Information Builders

# iWay

## Best Practices for iWay Service Manager (iSM) Installation, Deployment, and Analysis

*iWay Enterprise Enablement Program (EEP)*

August 22, 2018

# Contents

# Preface

This manual is a component of the iWay Enterprise Enablement Program (EEP) for iWay Service Manager (iSM). It is intended for iWay ATS and SA experts to assist customers who are ready to implement and fine tune iSM in their environment.

Visit the iWay and Omni Information Center at *http://ecl.informationbuilders.com/iway/index.jsp* for information regarding:

- Related Publications
- Provide Feedback or Make a Customer Connection
- Information Builders Services & Support

## How This Manual Is Organized

This manual includes the following chapters:

| | Chapter/Appendix | Contents |
|---|---|---|
| 1 | Design and Architecture Best Practices | Outlines and describes key design and architecture best practices for iWay Service Manager (iSM). |
| 2 | Installation and Configuration Best Practices | Outlines and describes key installation and configuration best practices for iWay Service Manager (iSM). |
| 3 | Sizing and Capacity Best Practices | Outlines and describes key sizing and capacity best practices for iWay Service Manager (iSM). |
| 4 | Analysis and Tuning Best Practices | Outlines and describes key analysis and tuning best practices for iWay Service Manager (iSM). |

Chapter 1

# Design and Architecture Best Practices

This chapter outlines and describes key design and architecture best practices for iWay Service Manager (iSM).

**In this chapter:**



## High Availability

High availability, in a general sense, means that systems should be available when needed. Specifically, high availability describes the ability of a system to accept and process transactions at a high percentage of the time, achieving as close to 100% as technically possible. The features and characteristics of a specific software product are not solely responsible for the ability of a system to be highly available. For example, choosing high reliability hardware, and ensuring uninterrupted power, network connectivity, and sufficient capacity and throughput are all essential to achieving high availability.

There are specific architectural mechanisms and design patterns employed to make a system highly available, the most important being failover and scaling. iWay Service Manager (iSM) is compatible with architectures comprising third-party high availability solutions and also has its own native features to facilitate high availability.

Customers must define high availability for their systems, which can be 24/7 or for a specified period. This leads us to the topic of a maintenance window and unplanned downtime.

### Maintenance Window and Unplanned Downtime

Any system can undergo disruptions that are within a maintenance window or unplanned. In all cases, procedures and control measures must be put in place to prevent, detect, and address these situations.

### Maintenance Window Downtime

A maintenance window is a critical consideration when dealing with high availability systems. For instance, when applying Service Packs or Hotfixes, you will need a plan that minimizes disruptions to the end user. In this example, measuring the length of planned downtime has to take into account:

- Advance user notification
- Service Level Agreements
- Testing of Service Packs or Hotfixes
- Rollback plans if testing fails

### Unplanned Downtime

Unplanned events can occur at any time. You can have a hardware failure or a power outage. Regardless of how high availability is defined, the deployment plan must have built-in control measures for preventing, detecting, and addressing unplanned events.

For systems that have high availability defined as 24/7 or for a specified period, there may be some flexibility with the system being unavailable. However, for mission-critical systems, control measures take on an even greater significance. A Disaster Recovery Plan (DRP) must be in place.

## Scaling and Load Balancing

Understanding scaling and load balancing concepts are key during any high availability discussions involving iWay Service Manager.

### Vertical Scaling

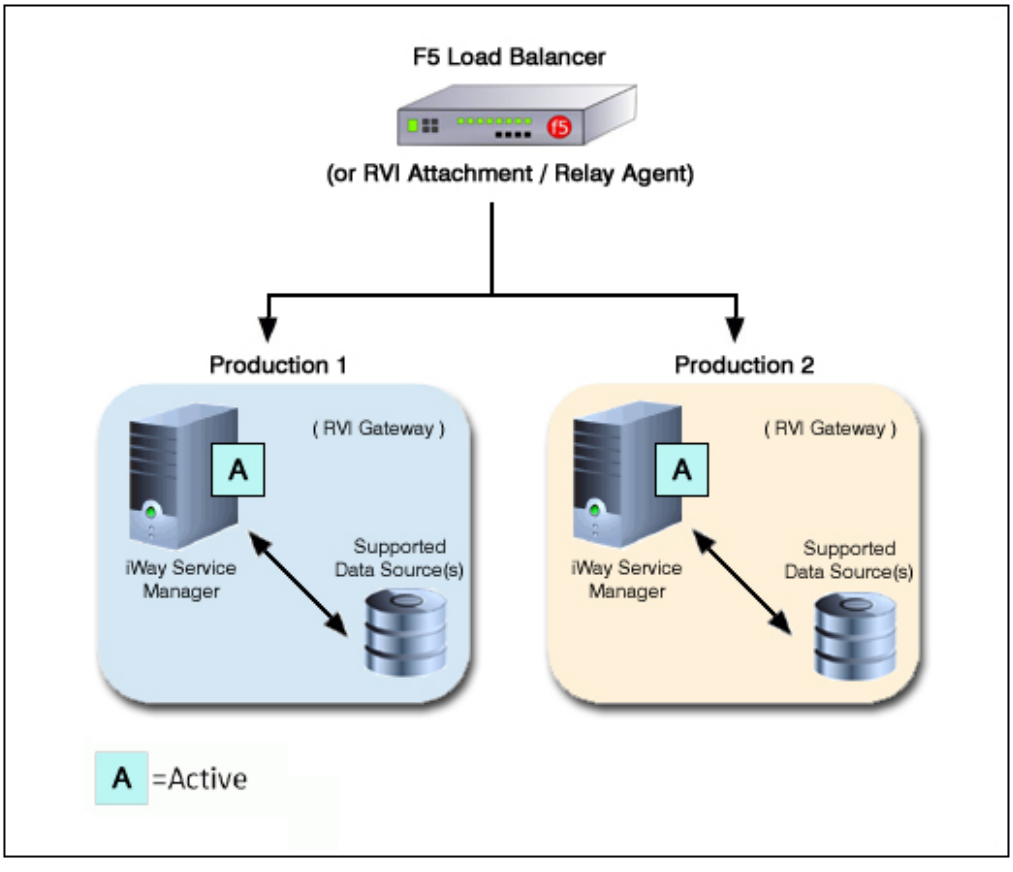
Vertical scaling refers to increasing the processing capability of a host system. This is accomplished by adding processors, memory, faster storage, and so on. Vertical scaling is primarily a hardware effort that does not affect the system topology or software configuration.

### Horizontal Scaling

Horizontal scaling refers to increasing the number of hardware systems hosting the software. For example, two hardware hosts running iWay achieve roughly double the throughput of one, assuming other dependent resources are available and adequately performing. Effectively distributing the workload across two or more iWay Service Manager instances is referred to as *load balancing* and is a key factor in achieving maximum throughput with horizontal scaling. Supporting adequate throughput is an important aspect of high availability, because while a system may be online, if it is running at or close to capacity, it may appear unavailable to clients. Scaling directly addresses the throughput issue, and also provides some of the benefits of failover because it eliminates a single point of failure. Failure of one host (out of two or more) will not make the supported service(s) unavailable, although it may impact throughput and response times until the failed host has *failed over* to its backup or is brought back online.

## Sample Architecture

In the following sample high availability deployment, there are two machines, which are both in *active* mode. Each machine is hosting iWay Service Manager. Since the machines are in *active* mode, they are both actively listening for incoming messages (transactions) and will process all messages concurrently. If one of the machines become unavailable, then all messages will be routed to the remaining machine.




## High Availability Options

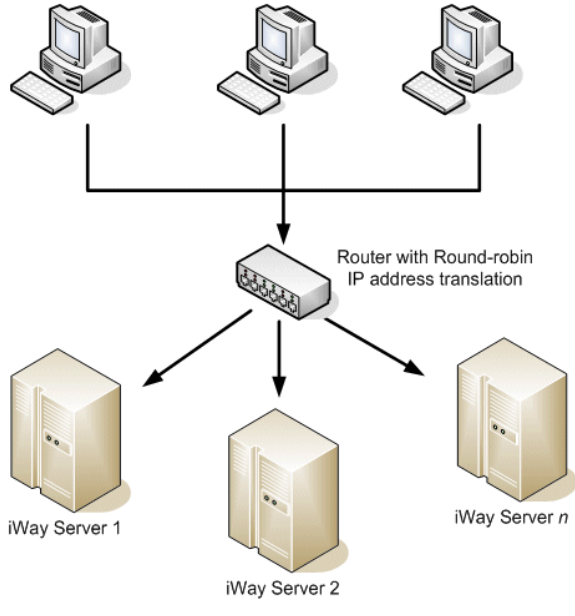
There are a number of strategies and techniques you can use to implement high availability in your iWay Service Manager architecture, including:

- IP-based horizontal scaling
- Web-based horizontal scaling

- Web-based scaling using iWay Performance Monitor
- Horizontal scaling for queuing
- Horizontal scaling and transactions

## IP-Based Horizontal Scaling

IP traffic is very easy to redirect and load balance, and there are very efficient and robust solutions for managing communications at this level in the protocol stack. Because the content of messages is not inspected, this method of work distribution is extremely fast. Devices such as Cisco 7500 series routers can provide round-robin address translation to distribute requests across several identical iWay Service Managers. In the case of a single server stoppage, the router detects the failure and processing continues on the remaining servers. Sharing of iWay repositories (not shown in the diagram) may also be part of this solution. For maximum reliability, each of the iWay instances can have hot backup failover, implemented either via iWay or a third-party tool.



### Web-Based Horizontal Scaling

For web traffic (for example, web services, HTTP), a web router can be used to distribute or load balance across the target iWay Service Managers. Stateful transactions can be supported by the use of *session affinity*.

### Horizontal Scaling for Queuing

An extensive explanation of configuring third-party message queuing products for the high availability environment is beyond the scope of this appendix. All mature queuing products support the configuration options needed to scale horizontally without adversely affecting guaranteed, non-duplicated message delivery. The simplest approaches entail allowing multiple consumers to access a queue and message filtering to balance the load between iWay listeners. If that is inappropriate, stateless horizontal scaling can be achieved by using additional instances of iWay Service Manager (iSM) and redistributing existing clients to these instances. Stateful horizontal scaling is generally achieved by connecting instances of iSM into a cluster, which allows those instances to communicate with each other, as well as to the application clients.

### Horizontal Scaling and Transactions

iWay is optimized for handling stateless processes. Scaling and load balancing may affect the order of processing of messages and may allow a series of related messages to execute on different iWay instances. Because of this, moving to a high availability architecture can reveal idiosyncrasies and/or limit design assumptions in the application. Applications that have implied transactions or implied message order dependence may behave differently in the high availability environment. Note that iWay is not the source of this changed behavior; any middleware deployed for high availability will reveal these types of application flaws. In situations where a web router is part of the iWay high availability solution, enabling session affinity may ensure correct application behavior.

## Understanding Active-Active and Active-Passive Schemes

There are two main types of schemes that support high availability:

- Active-Active
- Active-Passive

**Note:** There are iSM-related licensing considerations that must be discussed and understood for each scheme.

### Active-Active

In an Active-Active scheme, all iWay Service Manager nodes are active and handle requests concurrently.

### Active-Passive

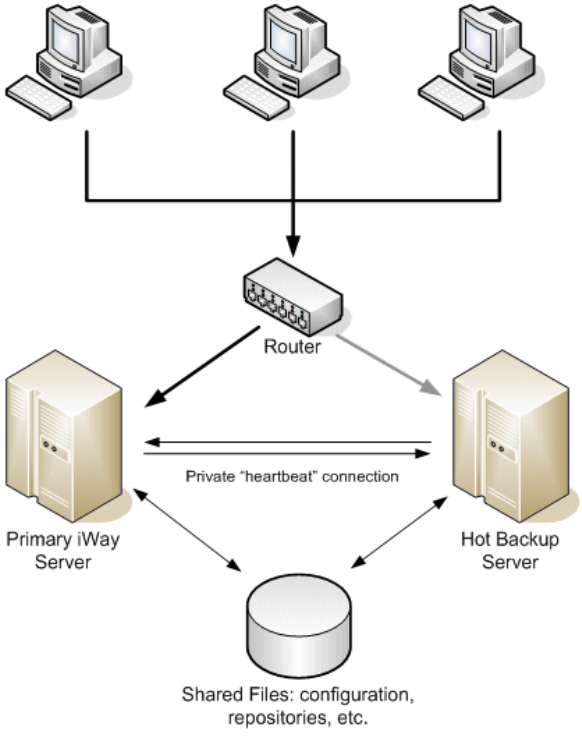
In an Active-Passive scheme, the workload is routed to an active node, while the other passive node waits for a failover to occur. A failover occurs when the active instance becomes inaccessible, and then the passive node is brought online to accept and process requests. The process of switching the roles of the active and passive nodes can be done automatically or manually, in which case there may be a small downtime.

## Fail Over

Failover is the capability to switch over automatically to a redundant or *hot* standby host or subsystem upon the failure or abnormal termination of the primary host or subsystem. Ideally, failover is accomplished without manual intervention since failures generally occur without warning.

## Sample Architecture

In the following sample failover deployment, there are two machines that are hosting iWay Service Manager (iSM). The primary machine is in *active* mode and the backup machine is in *passive* mode. The primary instance of iSM is deployed to automatically *fail over* to the backup instance of iSM on the *hot backup* host. In this sample architecture, configuration and repository files are shared so that the backup iSM behavior is identical to the primary iSM. The failover relies on the native functionality of iWay to emit and respond to *heartbeat* messages which signify normal operation of the primary server. When a failure is detected, the backup host executes a process that manages the switch-over (for example, by sending an appropriate message to the router, posting an email to the SysOp, and so on) and then assumes the workload of the primary server. It should be noted that the primary and backup servers need not be located in the same data center, for example, they may be geographically dispersed.



## Fail Over Options

There are a number of strategies and techniques you can use to implement failover in your iWay Service Manager architecture, including:

- Failover using iWay heartbeat
- Failover using third-party tools

### Failover Using iWay Heartbeat

Simple failover relies on the native functionality of iWay to emit and respond to *heartbeat* messages, which signify normal operation of the primary server. When a failure is detected, the backup host executes a process that manages the switch-over and then assumes the workload of the primary server.

### Failover Using Third-Party Tools

A third-party tool clustering or failover product, such as Veritas Cluster Server, can replace the iWay heartbeat, monitoring, and failover process flow logic. In this case, iWay is unaware of the failover management and is run in stand-alone mode. The topology, configuration, and other requirements will be dictated by the needs of the third-party tool.

## Reverse Invocation (RVI) Gateway

The Reverse Invocation (RVI) Gateway (also known as the iWay RVI Proxy extension) is a design/architecture option that has the ability to distribute transactions to multiple iWay worker instances on other hosts. The workers register themselves with the relay, informing it about which services they (the workers) can provide. Workers may register for mutually exclusive services; workers may register to handle the same services; or workers may do a combination of both, resulting in partially overlapping areas of responsibility.

The proxy is intended for applications where direct connection from the internet/demilitarized zone (DMZ) to the enterprise intranet is not permitted for security reasons. The proxy itself may be horizontally scaled and/or set up to support failover using the mechanisms previously discussed.

To configure RVI Gateway processing, you must:

1. Install the iWay RVI Proxy extension on the iWay Proxy server and the execution engine.

   To install the iWay RVI Proxy, you must add the Gateway extension to your iWay Service Manager (iSM) instance during the iSM installation. For more information on installing iSM, see the *iWay Installation and Configuration Guide*.

After the Gateway extension is installed, the RVIAttach listener, RVIGateway listener, and RVI Relay service are added to the design-time registry and run time configurations.

2. Configure the RVIAttach listener on the iWay Proxy server.
3. Add the RVI Relay service to the appropriate listener(s) configured on the iWay Proxy server.
4. Configure the RVIGateway listener on the execution engine.

iSM horizontal scaling through reverse invocation allows a message received by one iSM configuration to be processed on another configuration. Configurations are expected to be on separate machines, but this is not a requirement. Messages can be distributed over an arbitrary number of associated configurations to balance workload and provide for high availability of processing services.

Messages are received at a receiving engine (the iWay Proxy) and executed at an execution engine. Each message arriving at the iWay Proxy is assigned to a named service. This assignment can be configured in a fixed manner based on the receiving listener or it can be assigned using the full services of iSM intelligent routing services. Regardless of how the assignment is made, the receiving engine locates an execution engine offering the named service, and passes the message to that engine for execution.

Processing engines connect to the receiving engine on a secure, reverse channel. This enables the receiving engine to be located across a firewall, enabling execution to be carried on in a secure environment not open to outside, unauthorized access.

This is also referred to as Reverse Invocation because the execution engine connects to the receiving engine rather than the receiving engine connecting to the execution engine to pass a document.

## Understanding the Proxy Service

Messages arrive at the proxy through any of the protocols that are supported by iWay Service Manager (iSM). Each protocol is managed by a listener. The listener is configured to pass the message to a relay service, which selects an attached execution service and passes the message to the selected engine for execution. All other iSM capabilities are supported. For example, intelligent routing can examine the incoming message to select the appropriate relay service for execution.

## Understanding the Execution Service

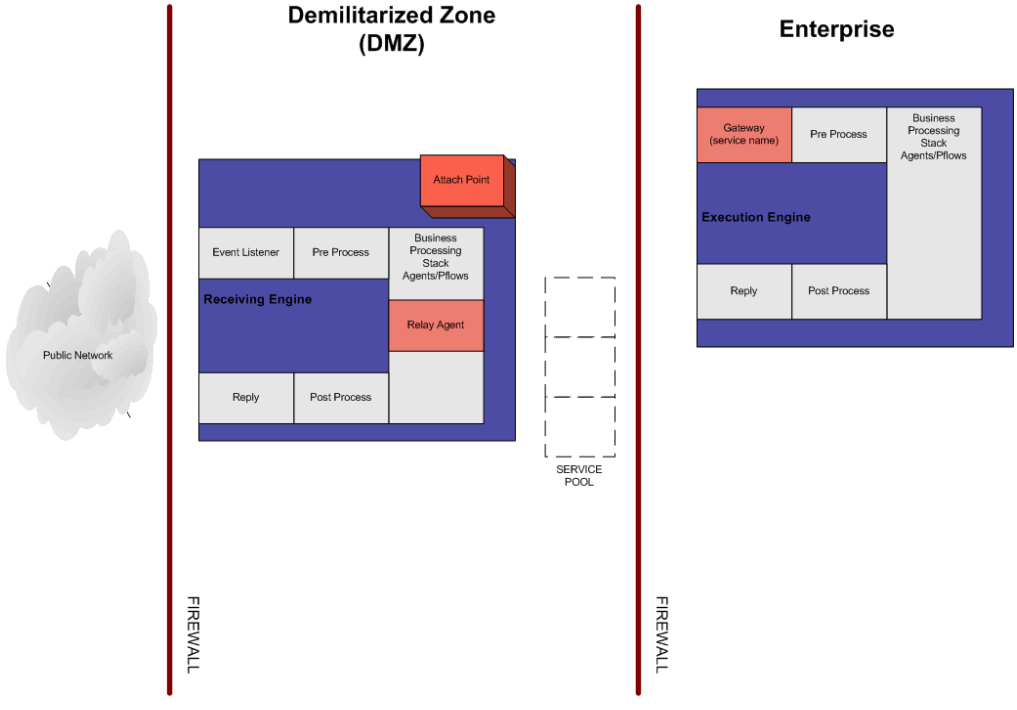
The execution engine accepts relayed messages, executes them, and returns the result to the relay service, which in turn relays the result back to the configured emitter(s). Usually, ancillary emit operations are performed on the execution engines, though this is not required.

An execution engine is configured with one or more gateway listeners. A gateway is a named service that attaches to the attach point of a receiving engine. There must be one gateway for each service name offered, at each receiving engine attach point.

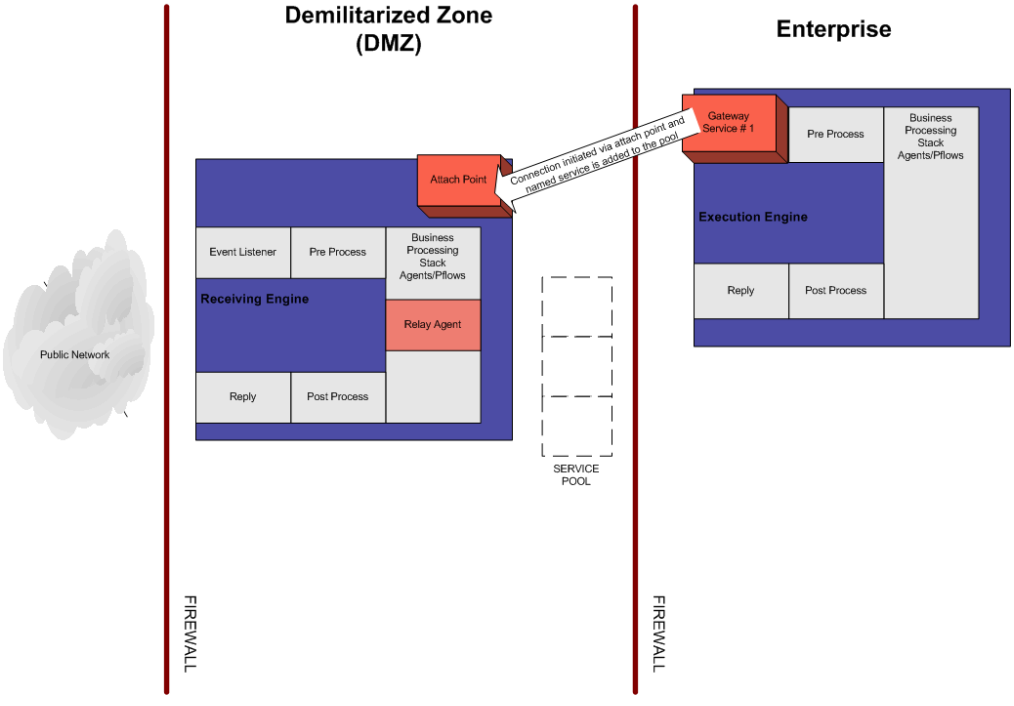
## Understanding the Reverse Invocation Process

This section depicts the reverse invocation process in a step-by-step fashion. In this depiction, iSM is deployed to two locations, one within the enterprise and one in the demilitarized zone (DMZ).

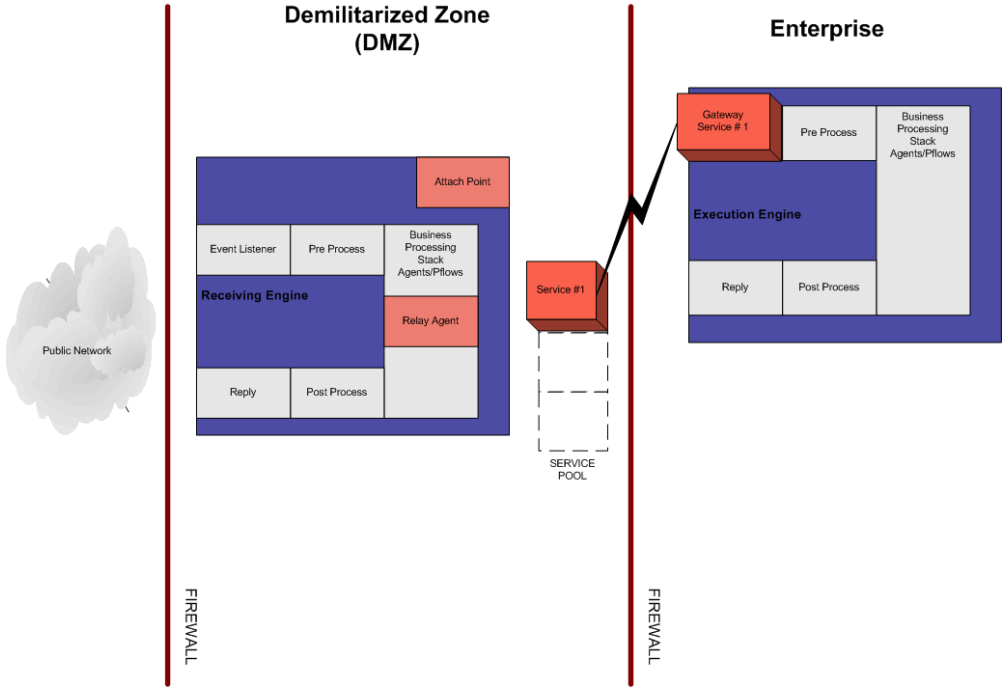
1. The iWay Proxy, or Receiving Engine, starts with the RVIAttach listener waiting for connections to be initiated from the Execution engine, as shown in the following image.



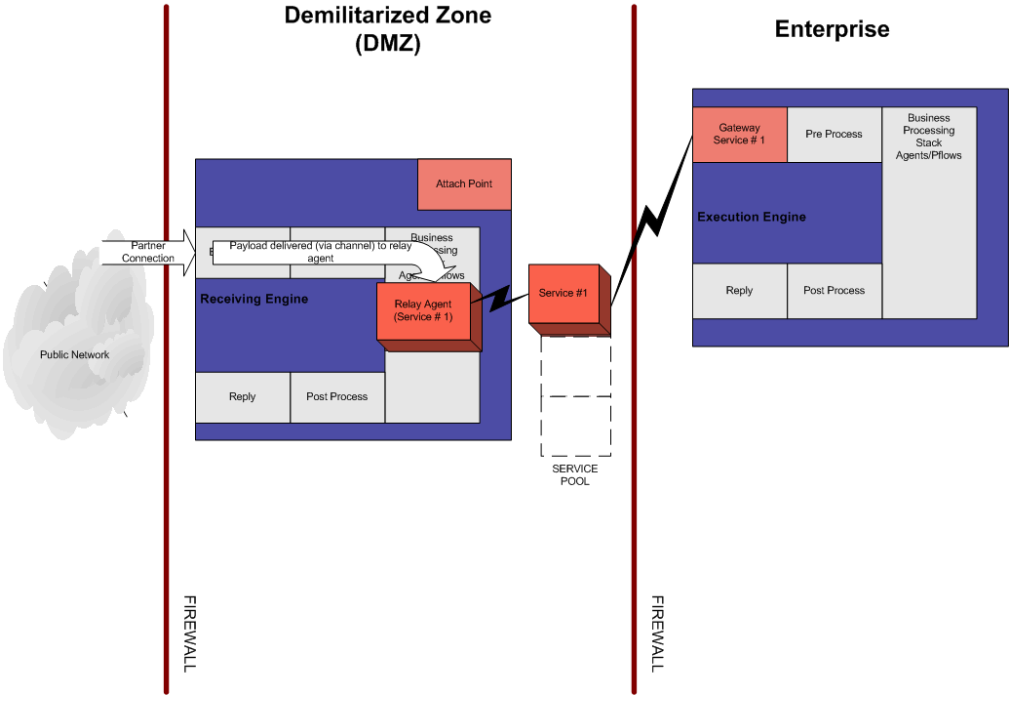
2. The connection is initiated by the Gateway listener configured on the execution engine located in the enterprise, behind the firewall. A service name is defined in the gateway listener configuration, as shown in the following image.



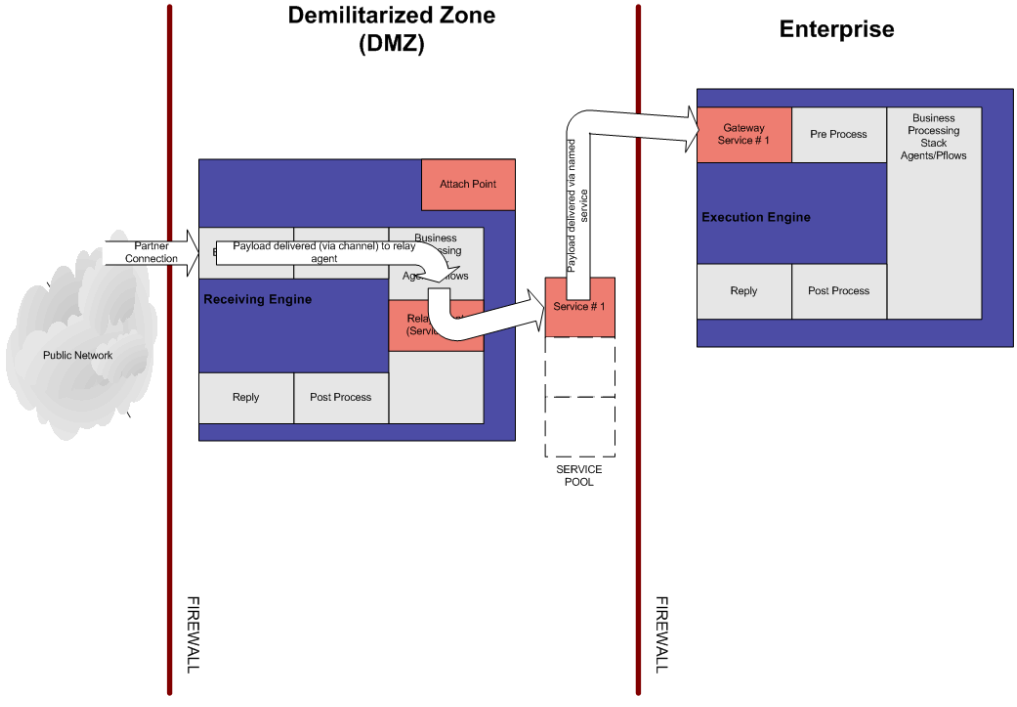
3. After the connection is established, it is added to a pool of connections and can be referenced by the service name, as shown in the following image.



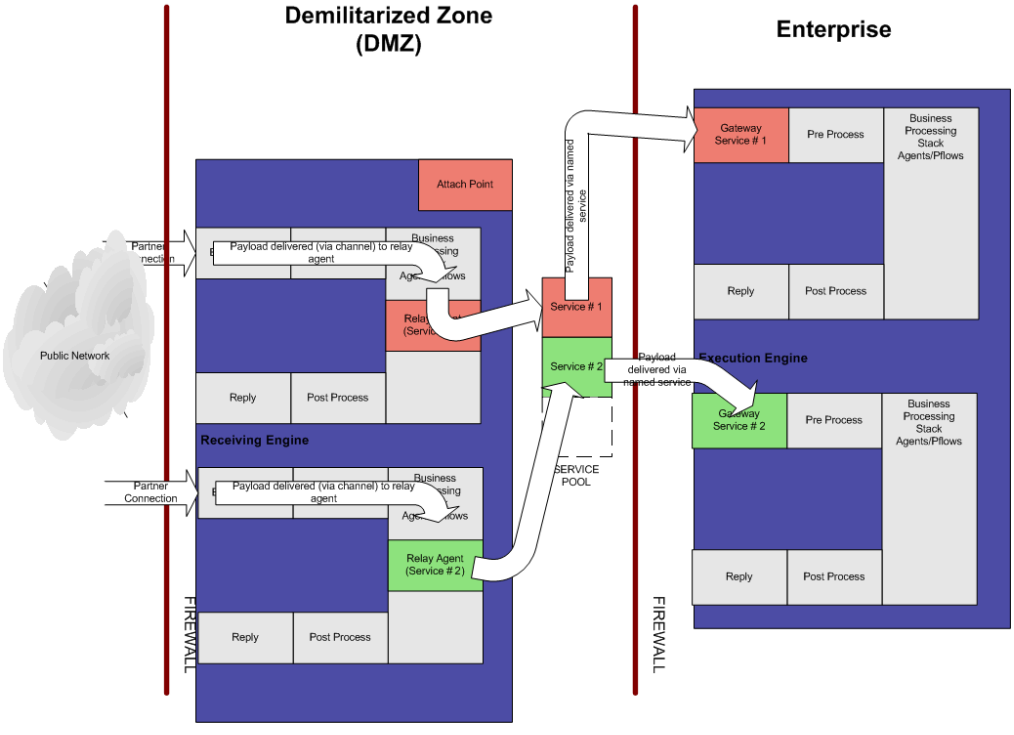
4. When a partner connects to the event listener defined on the iWay Proxy, the message is routed to the execution engine through the relay service that is added to the event listener. The relay service is configured with the service name defined in the gateway listener configuration, as shown in the following image.



5. After the connection between the iWay Proxy and the execution engine is established, messages pass securely through the configuration, as shown in the following image.



6. Multiple channels can be configured in the same way. Gateway listeners configured on the execution engine can spawn services that the iWay Proxy can use to pass data to the configured gateway listeners, as shown in the following image.

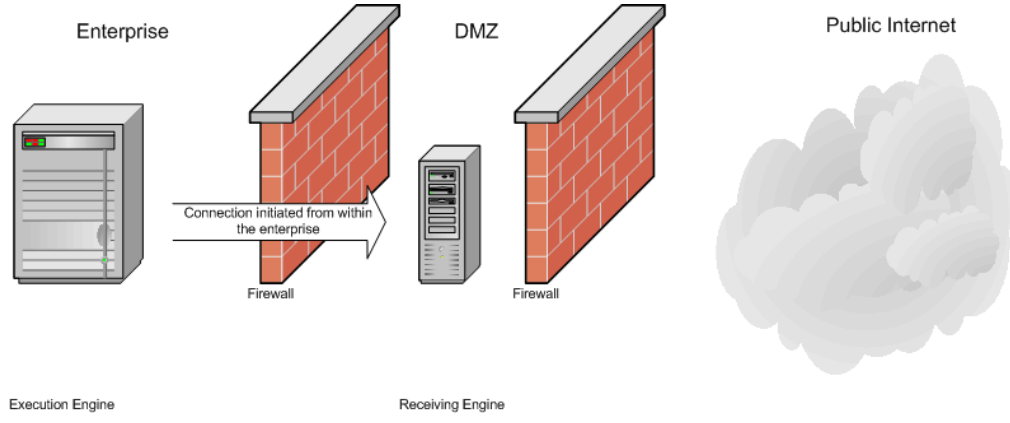



## Sample Architecture

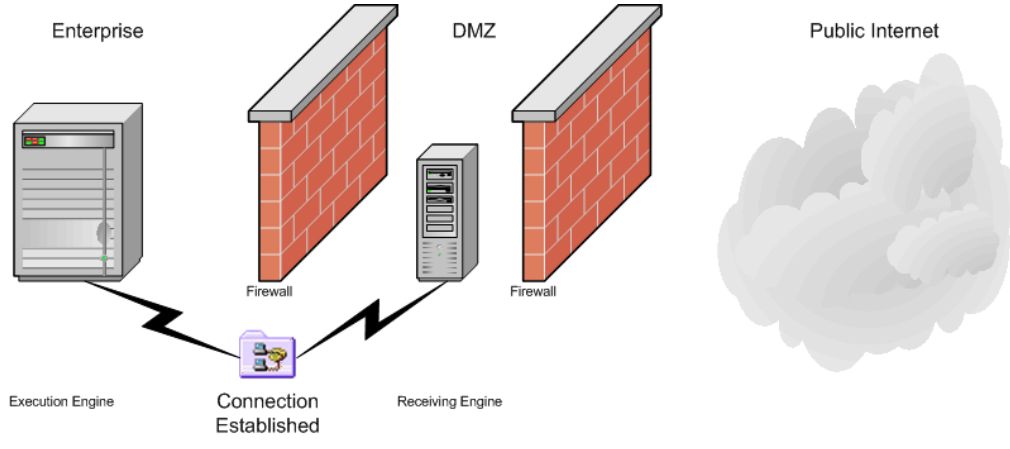
As an example of a Reverse Invocation scenario in which the payload is an EDI document, an AS2 message is routed over the public Internet. The message must be processed securely within the enterprise, where security certificates reside. The iWay Proxy server receives the message securely within the demilitarized zone (DMZ) and passes it back for secure processing to an iSM located inside the enterprise that acts as the Execution engine.

The following diagrams depict the process:

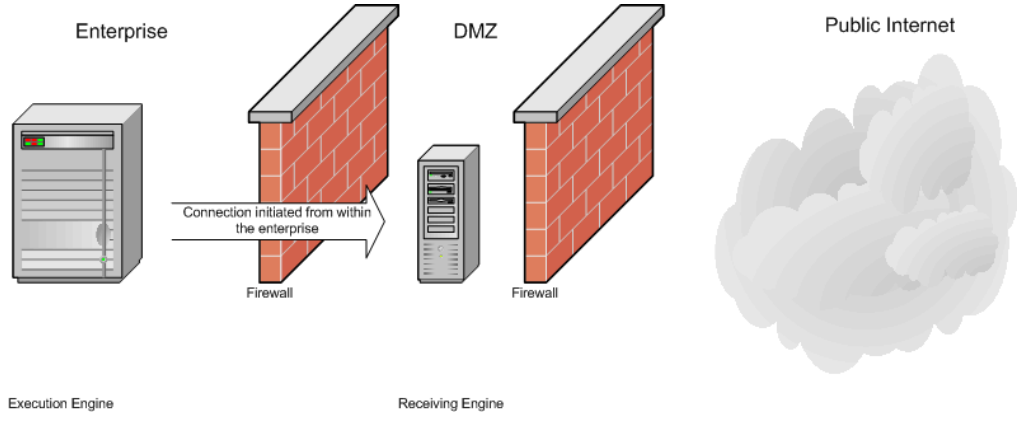
1. The Execution engine initiates a connection with the Receiving Engine (iWay Proxy).



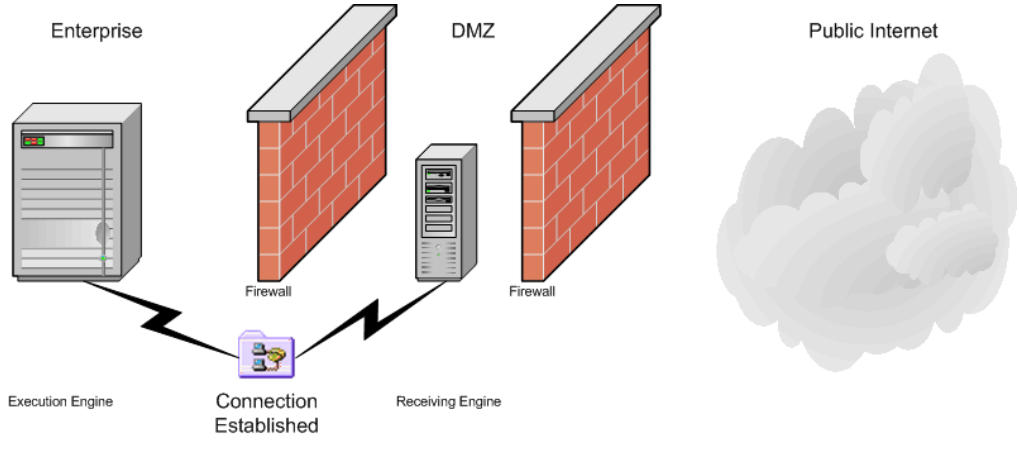

2. The session is established.



3. The trading partner initiates a connection with the iWay Proxy (the receiving engine).




4. The connection is established, and the iWay Proxy manages connectivity between the trading partner and the internal processes hosted by the Execution engine.




From the perspective of a trading partner, a secure connection is established, and information can safely pass through the firewall for secure processing.

## Configuring the iWay RVI Proxy Components

This section describes how to configure and test the iWay RVI Proxy components, which are required for all RVI Gateway implementations.

## Configuring the RVIAttach Listener

Each Relay server maintains a list of attachment points, which is used to direct relayed messages to available Execution engines. Attachment points are characterized by a service name, an IP address, and a port. The Relay service channel is configured with the service name, which is resolved at run time to the target IP and port. Service names should be descriptive, but need not be related to a message type, channel name or host name. Multiple attach points for the same service name may be registered with one relay server, in which case, the connections they represent are assigned to relay events using algorithms which maintain a balanced work distribution (for example, Least Recently Used).

The purpose of the RVIAttach listener is to process attach messages from running Execution Channels in order to construct the attachment point list. The RVIAttach logic maintains the integrity of the attach point list by removing connections which have become unavailable. When that happens, other attach points offering the same named service are not affected.

**Note:** This section describes how to configure an RVIAttach listener. To construct a fully populated iSM channel, incorporate the listener into an inlet and then use the inlet in a channel. For more information on how to design and build a channel, see the *iWay Service Manager User's Guide*.

To configure the RVIAttach listener:

1. Open the iWay Service Manager Administration Console, click *Registry* in the top pane, and then click *Listeners* in the left pane (under Components).
2. From the Listeners (Select listener type) pane that opens, ensure that you select *RVIAttach* as the listener type you are configuring.

The following table lists and describes parameters for the RVIAttach listener.

| Property | Description |
|---|---|
| Port * | The port on which the attach listener is listening to receive service attachments. |
| Local Bind Address | On a server with multiple physical network interfaces, this specifies which interface the listener is bound to. This can usually be left blank. |
| SSL Context Provider | The defined iWay Security Provider for SSL Context. |

| Property | Description |
|---|---|
| Allowable Clients | An optional host name or IP address, which, if entered, limits connections to those from the designated host or IP address. Only one host name or IP address is allowed per RVI Attach listener. If you wish to allow a set of Executor hosts to connect, one RVI Attach listener must be configured for each. |
| Timeout | The frequency with which the attach point checks for stop requests. |
| Keep Alive | The interval at which to poll to ensure that a connection is still available. If an interval is specified, the attach point sends a keep alive message on each attached link. Care should be taken in setting this property, as overly short polling intervals can impact bandwidth and CPU utilization. |

## Configuring the RVI Relay Service

The RVI Relay service is responsible for passing messages to the Executor Server from a channel running on the Proxy Server. To accomplish this, the service uses its service name to find a matching attachment point in the attachment point list. If there are several matching attachment points, the system applies a load balancing algorithm to select which attachment point to use. The RVI Relay service may be configured with the service name property defined as an expression, in which case the expression will be evaluated dynamically for each invocation (for example, for each message which will be relayed) prior to determining the attachment point.

The RVI Relay service can be added to a Channel definition through the iWay Service Manager Administration Console, or may be used in a process flow that is constructed using iWay Designer.

The RVI Relay service is synchronous. Depending on the timeout settings, this service will wait for a response document from the gateway before proceeding. The response document will include content, a header, and user special registers (SREGs). To return a SREG from the gateway, the SREG must be in *message* scope as local and flow scopes are cleared when the process flow running on the gateway ends.

**Note:** This section describes how to configure an RVI Relay service. To construct a fully populated iSM channel, incorporate the service into a process and then include the process as a route of the channel. For more information on how to design and build a channel, see the *iWay Service Manager User's Guide*.

To configure the RVI Relay service:

1. Open the iWay Service Manager Administration Console, click *Registry* in the top pane, and then click *Services* in the left pane (under Components).
2. From the Services (Select service type) pane that opens, ensure that you select *RVI Relay {com.ibi.agents.RVIRelay}* as the service type you are configuring.

The following table lists and describes parameters for the RVI Relay service.

| Property | Description |
|---|---|
| Service Name * | Name of the service that is supported by an Executor Server attach point. Service names should be short and descriptive. Service names are case-sensitive and may not contain punctuation or other special characters. This service name must be identical to the service name that is specified during the configuration of the gateway listener, since it refers to the service offered by the gateway. |
| Tolerance | The period to wait for an Execution server offering the correct service to be available. |
| Timeout | Maximum time period to wait for a response from the executing service. |
| Attempt Retry | If set to true, failed connections to the execution server will be retried. |
| Method of compression to use * | The form of compression that should be used on the output:<br>❑ none (default)<br>❑ smallest<br>❑ fastest<br>❑ standard<br>❑ Huffman |

## Configuring the RVIGateway Listener

The RVIGateway listener offers one service to one attach point. Each active RVIGateway listener offers service attachments to one attach point on a receiving engine. One channel is offered for each possible simultaneous execution. This is configured as the thread count for the listener. The number of offered channels will not grow by demand, although the gateway will attempt to reinstate a failing channel.

To configure the RVIGateway listener:

1. Open the iWay Service Manager Administration Console, click *Registry* in the top pane, and then click *Listeners* in the left pane (under Components).
2. From the Listeners (Select listener type) pane that opens, ensure that you select *RVIGateway* as the listener type you are configuring.

The following table lists and describes parameters for the RVIGateway listener.

| Property | Definition |
|---|---|
| Attach Point Host * | Host address of the attach point, which can be a list such as:<br>`host1:1234;host2:3456(ipi bind address)`<br>The list can also be stored as a file using the iFL _file() function. |
| Attach Point Port * | Socket port where the attach point is listening for gateway connections. This will be the default port, used if a host does not carry the port as host:port. |
| SSL Context Provider | The defined iWay Security Provider for SSL Context. |
| Service Name* | Name of the service that is supported by an Executor Server attach point. The service name is a locator that identifies the channel or listener that runs on the specified machine name. Therefore, it represents a combination of the channel name and the machine and port name for remote invocation. In addition, this is the service name that is referred to in the relay service at the attach point. |
| Reverify time | Period of time (in seconds) to verify the presence of the attach point. |

| Property | Definition |
|---|---|
| Preserve Stream | If set to *true*, an incoming RVI stream message will be processed as a stream document containing the input stream for the message. |
| IP Interface Host | Local IP interface from which the outgoing IP socket originates. This field is usually left blank. |

## Configuring a Service to Test the Reverse Invocation

The gateway listener performs the action requested by the relay service. For example, if a database operation is required to be performed, but the service is available on the gateway machine, the gateway listener picks up the message from the relay service and completes the processing. The result is then returned to the relay service or relay channel that is configured.

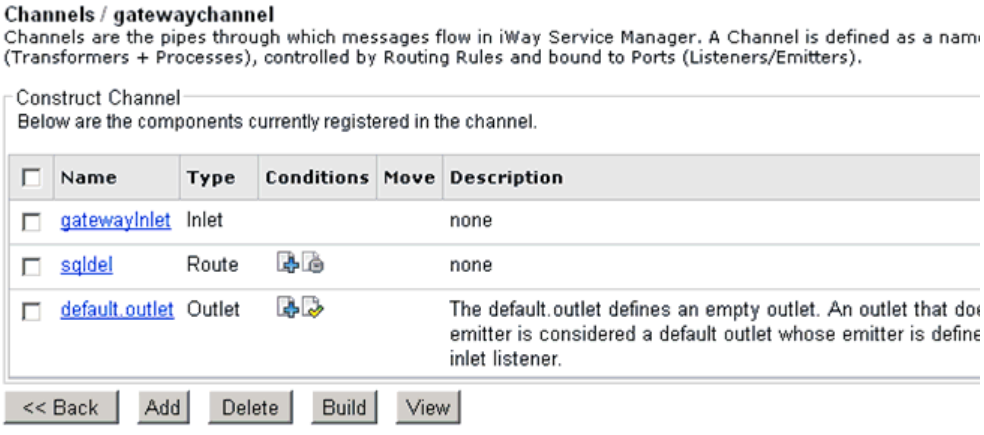
### *Procedure:* How to Create a Service on the Gateway

To create a service on the gateway:

1. Create a gateway listener.
2. Add the listener to an inlet, for example, *gatewayInlet*.
3. Create a service, for example, *sqlServicedel*, which is of type sqlAgent.
4. Create a process, for example, *sqlServicedel*, with the required service to perform the requested action on the gateway.

   This service would be invoked by the relay service on the proxy machine through the socket call. In this case, a service (sqlServicedel) is used to perform a database operation, for example, a delete action through an SQL object.
5. Add the process (sqlServicedel) to a route, for example, *sqldel*.

6. Construct a channel, for example, *gatewaychannel*, as shown in the following image.



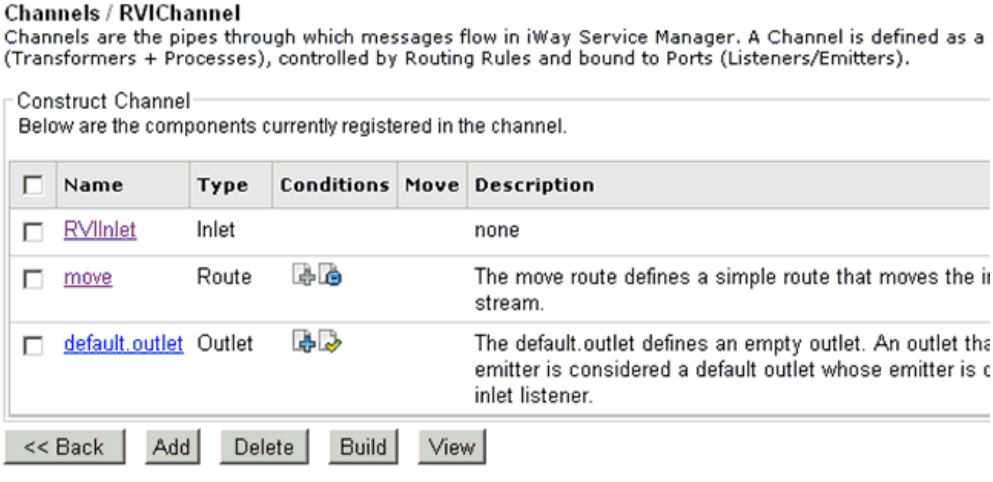

7. Build the channel.

   For more information on how to create a service and constructing channels, see the *iWay Service Manager User's Guide*.

## *Procedure:* How to Configure the RVIAttach Channel

To configure the RVIAttach channel:

1. Create an RVIAttach listener.
2. Construct a channel to perform the initial handshake with the gateway channel, as shown in the following image.

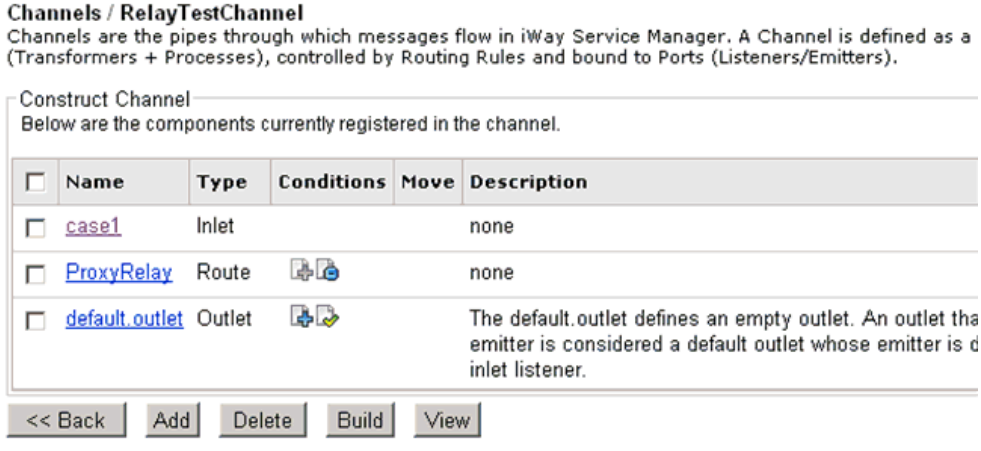


## *Procedure:* How to Configure the Channel to Invoke the Remote Gateway Service

As an example, assume that a channel exists with a file listener that picks up files from a specified directory. After the file is picked up, a service on the gateway is invoked through the attach point and the result is written to an output directory.

To configure the channel to invoke the remote gateway service:

1. Create a file listener.
2. Add the listener to an inlet, for example, *case1*.
3. Create a route, for example, *ProxyRelay*, which includes a relay service (RVIProxyService).
4. Construct a channel, for example, *RelayTestChannel*, to test the remote service on the gateway machine, as shown in the following image.




## *Procedure:* How to Test the RVI Invocation Using the Attach Point and Gateway

To test the RVI invocation using the attach point and gateway:

1. Build the channel, for example, *RVIChannel*.
2. Start the channel.
3. If the RVIChannel starts without any errors, start the channel (gatewaychannel) on the gateway machine.
4. If the channel (gatewaychannel) starts without any errors, a successful connection between the attach point and the gateway has been made.
5. Start the channel (RelayTestChannel) to invoke the service.
6. Place a file in the input path directory that was configured for the file listener (associated with the case1 inlet) to start the invocation process.

The file read is successful indicating a success test run on the RVIAttach side. To see if the gateway service was invoked successfully, check the database to see if the database operation was completed successfully on the gateway side. If the database operation was completed, this indicates that the gateway service ran successfully.

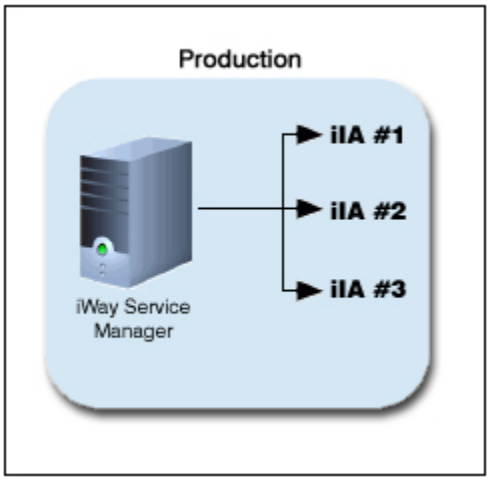
## Standalone Configuration

iWay Service Manager (iSM) is based on a concept of managed servers or configurations. When starting iSM, the *base* configuration is invoked as the *master configuration*. Essential to the internal processing of iSM, its console and deployment architecture, consequently any newly deployed configuration depends on the master configuration. Therefore, it is important never to delete the base configuration nor to deploy newly designed channel architectures against this configuration and use it in a production setting.

The standalone deployment architecture's method of choice is the usage of iWay Integration Applications (iIAs). iIAs provide an automated, scriptable, deployment toolset that enables you to build and deploy integration solutions (channels, process flows, services, and so on) into configurations, thus in a dedicated runtime environment.

### Sample Architecture

In the following sample standalone configuration, there is one machine that is hosting iWay Service Manager. Three iWay Integration Applications (iIAs) are configured and deployed. Each iIA serves as an independent managed configuration where an application is deployed to process incoming messages (transactions).



Chapter 2

# Installation and Configuration Best Practices

This chapter outlines and describes key installation and configuration best practices for iWay Service Manager (iSM).

**In this chapter:**



## Selected Architecture

This section describes best practices for single-tier and multi-tier architectures in reference to endpoints.

### Single-tier in Reference to Endpoints

A single-tier iWay Service Manager (iSM) install, in reference to endpoints, indicates that the act of consuming or emitting documents involves Services, Protocols or DISK on the same host machine. Examples of some endpoints would be FTP, FILE, TCP, HTTP and RDBMS. While DISK I/O is expensive, when it comes to performance, iSM should not be greatly impacted. When possible, it is always a best practice to have a resource-intensive endpoint, such as a RDBMS, on a standalone system.

### Multi-tier in Reference to Endpoints

As discussed earlier, a multi-tier install with iWay Service Manager (iSM) and a RDBMS is considered a best practice to avoid competition for resources. All RDBMS types, such as Oracle and MS SQL, are extremely resource-intensive and should be hosted on a robust server capable of handling these loads.

## Installation

This section describes installation best practices for iWay Service Manager (iSM).

## Minimum Recommended Hardware Requirements

iWay Service Manager (iSM) requires the following minimum hardware specifications on a Windows platform:

- 1 gigahertz (GHz) or higher Intel® Pentium® compatible CPU.
- 1 gigabyte (GB) of RAM.
- 1 GB of disk space after installation (2 GB during installation).

For other platforms, ensure that your machine has a reasonable supply of resources. For exact requirements, contact iWay Software Customer Support.

## Minimum Recommended Java Requirements

Java 2 Standard Edition (J2SE™) JDK 1.7 or higher is required for iWay components.

For Windows, Linux, and Solaris, you can download and install the latest supported JDK at no charge from:

`http://www.oracle.com/technetwork/java/javase/downloads/index.html`

For other platforms, contact the appropriate vendor.

After installing the JDK, the Java command must be in your search PATH to install and run iWay. On Windows platforms, the JDK installation normally handles this for you by placing a copy of java.exe in C:\WINNT\system32. On other platforms, ensure the following is in your PATH variable:

`/`*`java_home`*`/bin`

where:

*`java_home`*

Is the absolute path where the JDK is installed.

To determine if Java is properly installed and in your search PATH, execute the following at a command prompt or shell:

```
java -version
```

Information on the Java build appears, for example:

```
java version "1.7.0_06"
Java(TM) SE Runtime Environment, Standard Edition (build 1.7.0_06-b09)
Java HotSpot(TM) Client VM (build 25.2-b09, mixed mode, sharing)
```

**Note:** The terms JDK™ and Java SDK™ are synonymous.

## Understanding Licensing

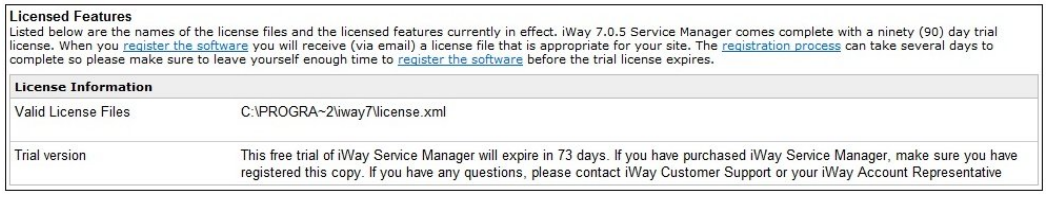
iWay Service Manager (iSM) uses a license file to validate the usage of purchased components. The license file is located under the iSM root directory.

### Temporary

Each iSM installation includes a 90-day trial period, which is also referred to as the *temporary* license. This license allows users to work with all of the installed iSM features. It is also possible to evaluate iSM components that may be installed at a later point in time.

To view iSM license information at any point, click *Licenses* on the top pane, as shown in the following image.


The Licensed Features page opens, as shown in the following image.



**Licensed Features**

Listed below are the names of the license files and the licensed features currently in effect. iWay 7.0.5 Service Manager comes complete with a ninety (90) day trial license. When you register the software you will receive (via email) a license file that is appropriate for your site. The registration process can take several days to complete so please make sure to leave yourself enough time to register the software before the trial license expires.

| License Information | |
|---|---|
| Valid License Files | C:\PROGRA~2\iway7\license.xml |
| Trial version | This free trial of iWay Service Manager will expire in 73 days. If you have purchased iWay Service Manager, make sure you have registered this copy. If you have any questions, please contact iWay Customer Support or your iWay Account Representative |

It is the responsibility of the user to apply for a permanent iSM license.

### Permanent

For more information about registering iWay Service Manager (iSM) to obtain a permanent license, see *Registering iWay Service Manager* in Chapter 2 of the *iWay Service Manager User's Guide*.

Once a permanent license has been received, all purchased components are validated and all non-purchased components are granted another 90-day trial period. If additional components have been purchased, please re-apply for a new permanent license.

**Note:** It is important for iSM to be installed with the correct iWay customer site code. Occasionally during iSM installation, the site code 9999 may be entered. If you think this is the case, edit the install.xml file, which is located in the <*iway_home*>\bin subdirectory. If an incorrect site code is found in the <sitecode> element, you may continue using iSM. However, open a HOT TRACK case with iWay Customer support and specify the correct site code value that should be used. A valid site code contains four alphanumeric values followed by a period (.) and terminated by two numeric values.

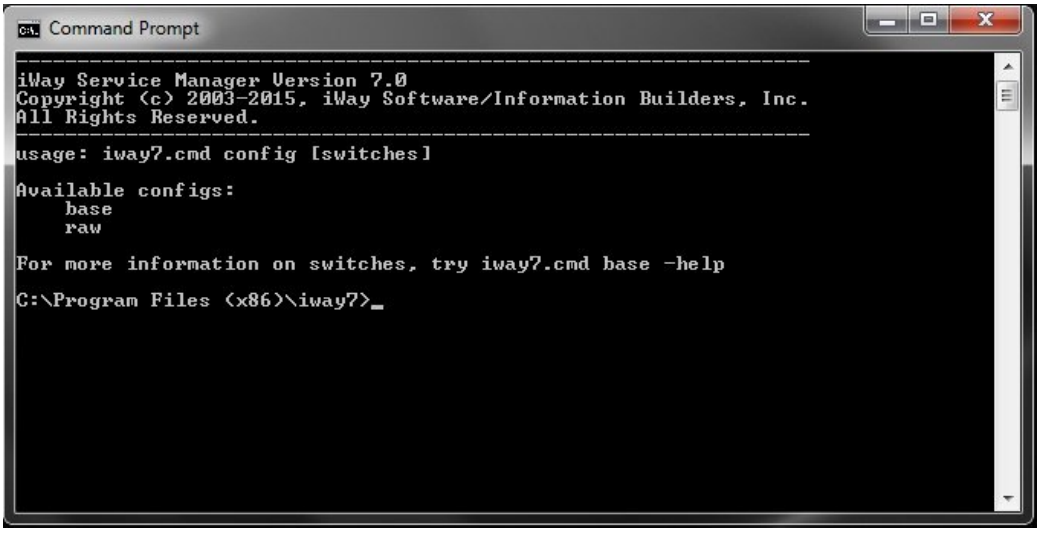
When installing the permanent license, ensure that the temporary license is overwritten. In addition, ensure that there is only one license file in the root directory where iSM is installed.

## Configuration

This section describes configuration best practices for iWay Service Manager (iSM).

### Creating and Using a Remote Command Console

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).



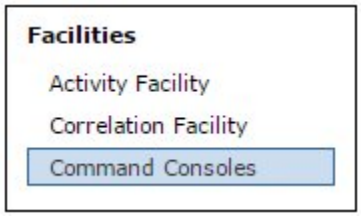

Additionally, commands can be entered using a remote command facility using Telnet (with or without Secure Sockets Layer (SSL)) or Secure Shell (SSH). In either case, the full set of iSM commands is available to the user, depending on the security level at which the logged in user has been granted.

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.

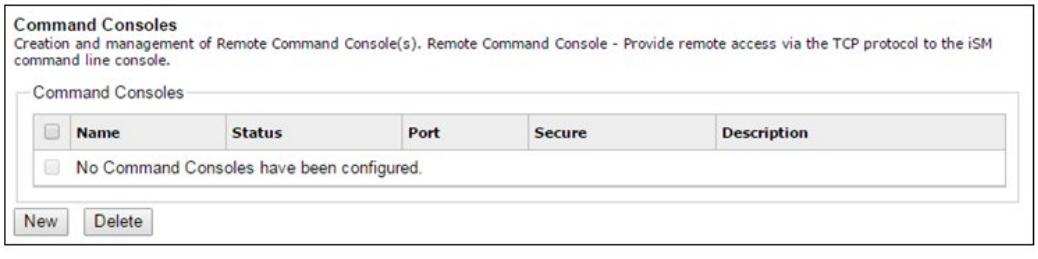
Usually the remote command channel runs off of the base configuration, and the remote command is used to address other running configurations either on the same or another host. A remote command console can be configured to any configuration that is currently running on a host.

## 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.

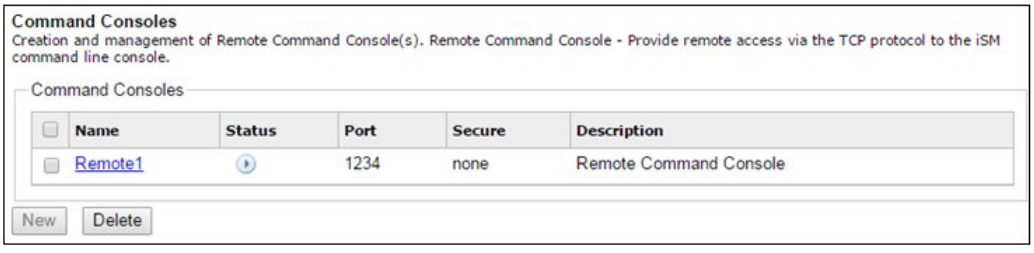



The Command Consoles pane opens, as shown in the following image.




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.

**Command Consoles**
Creation and management of Remote Command Console(s). Remote Command Console - Provide remote access via the TCP protocol to the iSM command line console.

| | |
|---|---|
| **Component Properties** | |
| Name * | Unique name to allow for easy identification of the Remote Console. |
| Description | Brief description of this Remote Console. This field will be displayed along with the console name in the console list. |
| **Configuration Parameters for Command Channel** | |
| Port * | TCP port for receipt of Command Console requests. |
| Local Bind Address | Local bind address for multi-homed hosts: usually leave empty |
| Session Timeout * | Max time between commands, in seconds. 0 means no timeout. Max is 10000 seconds. 600 |
| Number of Connections | Reject new connections after this many connections are active. Must be between 1 and 20. 1 |
| **Security** | |
| Allowable Clients | 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 FILE(). |
| Security Type | Select security type. 'none' - implies that the connection and command stream not encrypted. 'ssl' - 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. none |
| Client Authentication | When 'ssl' is enabled, if true, the client's certificate must be trusted by the the telnet server for a connection to be created. Not used when 'none' or 'ssh' is enabled. false Pick one |
| Authentication Realm | When Security Type is 'none' or 'ssl', 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 Security Type is 'ssh'. For ssh console, authentication options are configured in the SSH provider. |
| Security Provider | 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 'ssl', specify the name of an SSL Context Provider. For 'ssh', specify an SSH Provider. |
| **Events** | |
| Channel Failure Flow | Name of 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. |
| Channel Startup Flow | Name of published process flow to run prior to starting the channel. |
| Channel Shutdown Flow | Name of published process flow to run when the channel is shut down |

Back Clear Add

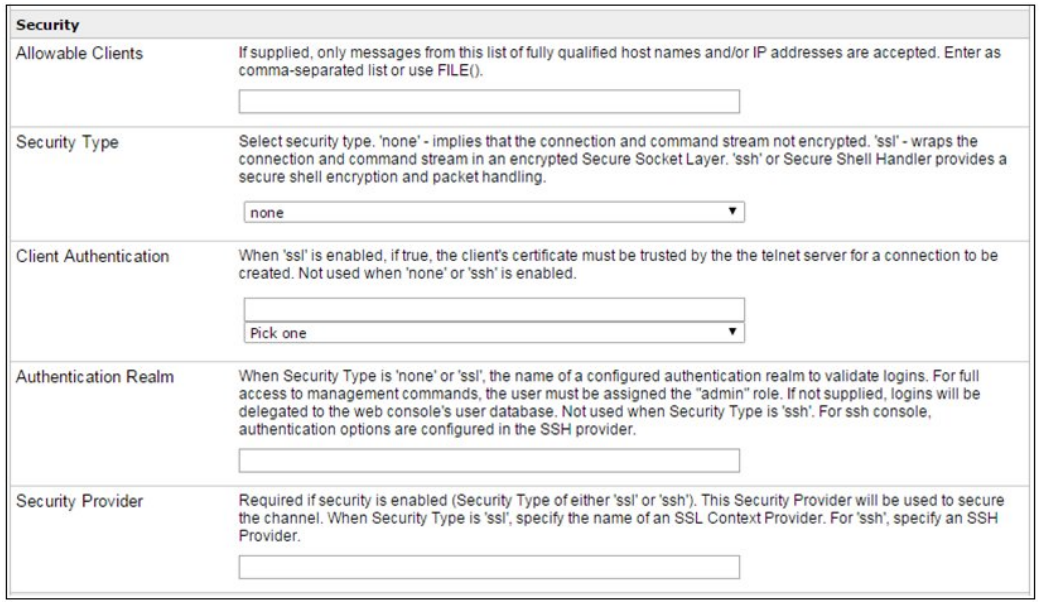
The Command Consoles configuration pane contains a table with the following groups of parameters:

- **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.

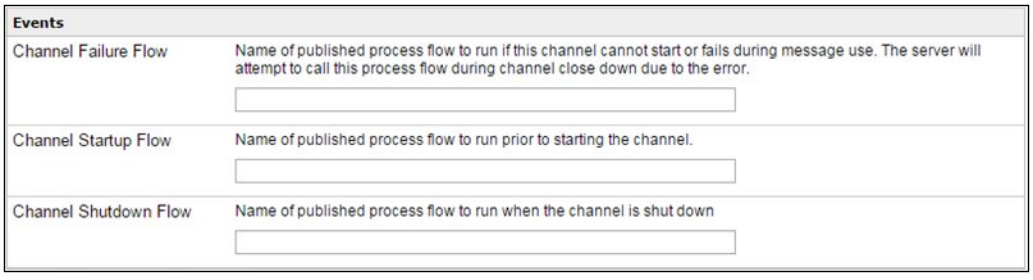
| **Component Properties** | |
|---|---|
| Name * | Remote1 |
| Description | Brief description of this Remote Console. This field will be displayed along with the console name in the console list.<br>Remote Command Console |
| **Configuration Parameters for Command Channel** | |
| Port * | TCP port for receipt of Command Console requests.<br>1234 |
| Local Bind Address | Local bind address for multi-homed hosts: usually leave empty |
| Session Timeout | Max time between commands, in seconds. 0 means no timeout. Max is 10000 seconds. |
| Number of Connections | Reject new connections after this many connections are active. Must be between 1 and 20. |

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:<br>❑ **none.** Implies that the connection and command stream are not encrypted.<br>❑ **ssl.** Wraps the connection and command stream in an encrypted Secure Socket Layer (SSL).<br>❑ **ssh.** Provides secure shell (SSH) encryption and packet handling.<br>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. |

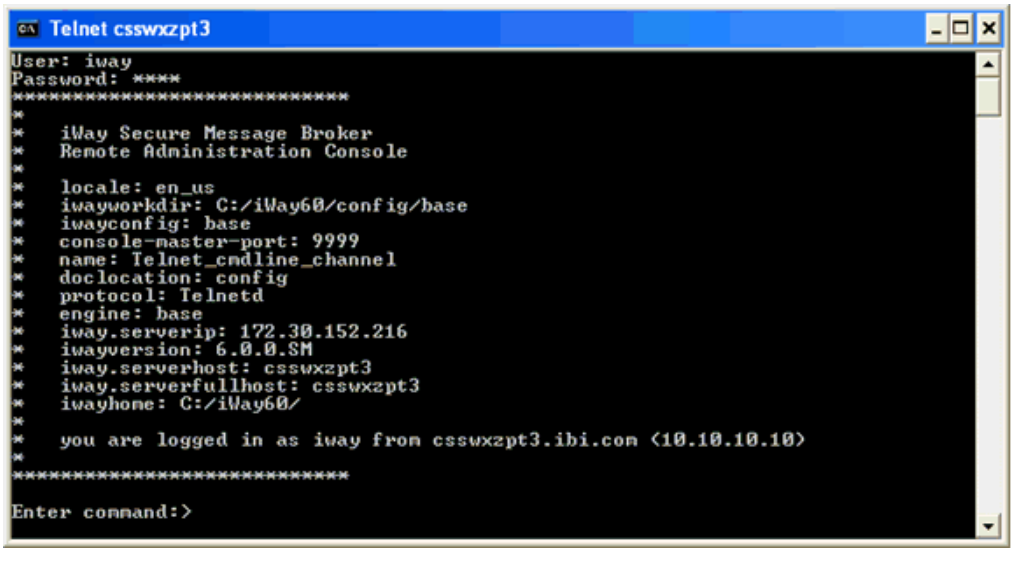
## Connecting to a Remote Command Console

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).

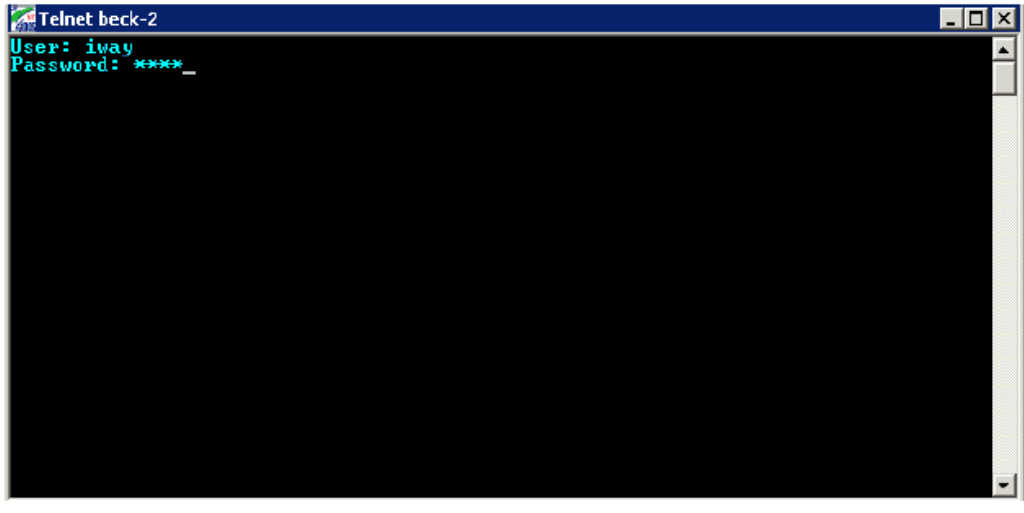



3. Once you are connected and logged in, you can now issue any command to monitor or control your iSM instance.

## 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.

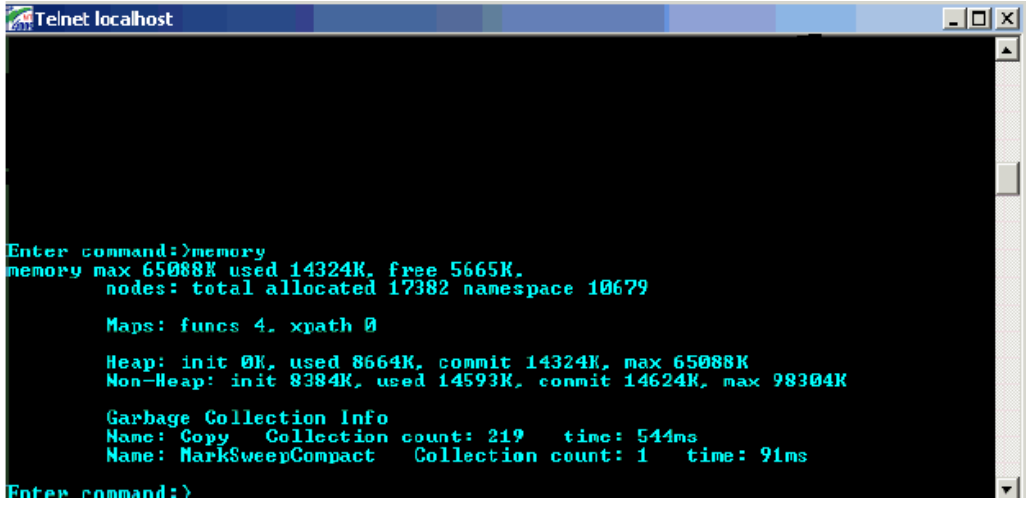
```
Telnet beck-2
User: iway
Password: ****
****************************
*
*    iWay Secure Message Broker
*    Remote Administration Console
*
*    iwayhome: c:/sm4bt/
*    protocol: Telnet
*    engine: base
*    iwayconfig: base
*    doclocation: config
*    console-master-port: 9999
*    locale: en_us
*    iwayversion: p3
*    iwayworkdir: c:/sm4bt/config/base
*    name: telnet
*
*    you are logged in as iway from beck-2.ibi.com (172.19.20.239)
*
****************************

Enter command:>_
```

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

```
Telnet beck-2
commands:
        SET             set a parameter [help := list parms that can be set]
        START           start the current instance or listener
        STOP            halt the current instance or listener
        INFO            run statistics on the current instance or listener
        QUIT            exit
        ERRORS          list last 10 errors
        MEMORY          list used and free memory [detail := analysis]
        GC              runs the Java garbage collector
        LINE            draws a line on the console
        TIME            prints the GMT time on the console
        THREADS         Lists outstanding threads [monitor on|off := track deadlock
s] [dump := dmp all threads]
        ROTATELOG       Closes the current log and causes a log rotation
        POOLS           Lists resource pools
        ROUTE           Display configured message routes
        SREGS           Display special registers.
        MANIFEST        Display the manifest of a named jar files
        PROVIDERS       Display providers currently in use
        EXITS           display loaded exits such as activity log and correlation m
anager
        RUN             run a command file
        SHOWLOG         dislpay the trace log
        HIDELOG         hide the trace log
Enter command:>_
```

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 * * *
*
```

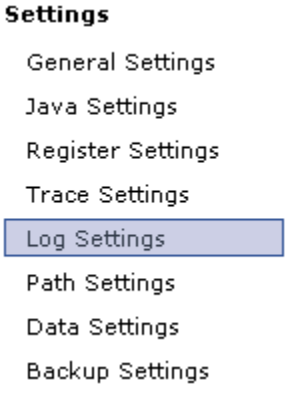
## Tracing and Log Levels

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

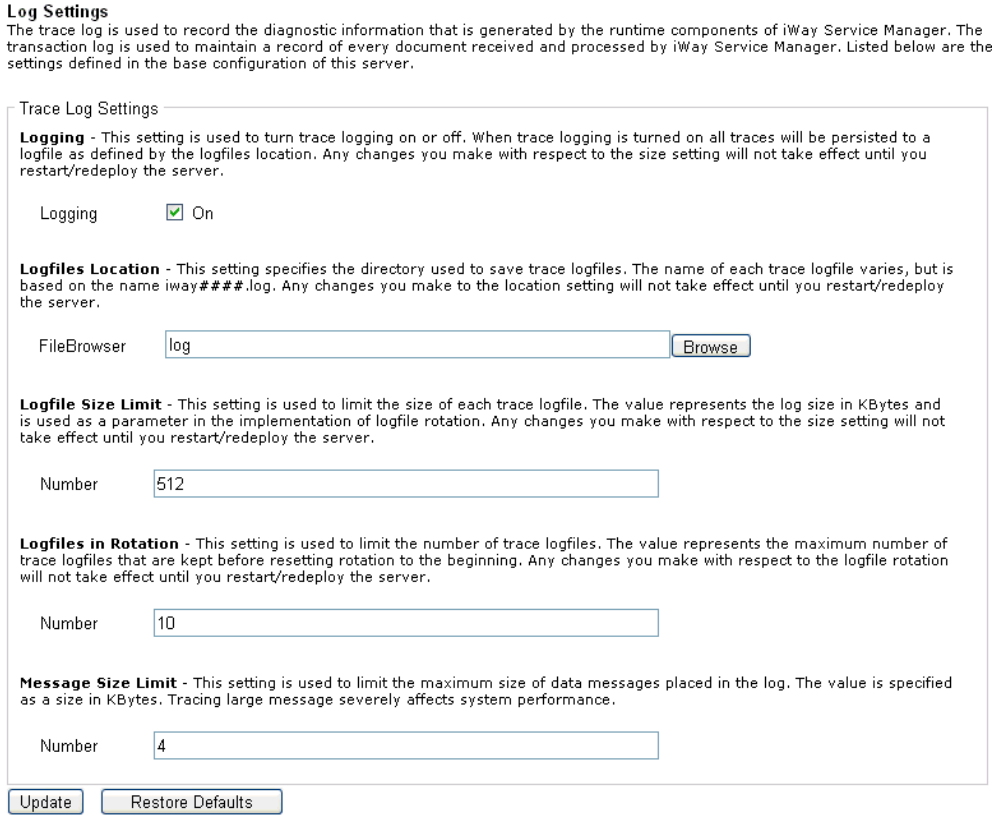
The Trace Log is used to record the diagnostic information that is generated by the run-time components of iWay Service Manager. The Transaction Log is used to maintain a record of every document received and processed by iWay Service Manager. The following procedure describes how to configure log settings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

### *Procedure:* How to Configure Log Settings




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

The Log Settings pane displays, as shown in the following image.




2. Change the default values.

   For more information, see *Log Setting Properties* on page 53.

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 |
|---|---|
| **Trace Log Settings** | |

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

## Trace Settings

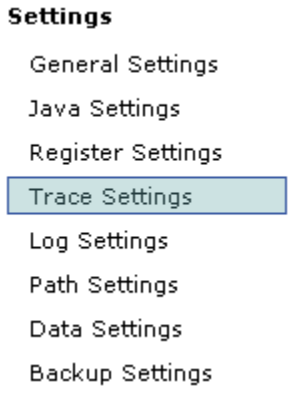
Tracing is key to diagnosing problems and thus to application reliability. iWay Service Manager provides a full complement of tracing services, oriented to diagnostic analysis of the running system. Tracing provides a step-by-step explanation of the internal activity of the server.

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 58.

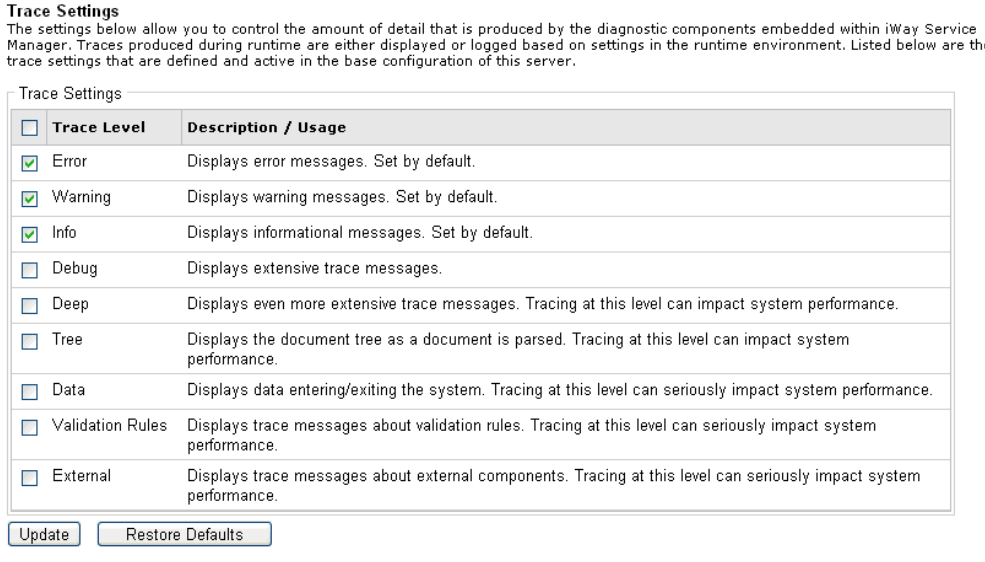
The following procedure describes how to control the amount of detail that is produced by the diagnostic components embedded within iSM. Traces produced during run time are displayed or logged based on settings in the run-time environment.

## *Procedure:* How to Select Trace Levels




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**

The settings below 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.

Trace Settings

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

Update  Restore Defaults

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 56.

3. Click *Update*.

## *Reference:* Trace Setting Properties

The following table lists and describes the trace setting properties.

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

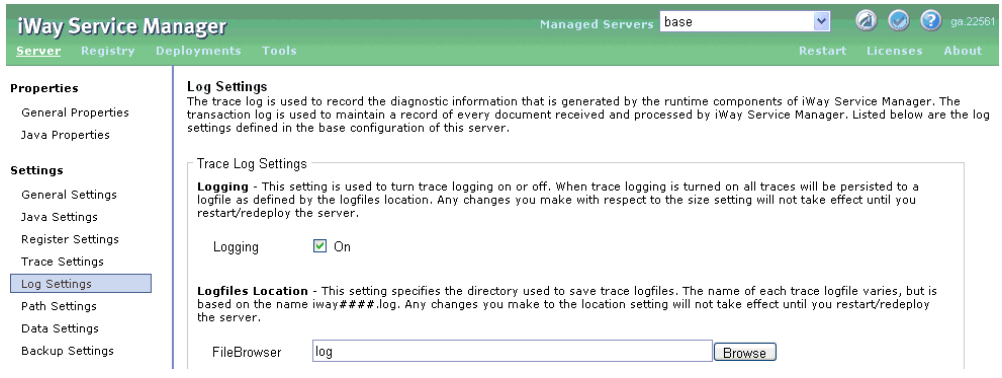
## *Procedure:* How to Log Traces to a File

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.

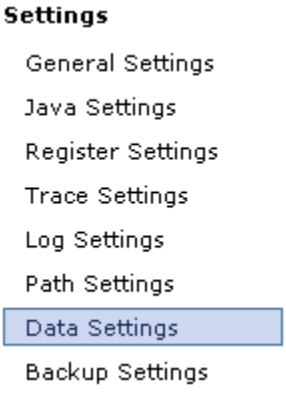
Traces are available in several levels and controlled independently. For more information, see *Trace Setting Properties* on page 56.

**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.

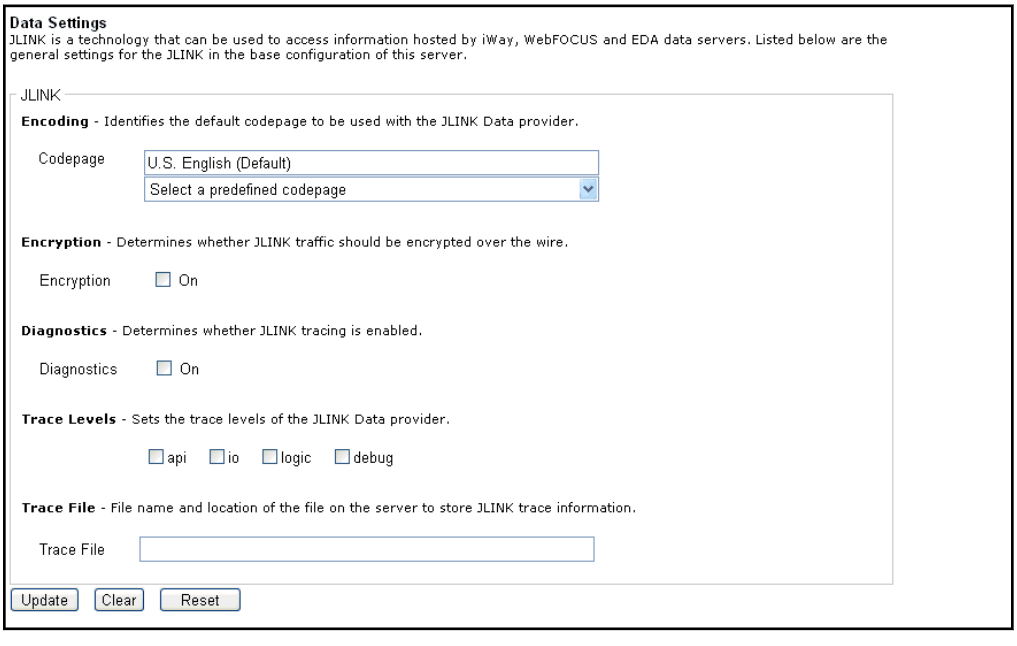
Unlike most design time settings, changing trace levels takes immediate effect in the run-time system. Changing the log file location does not take effect until iWay Service Manager is restarted.

## *Procedure:* How to Activate JLINK Tracing




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

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




   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*.

## Using the Log Viewer

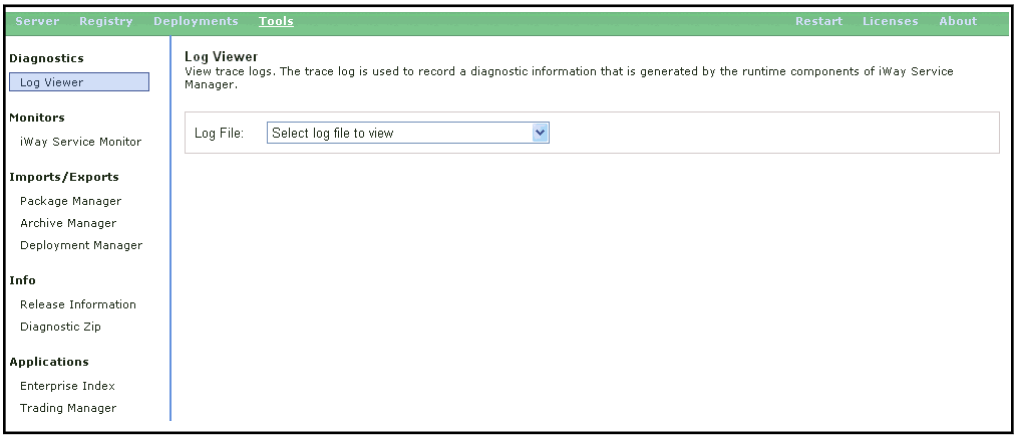
The Log Viewer manages the display properties of system debugging information when the logging and tracing functions are activated. It filters and displays debugging information as each transaction is received and processed. The Log Viewer also displays the date/time range, type, source, and message 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 52 and *Trace Settings* on page 54.

### *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.

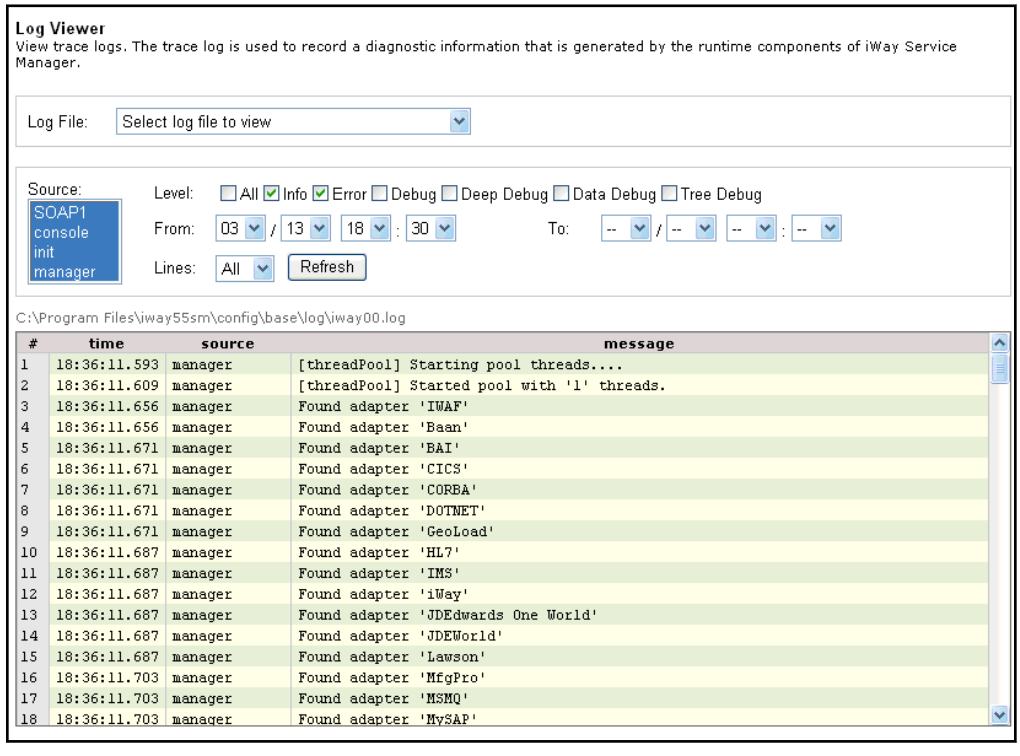
The Log Viewer pane opens, as shown in the following image.




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.

## Data Providers

A provider is a centrally configured resource that supplies services to run time components in iWay Service Manager (iSM).

Data Providers enable you to define and configure data servers and connections. The Data Provider properties include:

- JDBC connections
- JLINK to access iWay, WebFOCUS, and EDA data servers

## *Procedure:* How to Add a JDBC Connection

To add a JDBC connection:

1. In the left console pane of the Server menu, select *Data Provider*.
2. Beneath the JDBC section, click *Add*.

   The JDBC Data Provider Definition pane displays.
3. Provide the appropriate values for your JDBC connection as listed and defined in the following table.

| Parameter | Description |
|---|---|
| **JDBC Connection Pool Properties** | |
| Name * | Enter the name of the JDBC data provider to add. |
| Driver Class | The JDBC driver class is the name of the class that contains the code for this JDBC Driver. You can select a predefined database from the drop-down list or enter your own.<br><br>The following are sample values for the driver:<br><br>`com.microsoft.jdbc.sqlserver.SQLServerDriver`<br>`COM.ibm.db2.jdbc.app.DB2Driver`<br>`com.informix.jdbc.IfxDriver`<br>`sun.jdbc.odbc.JdbcOdbcDriver`<br>`oracle.jdbc.driver.OracleDriver`<br>`com.sybase.jdbc2.jdbc.SybDriver`<br><br>The required .jar files for the JDBC driver must be installed and registered in the Service Manager classpath. You can use the Path Settings option on the console to add Java classes and libraries. |

| Parameter | Description |
|---|---|
| Connection URL | The JDBC connection URL to use when creating a connection to the target database. The URL generally includes the server name or IP address, the port or service, the data source name, and a driver specific prefix. You can select a predefined database from the drop-down list or enter your own.<br><br>The following are sample values for the URL:<br><br>`jdbc:microsoft:sqlserver:`*`//server`*`:1433;DatabaseName=DB`<br>`jdbc:db2:`*`database`*<br>`jdbc:informix-sqli:`*`//HOST:PORT/DB`*`:INFORMIXSERVER=SERVER_NAME`<br>`jdbc:odbc:`*`DB`*`jdbc:oracle:`*`thin:@HOST:PORT:SID`*`jdbc:sybase:Tds:`*`HOST:PORT`*<br><br>For more information, see the JDBC documentation for the specific data source. |
| User | User name with respect to the JDBC URL and driver. SREG names can be used. |
| Password | Password with respect to the JDBC URL and driver. SREG names can be used. |
| **Connection Pool Properties** | |
| Initial Pool Size * | Number of connections to place in the pool at startup. |
| Maximum Number of Idle Connections * | Maximum number of idle connections to retain in the pool. A value of zero (0) means no limit except what is enforced by the maximum number of connections in the pool.<br><br>This value can be reset using the *jdbc* command, allowing this to be changed, often on a schedule, to respond to changing database conditions. |
| Maximum Number of Connections * | Maximum number of connections in the pool. A value of zero (0) means no limit.<br><br>This value can be reset using the *jdbc* command, allowing this to be changed, often on a schedule, to respond to changing database conditions. |

| Parameter | Description |
|---|---|
| Login Timeout | Time in seconds to wait for a pooled connection before throwing an exception. A value of zero (0) means to wait forever. |
| Behavior When Exhausted | What to do when the pool reaches the maximum number of connections. Block means wait for a connection to become available for the period defined by the login timeout parameter. Fail means throw an exception immediately. |
| Validation SQL | SQL statement that can be executed to validate the health of a pooled connection. The statement should return a result set of at least one row. |
| Validate on Borrow | If set to *true*, the validation SQL statement will be executed on a pooled connection before returning the connection to the caller. |
| Validate on Return | If set to *true*, the validation SQL statement will be executed on a pooled connection before replacing the connection in the pool. |

4. Click *Test* to check for proper connections.

   If a connection cannot be made, an error message displays describing the problem. Typically, the driver has not been installed or the classpath has not been set.

5. Click *Add* to return to the Data Provider pane.

## *Procedure:* How to Add a JLINK Data Source

Since drivers are stopped, they do not need to be added. However, if you need to add a JLINK data source:

1. In the left console pane of the Server menu, select *Data Provider*.
2. Beneath the JLINK section, click *Add*.

The Data Provider - JLINK pane displays, as shown in the following image.

**Data Provider - JLINK**
Listed below is the definition of the selected JLINK data provider. Add/Update the values as required.

**Identification**

| | |
|---|---|
| Name | Enter the name of the JLINK data provider to add. This parameter identifies the dataserver name (dsn) of the target JLINK accessible service. |
| Description | The description is used to help identify the target JLINK accessible service. JLINK Data Provider |
| Type | The server type is used to help identify the target JLINK accessible service. WebFOCUS Pro Server; Select a predefined JLINK server type. |

**Connections**

| | |
|---|---|
| Host | The host parameter identifies the host name or the ip address of the target JLINK accessible service. |
| Port | The port parameter identifies the TCP/IP service name or the port number of the target JLINK accessible service. |
| User | User name with respect to the JLINK accessible service. |
| Password | Password with respect to the JLINK accessible service. |

**Data**

| | |
|---|---|
| Engine | Identifies the default database engine style used by JLINK. 0; Select a predefined EDA Engine setting |
| Encoding | Identifies the code page to be used with the JLINK Data provider. 137; Select a predefined codepage |
| Encryption | Determines whether JLINK traffic should be encrypted over the wire. ☐ On |

**Diagnostics**

| | |
|---|---|
| Trace File | File name and location of the file on the server to store JLINK trace information. |
| Trace Settings | Sets the trace levels of the JLINK Data provider. ☐ api ☐ io ☐ logic ☐ debug |

Add Test

3. In the Name field, type the name of a new server. In this example, type *NEWSERV*.
4. In the Description field, type a brief description for the new server. The default is JLINK Data Provider.

5. From the Type drop-down list, select a JLINK server type.

   The default is WebFOCUS Pro Server.

6. Type the required values for Host and Port.
7. Type the required values for User and Password.
8. From the Engine drop-down list, select a database engine.

   The default is 0 (EDA).

9. From the Encoding drop-down list, select a codepage.

   The default value is 137 (U.S. English).

10. To encrypt data that is transported over the wire (optional), select the *Encryption* check box.
11. In the Trace File field, type the path and name of the file for the trace output.
12. To set trace levels, select any number of the check boxes listed (optional).
13. Click *Add*.

## Configuring Idle Connection Eviction

This feature allows you to specify the amount of time a connection can remain idle in the connection pool. Similar to a feature of Apache DBCP, the idle connection eviction thread runs at regular intervals, with the following options:

- **Eviction Interval.** Time between runs of the idle connection eviction thread, in seconds. A negative value means the eviction thread will never run.
- **Idle Connection Timeout.** The minimum amount of time a connection can remain idle in the pool before the eviction thread disposes of it. When the eviction thread runs and finds that a connection has been idle for at least this interval, the connection will be removed from the pool and closed. Note that a connection can remain in the pool longer than the timeout, depending on the scheduling of the eviction interval.
- **Maximum Number of Tests Per Run.** This is a performance tuning option. Since the eviction thread must lock the pool when it runs, this option allows you to specify how many connections can be tested in each run, thus limiting the duration of the lock. If the eviction interval is short, or the number of connections in the pool is large, performance may improve with fewer connections tested per run. Enter 0 to test all connections.
- **Validate Idle Connections.** If true, the eviction thread executes the validation SQL statement on connections that have not reached the idle connection timeout. If the statement does not execute successfully, the connection will be dropped from the pool.

Chapter 3

# Sizing and Capacity Best Practices

This chapter outlines and describes key sizing and capacity best practices for iWay Service Manager (iSM).

**In this chapter:**



## Managing and Monitoring Performance

Managing and monitoring performance using iWay Service Manager (iSM) allows you to:

- Record the inbound message or record the set size.
- Record outbound message or record the set size.
- Ensure data transforms and any other application changes to data execute.

## Understanding Expected Transactions Per Second (TPS)

Understanding expected transactions per second (TPS) through iWay Service Manager (iSM) allows you to:

- Monitor customers service license agreement (SLA) with partners.
- Monitor other business processes waiting on completion of an iSM/DQS application.

## Available Performance Monitoring Tools and Techniques

This section describes the available performance monitoring tools and techniques.

### Java Monitoring and Management Console (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

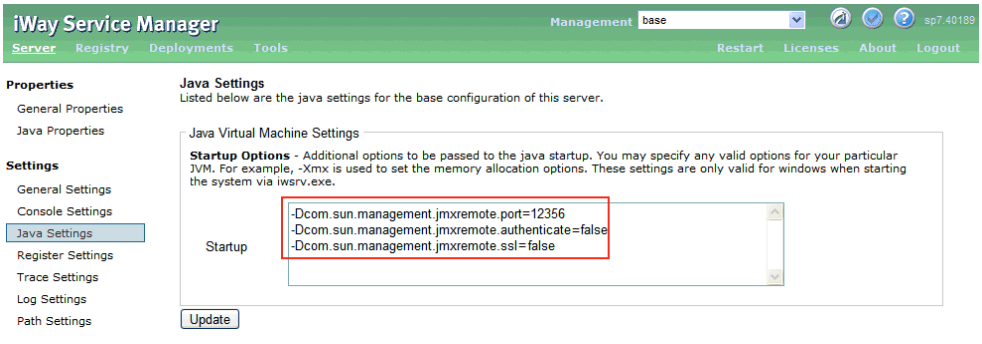
For more information on configuring and using JConsole, see:

*http://docs.oracle.com/javase/6/docs/technotes/guides/management/jconsole.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.

## Java Flight Recorder and Java Mission Control

Java Flight Recorder and Java Mission Control together create a complete tool chain to continuously collect low level and detailed runtime information enabling after-the-fact incident analysis.

Java Flight Recorder is a profiling and event collection framework built into the Oracle JDK. It allows Java administrators and developers to gather detailed low level information about how the Java Virtual Machine (JVM) and the Java application are behaving. Java Mission Control is an advanced set of tools that enables efficient and detailed analysis of the extensive of data collected by Java Flight Recorder. The tool chain enables developers and administrators to collect and analyze data from Java applications running locally or deployed in production environments.

## Available Resources in iWay Service Manager (iSM)

iWay Service Manager (iSM) provides resources that allos you to analyze the behavior of iSM.

Available statistics resources provide 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 available measurement resources can add significant overhead to the operation of iWay Service Manager. Therefore, do not use the measurement resources 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 resources are used, 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.

| Field | Description |
|---|---|
| *init* | 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.<br><br>The value of *init* may be undefined (0) on some platforms. |
| *committed* | 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).<br><br>The Java Virtual Machine may release memory to the system, and the value of *committed* could be less than the value of *init*. The value of *committed* is always greater than or equal to the value of *used*. |
| *max* | 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.<br><br>If *max* is defined, the amount of *used* and *committed* memory is always less than or equal to the value of*max*. A memory allocation may fail if it attempts to increase the *used* memory, such that the value of *used* is greater than the value of *committed*, even if the value of *used* is less than or equal to the value of *max* (for example, when the system is low on virtual memory). |
| *used* | Represents the amount of memory currently used. |

The heap display is more accurate than the memory display issued. 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

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

When iWay Service Manager is running without using the measurement resources, some statistics are generated with wall clock times. When using the measurement resources, 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
   name       count      low     high     mean variance  std.dev. ehr num/sec
mq1a       wall:  2  0.0470  0.1560  0.1015   0.0030    0.0545   -    9.85
           cpu :  2  0.0312  0.0625  0.0469   0.0002    0.0156   -
           user:  2  0.0312  0.0625  0.0469   0.0002    0.0156   -
```

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.

| Field | Description |
|---|---|
| count | The number of messages that have been handled, for which statistics have been gathered. |
| low | The lowest time recorded for the handling of a message. |
| high | The highest time recorded for the handling of a message. |
| mean | 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. |
| variance | The statistical variance of the times recorded. Variance is a measure of how numbers disburse around the mean. |
| std.dev. | The statistical standard deviation of the times recorded. Standard deviation is a measure of how numbers disburse around the mean. |

| Field | Description |
|---|---|
| ehr | 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 *const* is displayed.<br><br>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. |
| num/sec | 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. |

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

| Listener Name | Listener Type | Completed | Time clock | Mean (sec) | Std. Dev. (sec) | Variance (sec) | Ehrlang | Num/Sec |
|---|---|---|---|---|---|---|---|---|
| filein | FILE | 1 | Wall | 0.1250 | 0.0000 | 0.0000 | const | 8.00 |
| | | | CPU | 0.1094 | 0.0000 | 0.0000 | const | 9.14 |
| | | | User | 0.0938 | 0.0000 | 0.0000 | const | 10.67 |
| filelock | FILE | 0 | Wall | 0.0000 | 0.0000 | 0.0000 | | |
| | | | CPU | 0.0000 | 0.0000 | 0.0000 | | |
| | | | User | 0.0000 | 0.0000 | 0.0000 | | |

## 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.

| Field | Format | Description |
|---|---|---|
| type | String | The value 1, which is the record type. Other record types may be added in a future release. |

| Field | Format | Description |
|---|---|---|
| id | String | The generator (worker) ID. |
| tid | String | The transaction ID. |
| msglen | Integer | The message length (non-streaming). If the length cannot be determined, the value -1 is specified. |
| complexity | Integer | 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.<br><br>For most purposes, the number of digits that this integer has is a good value to analyze. |
| timestamp | Integer | 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. |
| gregtime | String | The timestamp in GMT (Greenwich Mean Time). The format is:<br><br>yyyy-mm-dd**T**hh:mm:ss:mmm**Z** |
| walltime | Float | The wall clock time expended, in milliseconds. |
| usertime | Float | The user time expended, in milliseconds. |
| cputime | Float | The CPU time expended, in milliseconds. |
| usedmem | Integer | The used memory, in kilobytes (K). See BAD XREF HERE "Memory. |
| committed | Integer | The committed memory, in kilobytes (K). See BAD XREF HERE "Memory. |
| $ | String | The end-of-record indicator. |

The following is a sample record in the file:

```
1, W.udp1.1, udp1-UDP-W.udp1.1_20050328191546135Z, -1,1735, 1112037346213,
2005-03-28-T19:15:46:135Z, 78.0, 15.625, 15.625,1200,800 $
```

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).

## Understanding Benchmarks (Throughput)

The CPU, memory, and disk usage must be recorded while benchmarks are recorded to understand the limitations of the environment in all three resource groups.

## Sample TPS Reports

The following table provides a summary of the benchmarking results that were performed against a customer application for two vendors.

| Vendor Information | Number of Records | Execution Time (In Seconds) | Records Per Second | CPU Usage |
|---|---|---|---|---|
| **Vendor: ABC** | | | | |
| Standalone - 2 Cores | 359,000 | 40 | 8975.00 | 80% |
| Standalone - 4 Cores | 359,000 | 34 | 10558.82 | 64% |
| Including iWay DataMigrator - 2 Cores | | Not Captured | | |
| Including iWay DataMigrator - 4 Cores | 359,000 | 36 | 9972.22 | 65% |
| **Vendor: XYZ** | | | | |
| Standalone - 2 Cores | 2,260,000 | 136 | 16617.65 | 96% |
| Standalone - 4 Cores | 2,260,000 | 76 | 29736.84 | 89% |
| Including iWay DataMigrator - 2 Cores | 2,260,000 | 206 | 10970.87 | 100% |
| Including iWay DataMigrator - 4 Cores | 2,260,000 | 71 | 31830.99 | 86% |

## Customer Example #1

### Test Case 1

The following table provides a summary of the benchmarking results for test case 1, where peak performance was reached during 100,000 transactions within a 20-minute timeframe.

**Message Type Used for Testing:** EDI X12 322

**Message Size:** 1KB

| | Server 1 | | Server 2 | | |
|---|---|---|---|---|---|
| | **iSM1** | **iSM2** | **iSM3** | **iSM4** | |
| **Transactions** | 25,000 | 25,000 | 25,000 | 25,000 | **100,000**<br>(Total Transactions) |
| **Average Time (mm:ss)** | 5:36 | 5:37 | 5:39 | 5:33 | **5:36**<br>(Average Time) |
| **TPS** | 74.52 | 74.07 | 73.75 | 75.19 | **297.53**<br>(Total TPS) |

**Test Case 2**

The following table provides a summary of the benchmarking results for test case 1, where normal performance was reached during 515,000 transactions within a 5-hour timeframe with a minimum of 33.33 transactions per second (TPS).

**Message Type Used for Testing:** EDI X12 322

**Message Size:** 1KB

| | Server 1 | | Server 2 | | |
|---|---|---|---|---|---|
| | **iSM1** | **iSM2** | **iSM3** | **iSM4** | |
| **Transactions** | 128,750 | 128,750 | 128,750 | 128,750 | **515,000**<br>(Total Transactions) |
| **Average Time (mm:ss)** | 27:40 | 27:50 | 28:01 | 28:00 | **27:53**<br>(Average Time) |
| **TPS** | 77.54 | 77.12 | 76.61 | 76.64 | **307.91**<br>(Total TPS) |

## Customer Example #2

The following table provides a summary of the benchmarking results for three test cases.

**Test Case 1** (500 files each containing 1 consignment)

| | IN_CON_B2BIT | | SPLIT_CON_B2BIT | | ROUTE_CON_IT | |
|---|---|---|---|---|---|---|
| | **1 Thread** | **10 Threads** | **1 Thread** | **10 Threads** | **1 Thread** | **10 Threads** |
| **Seconds** | 23 | 22 | 98 | 23 | 166 | 31 |
| **Throughput** | 21.74 | 22.73 | 5.10 | 21.74 | 3.01 | 16.13 |

**Test Case 2** (5 files each containing 100 consignments)

| | IN_CON_B2BIT | | SPLIT_CON_B2BIT | | ROUTE_CON_IT | |
|---|---|---|---|---|---|---|
| | **1 Thread** | **10 Threads** | **1 Thread** | **10 Threads** | **1 Thread** | **10 Threads** |
| **Seconds** | 6 | 5 | 74 | 21 | 171 | 25 |
| **Throughput** | 83.33 | 100.00 | 6.76 | 23.81 | 2.92 | 20.00 |

**Test Case 3** (1 file each containing 500 consignments)

| | IN_CON_B2BIT | | SPLIT_CON_B2BIT | | ROUTE_CON_IT | |
|---|---|---|---|---|---|---|
| | **1 Thread** | **10 Threads** | **1 Thread** | **10 Threads** | **1 Thread** | **10 Threads** |
| **Seconds** | 5 | 5 | 77 | 76 | 163 | 35 |
| **Throughput** | 100.00 | 100.00 | 6.49 | 6.58 | 3.07 | 14.29 |

## Load Tools

This section describes various load tools that can be used to achieve optimal sizing and capacity configurations.

### 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/*

### QAStresser Tool

The QAStresser is a tool that is packaged with iWay Service Manager (iSM). This tool allows you to select a test input file and copy it as many times as required into a file listener input directory.

The current syntax requires the QAStresser tool to be executed from the iSM \lib directory. For example:

```
C:\iway7\lib> java -cp iwcore.jar com.ibi.tools.QAStresser C:\file\Test.xml
C:\file\in 5
```

In this example, the input file *Test.xml* will be copied five times into the *C:\file\in* directory.

**Note:** In iSM Release 7.0.5, the QAStresser tool can be called from the iSM command line prompt as follows:

```
Enter command:>tool QAStresser
```

Chapter 4

# Analysis and Tuning Best Practices

This chapter outlines and describes key analysis and tuning best practices for iWay Service Manager (iSM).

**In this chapter:**



## Analyzing Your Java Virtual Machine

This section describes the available tools and facilities that you can use to analyze your Java Virtual Machine (JVM).

### Analyzing Your CPU Usage

CPU analysis is critical and assures that you are getting the most out of the hardware you have selected for your application. There are several tools built into the JVM, which can be used to monitor CPU usage (for example, Jconsole, JMC Java Mission Control, and JMX).

## Tuning

To get the most out of the Java Virtual Machine (JVM), you must tune the following areas:

- Heap (Memory, JVM, and OS)
- CPU
- Garbage Collector (for IBM JVM platforms, this would be Policies)
- Hardware (disk arrays, network connections, check physical memory, and so on)
- JVM Runtime Options (for example, stack size)

### Types of Tuning Based on Configuration and Container

All iWay Service Manager (iSM) services (JVM) may not require the same resources. Heap (Memory) will most likely have to be adjusted for each iSM service (JVM). This means that if iSM service A is consuming large flat files (EDI) that need to be converted to XML (the native document format for iSM) the iSM service may require a much larger Heap size (for example, greater than 2048 MB) to handle the conversion from flat to XML. However another iSM service that simply Moves a file from one FTP site to another may require a minimal Heap size (for example, less than 256 MB).

### Tuning at the Channel Level

Settings within a channel can increase overhead thereby decreasing performance. These settings are recording activity, generating an XML tree (Accept Flat = false), persistence (for queue type listeners), and writing to external logs. This list keeps increasing as our need to monitor increases.

### Tuning at the Thread Level

Increasing the thread count for an application may not be the only factor you need to observe when tuning at the thread level. Available TCP ports may also govern the speed or rate at which threads run and are recreated. Variables (Special Registers) created within an application must be isolated properly if an application runs using parallel threads.

## Recommendations

The recommendations described in this section are based on the results of the analysis and tuning that are performed for iWay Service Manager (iSM).

### Configuring the Heap (Memory) Size

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:

`-Xms`*`256M`*

where:

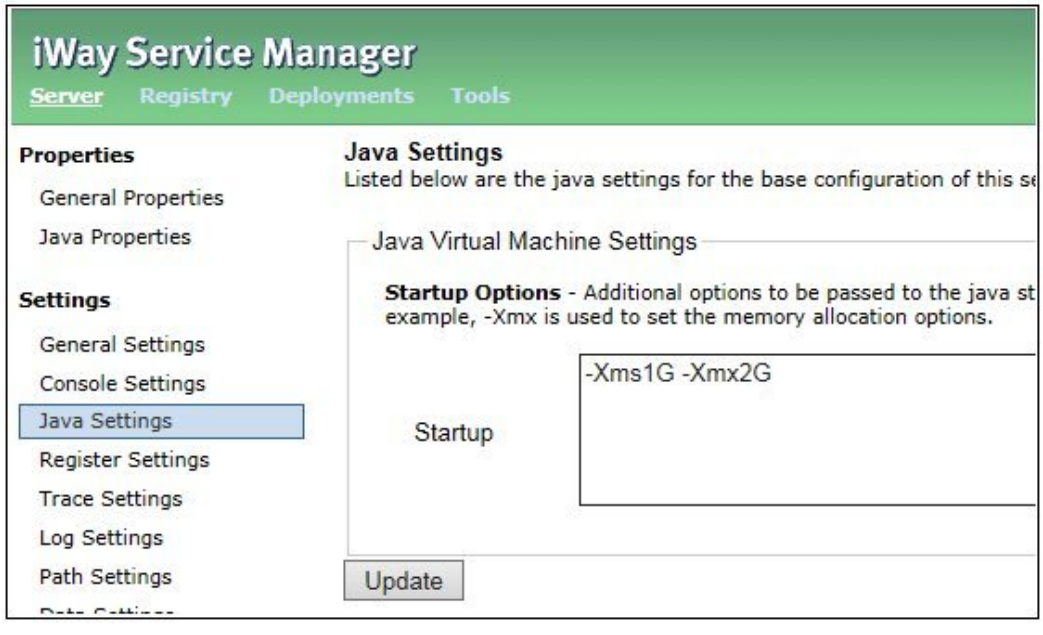
*256M*

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.



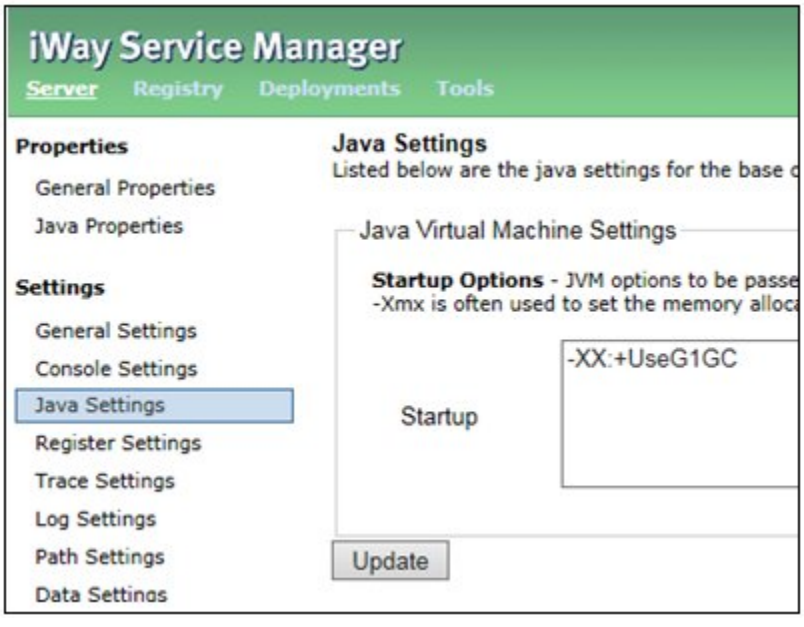

**Note:** The `-Xms1G -Xmx1G` configuration settings set the Heap size to one Gigabyte.

## Benefits of Multiple Cores

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 collector. Using the following Java option will also instantiate the G1 collector at JVM startup:

```
-XX:+UseG1GC
```




## 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:

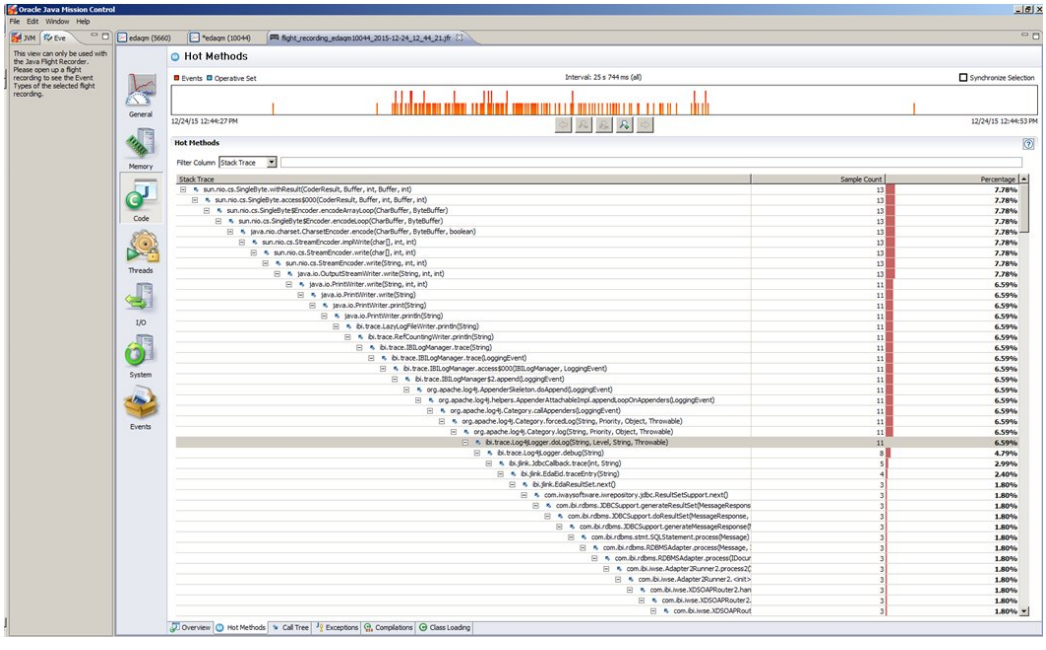
```
grep "^processor" /proc/cpuinfo | wc -l
```

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

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

## Determining Performance Using Java Mission Control (JMC)

In the scenario below, a transaction was invoked through the RDBMS Adapter, which calls JLINK to access the EDA Server. The JMC Console, can be configured to capture the methods that consumed the majority of the application runtime. In this case, it can be observed in the stack trace, the ibi.trace.Log4jLogger.doLog() method is taking about 6.59% of the runtime. It is evident the method indicates that traces are enabled, and that switching traces off would improve performance.




## JDBC Data Provider Best Practices and Troubleshooting Guidelines

This section describes best practices for the JDBC data provider and provides troubleshooting guidelines.

- JDBC data provider recommendations
- Tracing recommendations ranging from the JDBC data provider to the JDBC driver level.

## Advantages and Disadvantages

A JDBC data provider is a centralized location for database connectivity; that encapsulates the database connection details from the front end application. As a result the data provider is recommended for most use cases since it provides the following benefits.

- **Easy access to data.** The database can easily be accessed since only the provider name and the context factory name (com.ibi.jndi.XDInitialContextFactory) is specified on the JDBC client component (SQL Object, RDBMS Adapter, and so on) in order for a connection handle to be made available.
- **Central point of administration.** The data provider makes the database configuration changes transparent to the application. For example, if the iSM Administrator needs to switch to a different DBMS server, the data provider can be modified without any changes on the application side. Additionally, if resources are limited, the system administrator could choose to reduce the connection pool size after hours and increase the pool size during regular business hours while not making any application changes.
- **Connection pool considerations.** It is very costly for an application to constantly be opening and closing connections when transacting against a DBMS. As a result, it is imperative that the connection pool size is adjusted to the max number of concurrent active connections. This can be achieved by setting the *Maximum Number of Connections* field seen below to the max number of concurrent connections.

  If the use case allows for the application to spend the time (varies based on resources available) initializing the pool upon initial execution, the *Initial Pool Size* should also be set to equal to the value of *Maximum Number of Connections*.

  If the use case performance allows the time for additional connections to be established later in the application life cycle, the *Initial Pool Size* can be made smaller than the *Maximum Number of Connections* field. Keep in mind that in this case, the *Maximum Number of Idle Connections* sets the criteria for the data provider to close any idle connections exceeding the specified value.

  In the example below (see the data provider configuration), when the application is launched, upon initial connection to the data provider the pool is established with 10 idle connections (see the image below). The application will borrow 1 of 10 connections and return it to the pool when the transaction is complete. If the application were to generate 12 threads concurrently, there will be 10 borrowed connections, plus two newly created connections (a total of 12), and at the end of processing the data provider will return 10 connections to the pool and close the two connections since the idle limit is set at 10. If the two connections that were newly created are taking significant time, the data provider *Initial Pool Size* can be set to 12, so the time is spent initially, and save on resources later in the application life cycle.

On a side note, it is recommended to set the *Maximum Number Of Idle Connections* to a number that is greater or equal to the *Initial Pool Size*. The reason is the initial connections are in an idle state at first. Otherwise the data provider will not load successfully.

**Data Provider Configuration Example**

| | |
|---|---|
| Initial Pool Size * | Number of connections to place in the pool at startup. |
| Maximum Number of Idle Connections * | Maximum number of idle connections to retain in the pool. 0 means no limit except what is enforced by the maximum number of connections available from the pool. |
| Maximum Number of Connections * | Maximum number of connections available from the pool. 0 means no limit. Connections above the Idle Connections limit are not retained in the pool. |

- **Validate On Borrow.** This feature is an imperative feature of the data provider. The feature allows the provider to check the validity of a connection before passing back the connection to objects such as the SQL Object, or RDBMS Adapter. For example, in certain cases, an application may encounter a brief network outage. In such a case, certain JDBC drivers will not retry the connection and return a stale connection object resulting in thrown errors/exceptions. By using the Validate On Borrow feature, a short SQL statement (specified in the *Validation SQL* field) is executed to validate the connection. In contrast, if a data provider is not used, the application would have to manage the logic to validate and re-instate the connection object which leads to further logic and maintenance overhead at the application level.

There are two main reasons to consider before using a JDBC data provider:

1. Database login credentials and/or URL will not change. If login credentials vary, different instances of the data provider are created, resulting in different connection pool instances, which leads to an excessive use of resources which can cause performance issues. This would defeat the purpose of the connection pool benefits.
2. Single threaded application which only requires a single connection. Using a data provider will cause more resource overhead than the one non-pooled connection. Therefore, it is important to consider the performance trade off.

## Troubleshooting the Data Provider

When troubleshooting issues with the provider there are many levels at which an issue can occur. Therefore, it is important to be able to trace the different components in order to find the cause of the issue. iWay Service Manager provides many logging levels of which the most important are WARN, and DEEP when tracing the data provider. It is recommended to enable *Deep* and *Warn* level logging to capture the connection pool activity. Note that the *Deep* level messages will occur more frequently.

For example, when a connection is borrowed from the pool, the following will be logged:

```
DEEP (manager) in pooled connection factory makeConnection()
```

When a connection fails validation, the following message will be logged:

```
WARN (manager) connection fails validation test
```

When the connection validates successfully the following message is logged:

```
DEEP (manager) connection passes validation test
```

## Troubleshooting the JDBC Driver

If the data provider traces indicates the issue could occur at the JDBC Driver level, it is important to enable JDBC driver tracing. For the purpose of the article, I selected the DB2 JCC JDBC driver as an example. The benefit of driver level tracing is that the JDBC API level calls are logged. Therefore, it is easier to verify the state of the APIs and troubleshoot exceptions raised such connection and/or SQL memory resources.

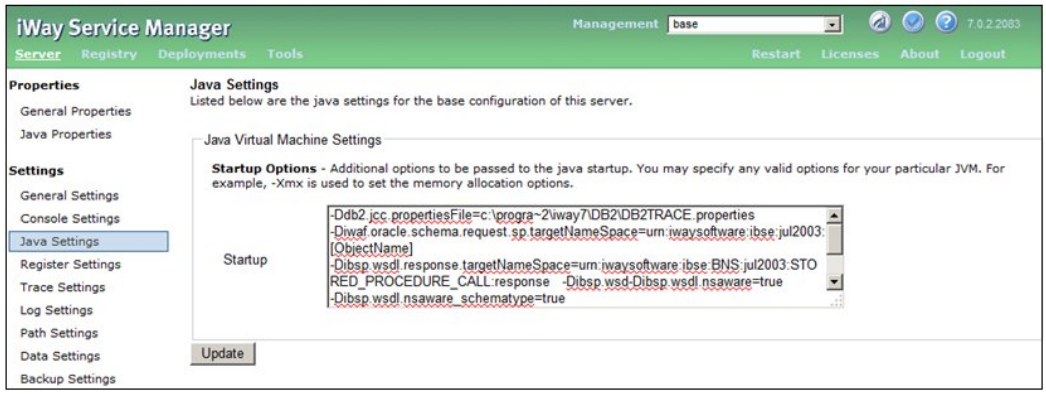
To enable DB2 JDBC driver traces a properties file needs to be created with the following content:

```
db2.jcc.traceDirectory=path to directory where traces are to be written
db2.jcc.traceFile=name of the trace file
db2.jcc.traceFileAppend=overwrite existing trace file (true / false)
db2.jcc.traceLevel=(level of tracing.  -1 = all, 0 = none)
```

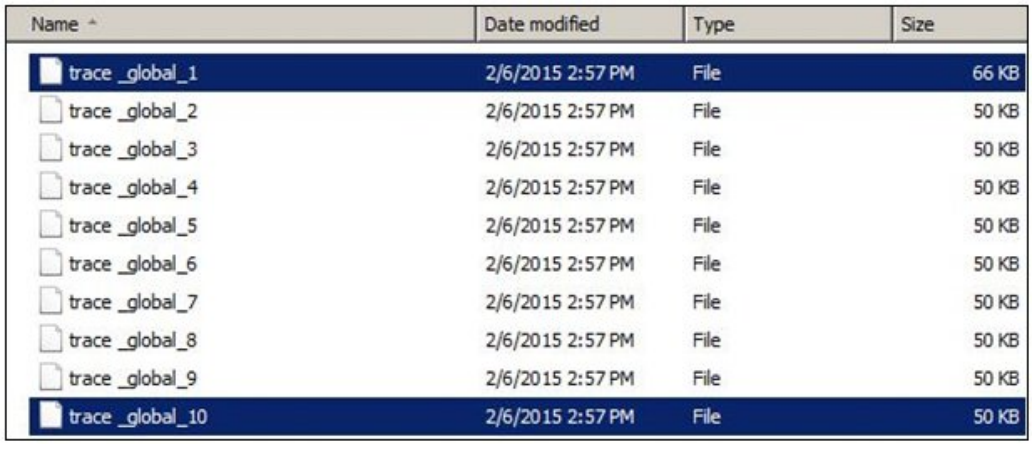
```
db2.jcc.traceDirectory=/Users/Administrator/DB2/jcctrace
db2.jcc.traceFile=trace
db2.jcc.traceFileAppend=true
db2.jcc.traceLevel=-1
```

In order for the JDBC application (in this case iSM), to locate the property file, the following JVM property must be set. The property can be set through the iSM Administration Console as seen in the *Java Settings* section.

```
-Ddb2.jcc.propertiesFile=[absolute path to properties file]
```




Once the property is specified, a cold restart of iSM is required. Once the iSM JDBC application is called, the traces will generate. The application used consists of an iSM Process Flow which calls a SQL Object. The SQL Object looks up the JDBC Provider through JNDI. Once the data provider returns the JNDI context, the SQL Object can connect to the DB2 database through the data provider. In this scenario, the data provider was configured to instantiate with an initial connection pool size of 10. As a result, 10 trace files are generated, as seen in the following image (one for each connection in the pool). The data provider manages the pool.



| Name | Date modified | Type | Size |
|---|---|---|---|
| trace _global_1 | 2/6/2015 2:57 PM | File | 66 KB |
| trace _global_2 | 2/6/2015 2:57 PM | File | 50 KB |
| trace _global_3 | 2/6/2015 2:57 PM | File | 50 KB |
| trace _global_4 | 2/6/2015 2:57 PM | File | 50 KB |
| trace _global_5 | 2/6/2015 2:57 PM | File | 50 KB |
| trace _global_6 | 2/6/2015 2:57 PM | File | 50 KB |
| trace _global_7 | 2/6/2015 2:57 PM | File | 50 KB |
| trace _global_8 | 2/6/2015 2:57 PM | File | 50 KB |
| trace _global_9 | 2/6/2015 2:57 PM | File | 50 KB |
| trace _global_10 | 2/6/2015 2:57 PM | File | 50 KB |

The iSM process flow executed a basic SQL statement (select * from SAL1). As a result, the trace contains generated connection, statement, and result set object references as seen in the excerpt below. Each reference has an identifier which would be different for each newly created instance. When reviewing the complete trace file, it can be confirmed that each object that is opened is also closed successfully to ensure there are no memory or resource leaks. Also, any exceptions or errors can be associated with a specific JDBC object. This allows confirmation on whether the issue is at the JDBC driver level, the data provider level, or at the iSM Enterprise Service Bus level.

The following is a trace excerpt showing the connection, statement, and result set objects.

```
[Connection@39d029f6] BEGIN TRACE_CONNECTS
[jcc][Connection@39d029f6] Successfully connected to server jdbc:db2://
aixppc53:60024/sample
[jcc][Connection@39d029f6] User: csssgh
[jcc][Connection@39d029f6] Database product name: DB2/AIX64
.
.
.
[jcc][Time:2015-02-06-15:21:35.397][Thread:W.SOAP1.1][Connection@39d029f6]
createStatement () called
[jcc][Time:2015-02-06-15:21:35.407][Thread:W.SOAP1.1][Connection@39d029f6]
createStatement () returned Statement@d0ce0e3
[jcc][Time:2015-02-06-15:21:35.407][Thread:W.SOAP1.1][Statement@d0ce0e3]
setMaxRows (0) called
[jcc][SystemMonitor:start]
[jcc][Time:2015-02-06-15:21:35.407][Thread:W.SOAP1.1][Statement@d0ce0e3]
execute (SELECT * FROM SAL1) called
.
[Time:2015-02-06-15:21:35.527][Thread:W.SOAP1.1][ResultSet@5f87d412] next
() called
.
[jcc][Time:2015-02-06-15:21:35.547][Thread:W.SOAP1.1][Connection@39d029f6]
close () called
```

# 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.*

Information Builders

# iWay

## Best Practices for iWay Service Manager (iSM) Installation, Deployment, and Analysis

*iWay Enterprise Enablement Program (EEP)*

DN3502110.0818

Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898