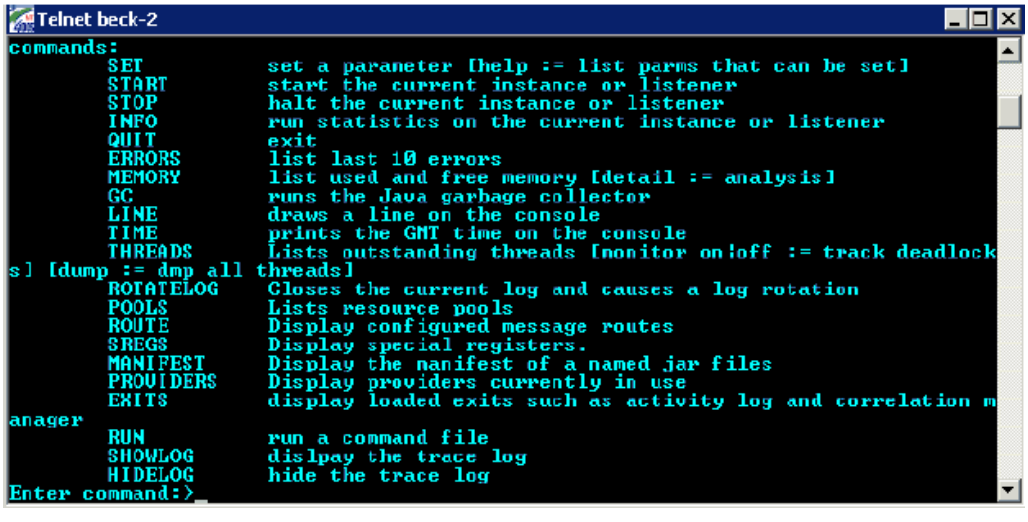


Provided that the connection meets the selected security criteria you are prompted for a user ID and password. These must be configured in the iSM Administration Console, and may have administrative capabilities or not. Lack of administrative capability means that commands that reconfigure iSM, such 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.

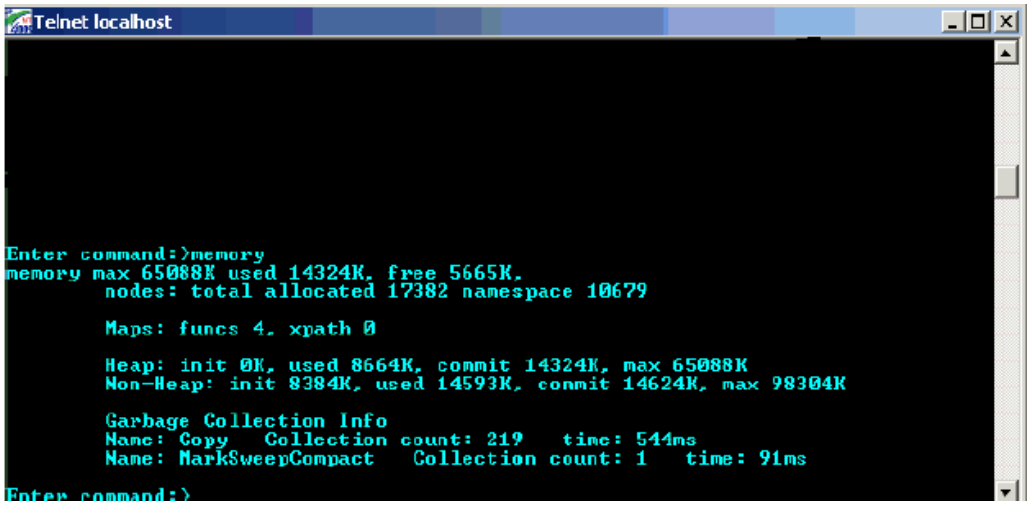


At the command line, you can use any authorized command. The *help* command lists these commands, as shown in the following image.



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.

```
* * * 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 * * *
*
```

Chapter 8

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

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

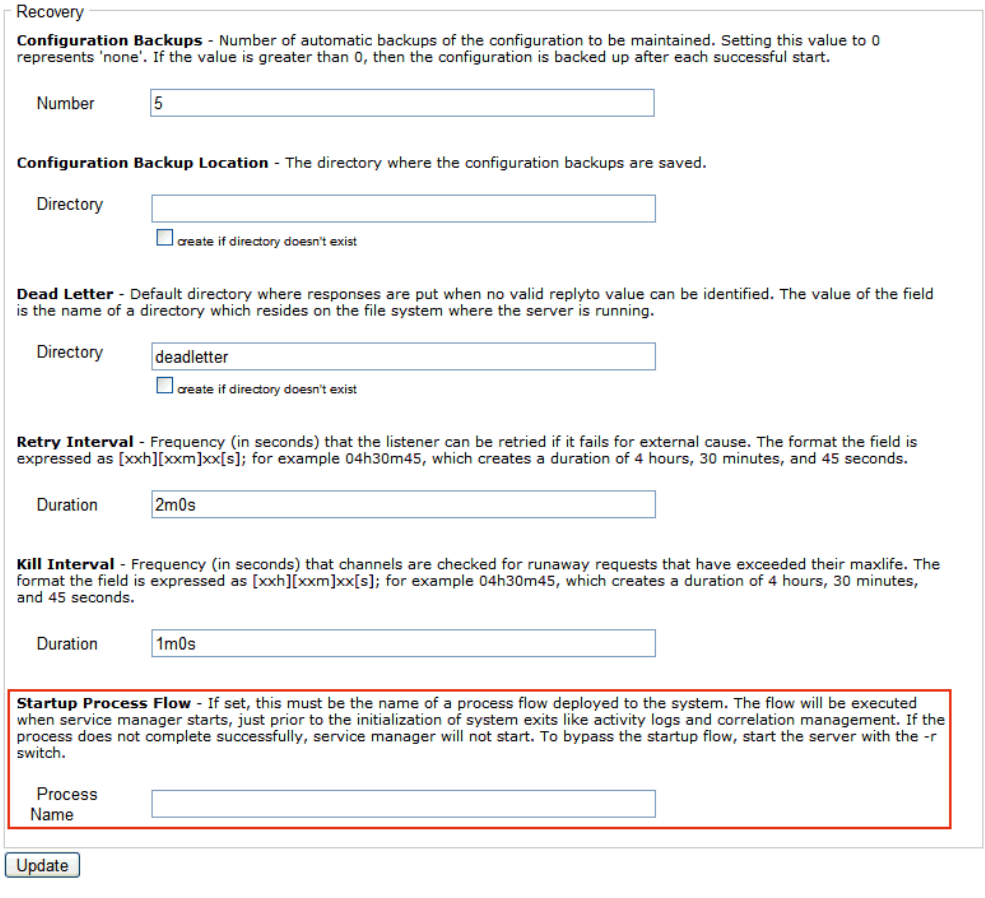
- Send to Dead Letter

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.



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

Edge	Description
success	Continue with iSM startup.
<other> or flow fails	Do not continue to start iSM.

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:

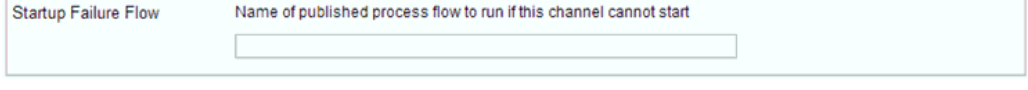
- iSM continues, but BAM update is ignored.
- iSM terminates.
- iSM maintains a local file on disk containing BAM information and attempts to update the database when connectivity is restored.

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.

Note: The Channel Startup Failure process flow feature is available as of iSM Version 6.1.8 and higher.

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:

```
<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.

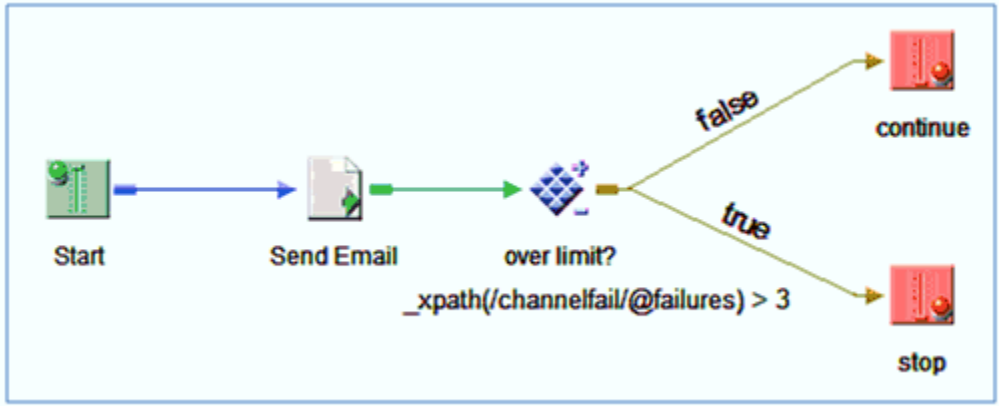
`time`

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).



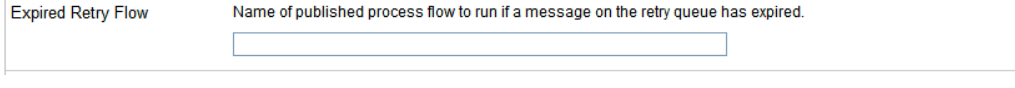
Normally this process flow would run during iSM startup or channel restart. To have the process flow run if the start is attempted from an iSM *start* command whether standalone or in a script, use the *-doflow* switch on the start command. For more information on using the start command, see the *iWay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

Register Name	Description
iway.eventflow.exitflow	Identifies the purpose of the process flow (for example, *expiredRetry*).
iway.eventflow.attempts	Count of the number of retry attempts made before the expiration.
iway.eventflow.expiredtime	Time of the expiration.

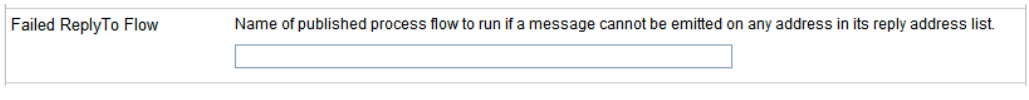
The following table lists and describes the possible edges that are returned by the Retry Expired process flow.

Edge	Description
success	The process flow overrules the expiration. iSM will attempt to resend, this time with the output of the process flow.
<other> or flow fails	An error document is sent to the error addresses.

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.

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

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

Edge	Description
success	The process flow took responsibility to deliver the message.
<other> or flow fails	An error document is sent to the error addresses.

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).

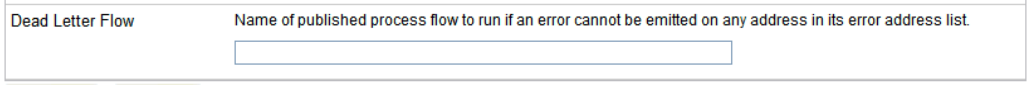
Send to Dead Letter

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.

Register Name	Description
iway.eventflow.exitflow	Identifies the purpose of the process flow (for example, *deadLetter*).
iway.eventflow.replyname	Configured name of the reply or error specification.
iway.eventflow.destination	The address configured for the emit, as evaluated for use.
iway.eventflow.errormsg	An error message, if any, describing the cause of the failure that caused this event to be raised.

Register Name	Description
iway.eventflow.replyprotocol	Protocol used for the emit attempt (for example, File, MQ, and so on).

The following table lists and describes the possible edges that are returned by the Send to Dead Letter process flow.

Edge	Description
success	The message was successfully handled.
<other> or flow fails	The output of the process flow to be written to the dead letter directory, if configured.

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

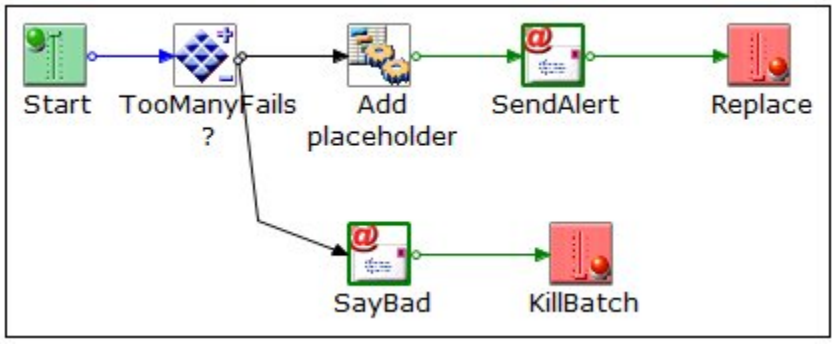
The Parse Failure flow is invoked if an incoming message fails the *parse to XML* operation for a channel. This does not apply to a parse that is handled within a process flow by a service (agent) for that purpose.

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.



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.

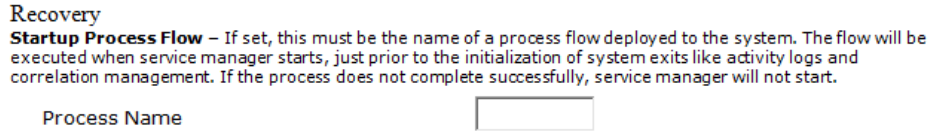
If the process flow ends successfully, the server continues with its startup process. If the process does not end successfully (for example, a fail service is encountered), the server does not start.

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:

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

where:

currentversion

Is the server version number, such as 6.1.6.

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.

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

TIBCO, the TIBCO logo, the TIBCO O logo, FOCUS, iWay, Omni-Gen, Omni-HealthData, and WebFOCUS are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle Corporation and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.