TIBCO iWay® Service Manager

Command Reference Guide

Version 8.0 and Higher
March 2021
DN3502197.0321

Contents

Chapter 1

Introducing iSM Commands

This section provides an introduction to the iWay Service Manager (iSM) command environment and describes the available facilities you can use to run iSM commands.

In this chapter:

iWay Service Manager Command Environment

iWay Service Manager (iSM) offers a command handler to manage local or remote instances of iSM. Commands can control an iSM instance, such as:

- Starting or stopping listeners.
- Setting server attributes.
- Displaying diagnostic information.
- Performing specific services, such as copying files and updating property files.

For a full description of the iSM command repertory, see *iWay Service Manager Commands Reference* on page 51.

Examples of commands might include:

```
start group1:inlet1:listener1
```

This command Instructs the listener to begin processing messages.

```
set debug on -save
```

This command changes the trace setting for iSM.

```
show stats
```

This command displays execution statistics for the current iSM instance.

```
Copy c:/docs/input1.xml /inlocation
```

This command copies a file to the specified input location of a listener.

iSM commands can be entered at:

- The startup window, if one is available.

  Note: Some operating systems do not offer such a window (for example, when starting iSM as a Windows service).

- In a startup script that runs automatically when iSM starts.
- Through the command scheduler, which executes commands at a specified time.
- At an auxiliary command window configured for iSM and started. There are two types of command windows:
  - **Telnet.** Supporting commands entered at a telnet client selected by the user.
  - **SSH.** Accepting commands through the Secure Shell (SSH) protocol.

Note: The commands available at any specific iSM instance will depend on the services that are supported by that iSM instance.

Using Command Help

In addition to this documentation, the iSM command handler offers an abbreviated online help through the *help* command. The following example shows a partial command list. A complete list is available in *iWay Service Manager Commands Reference* on page 51 or in documentation relating to specific iSM products, applications, or solutions.

```
Enter command:>help

Bam - Business Activity Monitor support command.

calendar - Manage the Service Manager Calendar Provider's data.

Copy - Copy a file from source to target.

cron - Manage the Service Manager Scheduler
(also see schedule command).

diagzip - Creates a diagnostic information file for use by iWay Support.

Enqueue - Enqueue a message to an internal or ordered queue.

errors - List last errors.

flow - Run a named and published process flow.

gc - Runs the Java garbage collector.
```

```
help - Display help for commands. Use "help <command>..." or "help ifl..."
for additional help.

info - Display channel information.

jdbc - Manage a JDBC provider connection pool.

line - Draw one or more lines on the console.

package - Manage packages.

publish - Publish or remove iIT process flows to the Service Manager.

pull - Load information from another configuration/installation.

quit - Exit the server.

refresh - Reinitialize a channel.

remote - Directs commands to a named configuration.

run - Run a command file.

say - Emits a line to the console and spool.

schedule - Manage the Service Manager Scheduler (also see cron command).

script - Execute Script through the Scripting engine.

set - Set a parameter.

shell - Attempt to run an operating system command.

show - Display server information.

spool - Record commands and responses in a spool file.

start - Start one or more channels.

stats - Run statistics on the current instance or listener.

stop - Stop one or more channels.

time - Print the time on the console.

tool - Run a named tool such as 'debugger'.

version - Display product version and all later versioned jars.
```

```
xalog - Start or stop an activity log driver.
```

Additional help is available for most commands, for example:

```
Enter command:>help start

Starts one, several or all channels.

start <name | namelist> [switches]

name = a single name of a channel/listener or a protocol. Names can end
with * for multiselection

namelist = a list of channels/listeners or protocols in the form
(one,two...) The special entry 'all' or '*' starts all listeners (non
protocol form)

switches

-protocol = the names are interpreted as protocols for example to start
inactive file channels start file -protocol -inactive

-pulse = attempt to start a pulsable protocol for a single access/poll
operation

-active = (default) only active channels are started. A channels is only
considered to be active when the channel has a green check in the column
labeled Active in the iSM Deployments console. Active channels start with
general startup such as when the server starts or via 'start all' requests.

-inactive = only inactive channels are started. A channels is considered to
be inactive when there is a red 'x' in the column labeled Active in the iSM
Deployments console. This allows channels to be reserved for special use.

-both = both active and inactive channels are started

-doflow = run the channel startup failure flow if the channel cannot start
[note, you may want this when start is called from a script]

-noflow = do not run the channels startup failure flow (default)

Enter command:>
```

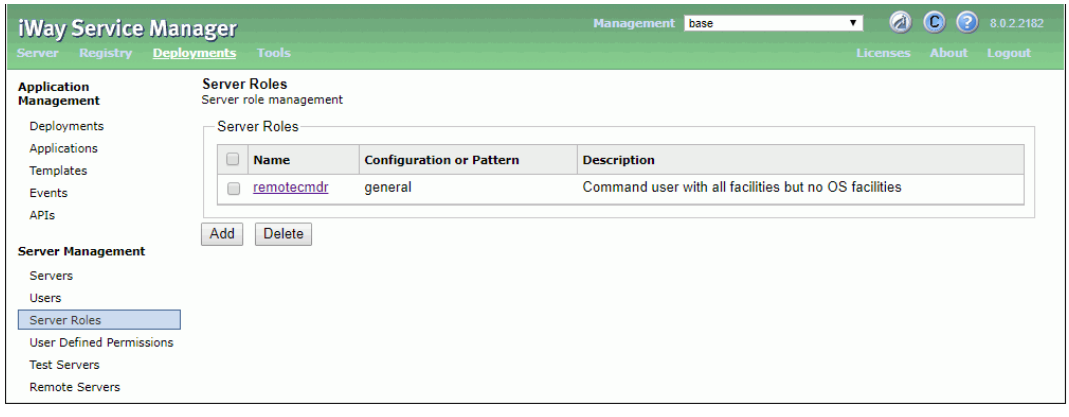
Managing Command Security

Security for the iWay Service Manager (iSM) command facility is controlled and managed by the Access Control List (ACL) of iSM. For more information on this topic, see the *iWay Service Manager Security Guide*.

To issue a command, a user must be logged into the command facility with the appropriate authority. The main iSM Administration Console always runs at the *administrator* authority level. It is the responsibility of the system administrator to provide physical security to this console.

Command windows (telnet and SSH) are secured by their authentication realm, while each command issued by the Scheduler carries its own security. Access to the Scheduler is controlled by the system in which the schedule is created.

To quickly summarize the ACL facility, the system administrator first creates an ACL of a given name. Each user being authenticated by a security realm can be granted one or more such ACLs. In this example, we will create a list called *remotecmdr*, which will support some (but not all) commands.



Once this list is created, the specific commands (and in some cases groups of commands) that are available in that list are configured. In our example, the user issuing the command can perform all operations with the exception of the *shell* command and the *set acl* command.

Command Permissions for Authentication Realms (Remote Command Channel, local command window or Scheduler)	
Can use remote command	The **remote** command enables a command user to manage a remote configuration/iIA. A Telnet Command Channel need not be configured on the remote target. ☑ On
Can use flow command	The **flow** command enables a command user to run a system process flow on demand. ☑ On
Can use run command	The **run** command enables a command user to run a command script. The commands in the script run under the user permission level. ☑ On
Can use set register command	The **set register** command enables a command user to change special register values ☑ On
Can use set property command	The **set property** command enables a command user to update a property file. ☑ On
Can use shell command	The **shell** command enables a command user to run an authorized and valid OS command ☐ On
Can use start command	The **start** command enables a command user to start one or more listeners. Startup dependencies apply. ☑ On
Can use stop command	The **stop** command enables a command user to stop one or more listeners ☑ On
Can use set acl command	The **set acl** command enables a command user to add or delete ACL authorizations for a user/group ☐ On

The user holding the *remotecmdr* list must be added to the realm (for example, LDAP) in order for the command user to log in.

Chapter 2

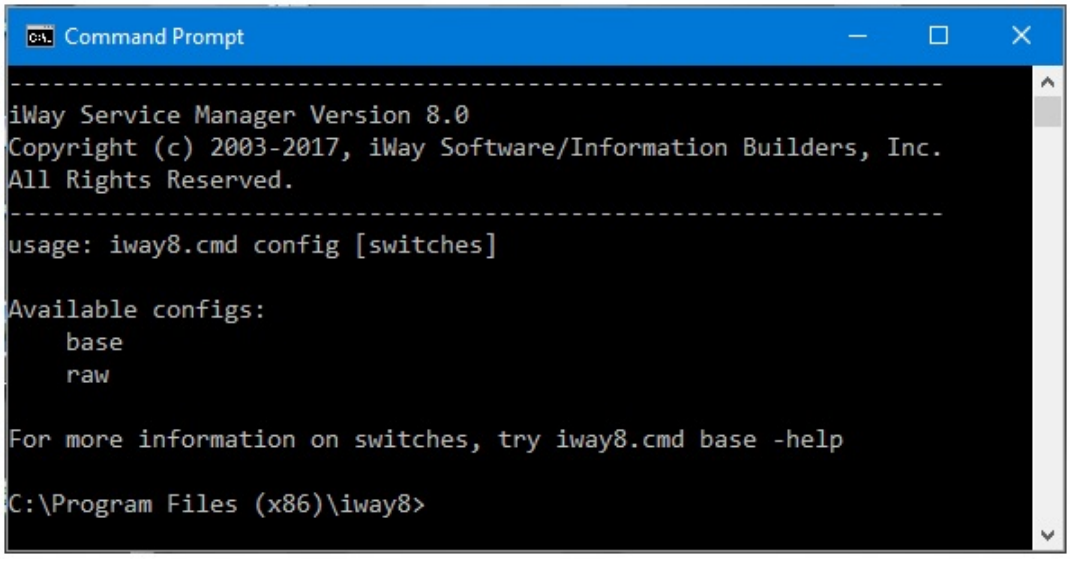
Creating and Using a Remote Command Console

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

In this chapter:

Remote Command Console Overview

iWay Service Manager (iSM) commands such as *start* or *flow* can be entered at the original command window if iSM (the server) is started as a task with a visible window (for example, starting from a command line such as iway8.cmd).



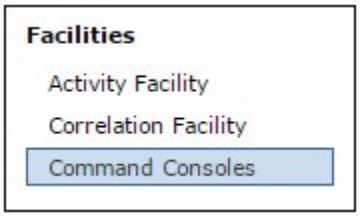
Additionally, commands can be entered using a remote command facility using Telnet (with or without Secure Sockets Layer (SSL)) or Secure Shell (SSH). In either case, the full set of iSM commands is available to the user, depending on the security level at which the logged in user has been granted.

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.

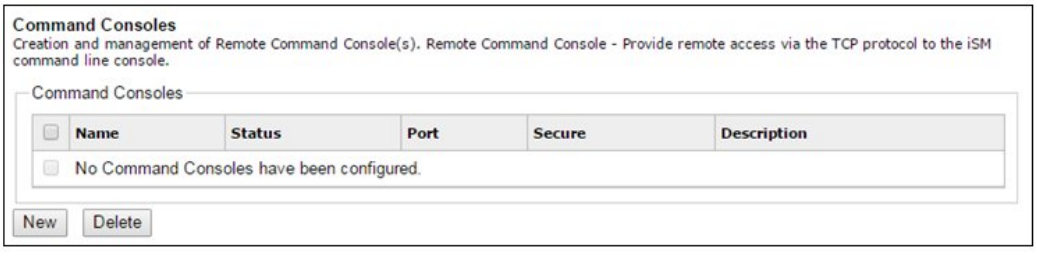
Usually the remote command channel runs off of the base configuration, and the remote command is used to address other running configurations either on the same or another host. A remote command console can be configured to any configuration that is currently running on a host.

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.

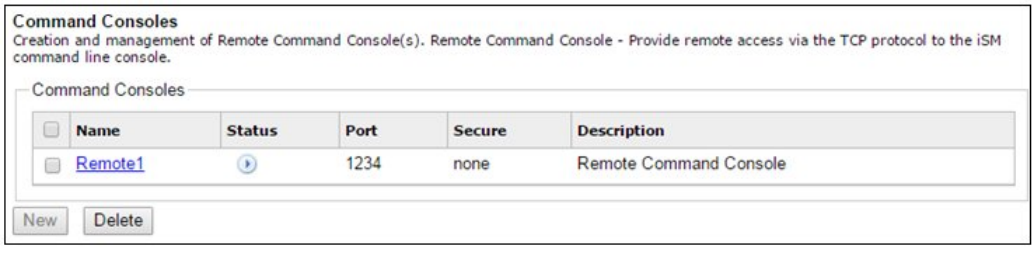


The Command Consoles pane opens, as shown in the following image.



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 *	
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.
Number of Connections	Reject new connections after this many connections are active. Must be between 1 and 20.
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	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.
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.
Allowable Access Attempts	Number of access attempts that will be allowed before invoking the Access Denied Flow.
	3
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
Access Denied Flow	Optional iSM process flow to call when user fails to login within the Allowable Access Attempts.

Back Reset Update

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.

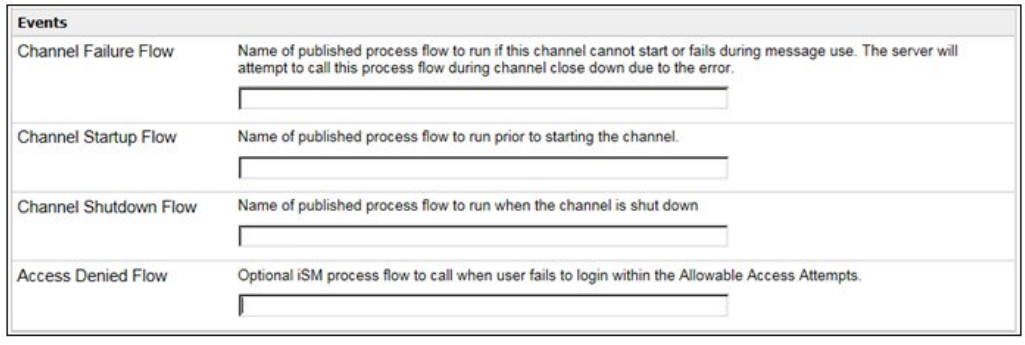
Component Properties	
Name *	Remote1
Description	Brief description of this Remote Console. This field will be displayed along with the console name in the console list. Remote Command Console
Configuration Parameters for Command Channel	
Port *	TCP port for receipt of Command Console requests. 1234
Local 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*.

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	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.
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.
Allowable Access Attempts	Number of access attempts that will be allowed before invoking the Access Denied Flow. 3

Parameter	**Applies to Telnet?**	**Applies to 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)
Allowable Access Attempts	Yes	Yes	Yes

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



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

Note: An asterisk indicates a required parameter.

Parameter	Definition
Component Properties	
Name*	A unique name that will be used to identify the remote command console.
Description	A brief description for the remote command console, which will also be displayed in the Command Consoles pane.
Configuration Parameters for Command Console	
Port*	TCP port for receipt of Command Console requests.
Local Bind Address	Local bind address for multi-homed hosts: usually leave empty
Session Timeout*	The maximum time between commands, in seconds. A value of zero (0) means no timeout. The highest maximum value that can be entered is 10000 seconds. The default value is 600 seconds.
Number of Connections	Reject new connections after the specified number of connections are active. A value between 1 and 20 must be entered. The default value is 1 connection.
Security	

Parameter	Definition
Allowable Clients	If supplied, only messages from this list of fully qualified host names and/or IP addresses are accepted. Enter as a comma-separated list or use the _file() function.
Security Type	Select one of the following values from the drop-down list: ❑ **none.** Implies that the connection and command stream are not encrypted. ❑ **ssl.** Wraps the connection and command stream in an encrypted Secure Socket Layer (SSL). ❑ **ssh.** Provides secure shell (SSH) encryption and packet handling. The default value selected is *none*.
Client Authentication	If set to *true* and when the Security Type parameter is set to *ssl*, then the client's certificate must be trusted by the Telnet server for a connection to be created. Not used when the Security Type parameter is set to *none* or *ssh*.
Authentication Realm	When the Security Type parameter is set to *none* or *ssl*, the specify the name of a configured authentication realm to validate logins. For full access to management commands, the user must be assigned the *admin* role. If not supplied, logins will be delegated to the web console's user database. Not used when the Security Type parameter is set to *ssh*. For SSH console, authentication options are configured in the SSH provider.
Security Provider	Required if security is enabled (Security Type parameter value of *ssl* or *ssh*). This security provider will be used to secure the channel. When the Security Type parameter is set to *ssl*, then specify the name of an SSL Context Provider. When the Security Type parameter is set to *ssh*, then specify an SSH Provider.

Parameter	Definition
Allowable Access Attempts	Specifies the number of access attempts that will be allowed by the client before disconnecting the socket connection. If the value is set to 0, then the client is allowed an unlimited number of access attempts. If the *Access Denied Flow* parameter is defined when the allowable access attempts is surpassed, then the specified process flow will be executed by iSM.
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.
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.
Access Denied Flow	Name of a published process flow to call when login attempts surpass the value specified by the *Allowable Access Attempts* parameter.
Note: Process flows that are published as part of a channel may not be used and cannot be found by the event handler. All event process flows must be published to the system.	

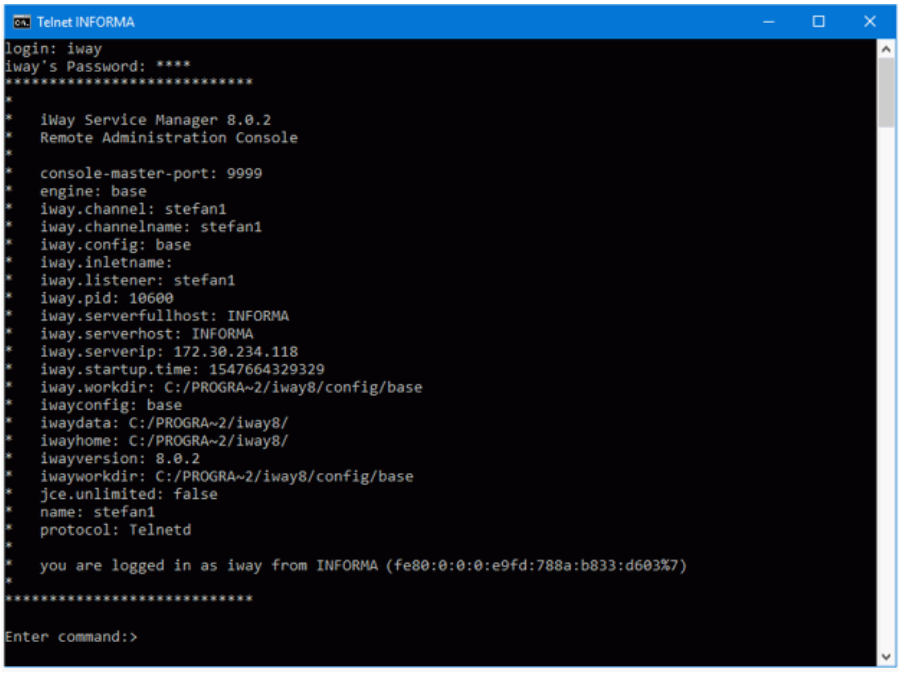
Connecting to a Remote Command Console

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 INFORMA
   ```

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

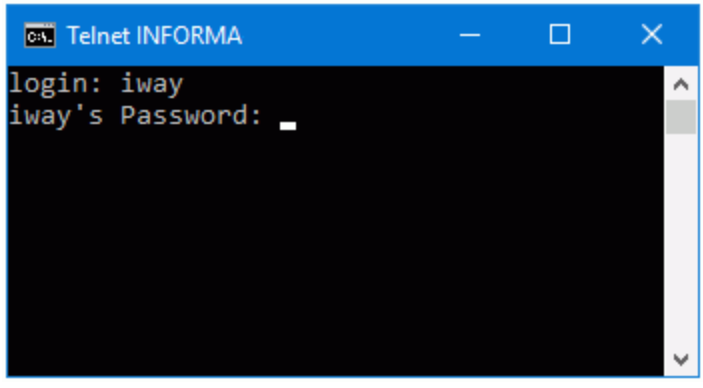


3. Once you are connected and logged in, you can now issue any command to monitor or control your iSM instance.

Using a Telnet Client

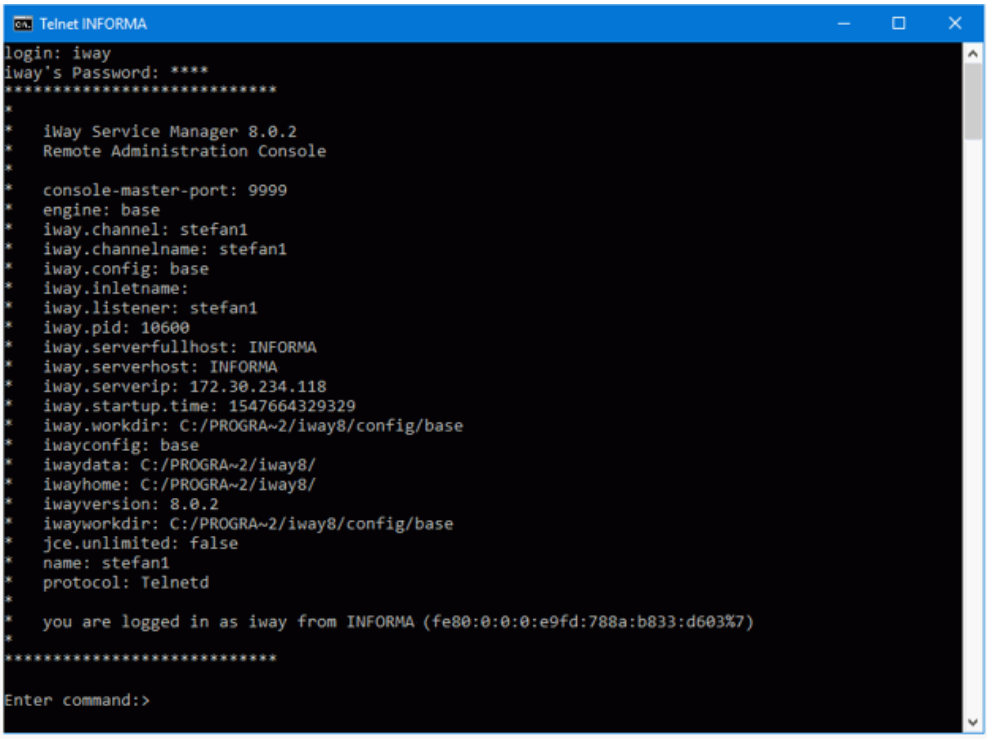
In this section, the default Telnet client that is available on Windows is used for demonstration purposes.

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



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

Once the logon is accepted, you are presented with a standard information screen, as shown in the following image.



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

```
Telnet INFORMA
Enter command:>help
        bam         Business Activity Monitor support command
        calendar    Manage the Service Manager Calendar Provider's data.
        copy        Copy a file from source to target
        cron        Manage the Service Manager Scheduler
                    (also see schedule command)
        diagzip     Create a diagnostic information file for use by iWay Support
        enqueue     Enqueue a message to an internal or ordered queue
        errors      List last errors
        flow        Run a named and published process flow
        gc          Runs the Java garbage collector
        help        Display help for commands.  Use "help <command>..." or "help ifl..." for additional help
        hidelog     hide the trace log
        info        Display channel information
        jdbc        Manage a JDBC provider connection pool
        line        Draw one or more lines on the console
        package     Manage packages
        publish     Publish or remove iIT process flows to the Service Manager
        pull        Load information from another configuration/installation
        quit        Exit the server
        refresh     Reinitialize a channel
        remote      Directs commands to a named configuration
        run         Run a command file
        say         Emits a line to the console and spool
        schedule    Manage the Service Manager Scheduler (also see cron command)
        script      Execute Script via Scripting engine
        set         Set a parameter
        shell       Attempt to run an operating system command
        show        Display server information
        showlog     display the trace log
        sleep       Sleep for a designated period
        spool       Record commands and responses in a spool file
        start       Start one or more channels
        stats       Run statistics on the current instance or listener
        stop        Stop one or more channels
        time        Print the time on the console
        tool        Run a named tool such as 'testfuncs'
        type        Type/display the contents of a text file
        xalog       Start or stop an activity log driver
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.

```
Telnet INFORMA
        remote     Directs commands to a named configuration
        run        Run a command file
        say        Emits a line to the console and spool
        schedule   Manage the Service Manager Scheduler (also see cron command)
        script     Execute Script via Scripting engine
        set        Set a parameter
        shell      Attempt to run an operating system command
        show       Display server information
        showlog    display the trace log
        sleep      Sleep for a designated period
        spool      Record commands and responses in a spool file
        start      Start one or more channels
        stats      Run statistics on the current instance or listener
        stop       Stop one or more channels
        time       Print the time on the console
        tool       Run a named tool such as 'testfuncs'
        type       Type/display the contents of a text file
        xalog      Start or stop an activity log driver
Enter command:>memory
memory max 1842688K used 147456K, free 94781K,
        nodes: total allocated 1526716 namespace 1480714

        Maps: funcs 3, xpath 0, iwxpath 0

        Heap: init 131072K, used 54030K, commit 147456K, max 1842688K
        Non-Heap: init 2496K, used 78418K, commit 79408K, max 0K

        Garbage Collection Info
        Name: PS Scavenge   Collection count: 237   time: 7505ms
        Name: PS MarkSweep   Collection count: 2   time: 294ms

Enter command:>
```

Remote Only Commands

All commands documented in *iWay Service Manager Commands Reference* on page 51 are supported. Two additional commands available only from remote command consoles are:

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

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: 8.0.0
*   iway.serverhost: UbuntuVM
*   iwayworkdir: /iway/prog/8.0.0.1306/config/base
*   iwayconfig: base
*   console-master-port: 9999
*   iway.pid: 3392
*   iway.serverfullhost: UbuntuVM
*   iwayhome: /iway/prog/8.0.0.1306/
*   name: Telnet1
*   doclocation: config
*
*   you are logged in as iway from localhost (0:0:0:0:0:0:0:1)
*
****************************
Enter command:>info
                              completed   failed   active   workers   free
SOAP1
   http          -- active --       0        0        0        3        3
   file          -- active --       0        0        0        3        3
Telnet1          -- active --       0        0        1        1        0
Enter command:>quit
goodbye!
* * * command output end * * *
*
```

Chapter 3

Using the Scheduler

This section describes how to configure the iWay Service Manager (iSM) Scheduler using the command line console to schedule iSM tasks to run at specific time(s) during the hour, day, or month.

In this chapter:

Using the iWay Service Manager Command Line Console

This section provides an overview of the iWay Service Manager (iSM) cron and schedule command line console.

Command Line Basics

The iSM command line console is available directly from the command line of the server when iSM is started as a standalone Java application. If iSM is running as a background task (either as a Windows Started Task or as a UNIX daemon), a Telnet connection to iSM can be used.

Command Line Help

Use the help command to see if the iSM Scheduler extension is installed. Type *help* after the Enter command:> prompt. If the extension has been properly installed, you should see cron and schedule in the listing of commands and descriptions that follows.

```
Enter command:>help
        bam         Business Activity Monitor support command
        calendar    Manage the Service Manager Calendar Provider's data.
        copy        Copy a file from source to target
        cron        Manage the Service Manager Scheduler
                    (also see schedule command)
        diagzip     Create a diagnostic information file for use by iWay Support
        enqueue     Enqueue a message to an internal or ordered queue
        errors      List last errors
        flow        Run a named and published process flow
        gc          Runs the Java garbage collector
        help        Display help for commands.  Use "help <command>..." or "help ifl..." for additional help

        info        Display channel information
        jdbc        Manage a JDBC provider connection pool
        line        Draw one or more lines on the console
        package     Manage packages
        publish     Publish or remove iIT process flows to the Service Manager
        pull        Load information from another configuration/installation
        quit        Exit the server
        refresh     Reinitialize a channel
        remote      Directs commands to a named configuration
        run         Run a command file
        say         Emits a line to the console and spool
        schedule    Manage the Service Manager Scheduler (also see cron command)
        script      Execute Script via Scripting engine
        set         Set a parameter
        shell       Attempt to run an operating system command
        show        Display server information
        sleep       Sleep for a designated period
        spool       Record commands and responses in a spool file
        start       Start one or more channels
        stats       Run statistics on the current instance or listener
        stop        Stop one or more channels
        time        Print the time on the console
        tool        Run a named tool such as 'testfuncs'
        type        Type/display the contents of a text file
        xalog       Start or stop an activity log driver

Enter command:>_
```

The schedule and cron commands also provide help for the user. To access either the schedule or cron help type either of the following commands after the command prompt:

```
help schedule
```

```
help cron
```

The following image shows the help schedule command screen:

```
Enter command:>help schedule
Manages the Service Manager scheduler

  schedule
    List all currently scheduled tasks.

  schedule [list [[-name task] | -all]]
    Get information on the specified task. The -name parm can be entered
    using a wild card to list all tasks that match the pattern.

  schedule add -name task[parameters...]
    Add a new task to the Scheduler. If -save is used the task will be
    added to the Service Manager repository. Otherwise, the task will not
    be available if iSM is restarted.

  schedule suspend [-name task]
    Suspends the task (or all tasks if -name is not specified).

  schedule resume [-name task]
    Restarts the suspended task (or all suspended tasks if -name is not
    specified).

  schedule cancel [-name task] | -all | -save
    Removes a task from the Scheduler.

 The Scheduler is used to run Service Manager commands on a given schedule.
 Each Scheduler entry represents a task and follows a particular format as a
 series of parameters, separated by spaces and/or tabs. Each parameter can
 have a single value or a series of values.  Parameter values that contain
 multiple words must be quoted.

 Tasks scheduled using the command line are only valid during the current
 instance of the Service Manager and are not rescheduled after the Service
 Manager has been recycled, unless the -save switch is used.

Enter command:>
```

The following image shows the cron help command screen:

```
Enter command:>help cron
Manages the Service Manager scheduler in a 'cron like' fashion

  cron
    List all currently scheduled tasks.

  cron [list [[-name task] | -all]]
    Get information on the specified task. The -name parm can be entered
    using a wild card to list all tasks that match the pattern.

    cron add minutes hours dayOfMonth month dayOfWeek command -name taskName
       [-active][-description ...][-user ...][-password ...][-calendar ...]
       [-save]
    Add a new task to the Scheduler.
    The -name parameter is required.
    If -save is used the task will be
    added to the Service Manager repository. Otherwise, the task will not be
    available if iSM is restarted.

  cron suspend [-name task]
    Suspends the task (or all tasks if -name parameter is missing).

  cron resume [-name task]
    Restarts the suspended task (or all suspended tasks if -name parameter
    is missing).

  cron cancel [-name task] | -all | -save
    Removes the -name task from the Scheduler.
    If the -all parameter is included all tasks in the iSM schedule are
    removed.
    If the -save parameter is included the removed task(s) are removed
    permanently from the Scheduler.

 The Scheduler is used to run Service Manager commands on a given schedule.
 Each Scheduler entry represents a task and follows a particular format as a
 series of parameters, separated by spaces and/or tabs. Each parameter can
 have a single value or a series of values.  Parameter values that contain
 multiple words must be quoted.

 Tasks scheduled using the command line are only valid during the current
 instance of the Service Manager and are not rescheduled after the Service
 Manager has been recycled, unless the -save switch is used.

Enter command:>
```

iWay Service Manager Cron Command Console

The cron command allows you to add, suspend, resume or cancel tasks during the current instance of iSM. Unless otherwise noted in the function, tasks that are scheduled using the command console will not be saved in the repository of iSM nor will they be carried over from one instance of iSM to the next when recycled.

Note: The preferred way to Schedule a recurring task is to use the Schedule Provider found in the iWay Service Manager Administration Console.

The cron command console was modeled after the UNIX cron and crontab entry. Most UNIX operating systems have a cron utility that allows tasks to be automatically run in the background at regular intervals by a cron daemon. These tasks are often termed as cron jobs in UNIX. To manage those cron jobs, a file called crontab (short for CRON TABle) is used. This file contains one or more lines, each line a cron job entry to be run at specified times based on the parameters of the line.

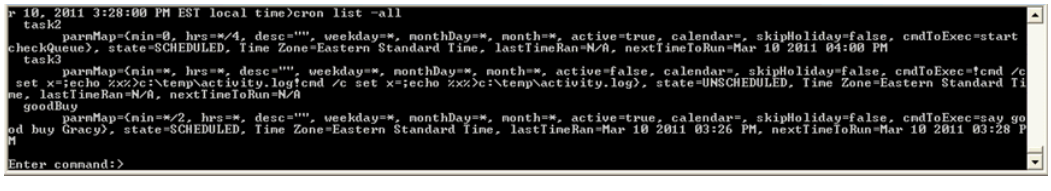
Listing a Task

Entering the command cron or cron list produces a listing of all currently scheduled iSM tasks. To list one or all of the currently scheduled iSM tasks, the general command format is:

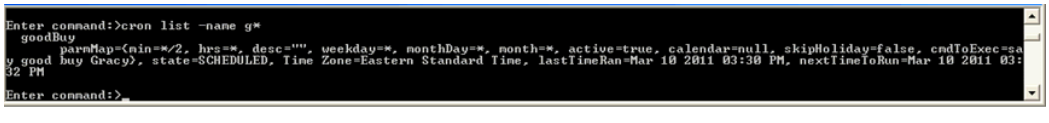
`cron [list [-name task] | [-all]]`

Parameter	Description
`-name`	Is the name or partial name of the task. The name can contain only part of a task name, for example, *t* to list all tasks that start with the letter *t*, *te* to list all tasks that start with the letters *te*, and so on.
`-all`	When the list command is issued without the all parameter, only active tasks are listed. Tasks that are suspended are not displayed. Using the -all parameter lists both active and suspended tasks.

The following image shows an example of the schedule list output.



Entering the command cron list -name <value> (in the following example case 'g*') produces a list of tasks that start with the value *g*. The following image shows an example of the schedule list -name function output.



Adding a Task

Unlike the schedule command of iSM, the cron command is very dependent on parameter positioning. Because the cron command emulates a crontab file entry, the first five fields following the cron add command specifies the day, date, and time followed by the command to be run at that interval. The table below shows the order (from top to bottom) and value of the first five fields that follow the cron add command.

Order	Value
Minutes	* or (0 - 59)
Hours	* or (0 - 23)
dayOfMonth	* or (1 - 31)
Month	* or (1 - 12)
Weekday	* or (0 - 6)
Command	Command to be executed.

Note:

- Day of week value of 0 indicates Sunday.
- An asterisk (*) in the value field above means all legal values as in braces for that column.
- The value column can have a *, a single value, or a list of values separated by commas. A value can be either a number in the ranges shown above or two numbers in the range separated by a hyphen. For example, 1-4 covers the values 1 through 4 inclusive of the numbers 1 and 4, 1-4,7-11 covers the values 1 through 4 and the values 7 through 11.
- Repeat pattern like (called a step) is supported. For example, /2 for every 2 minutes, /10 for every 10 minutes.
- The specification of days can be made in two fields: Day of Month and Weekday. If both are specified in an entry, they are cumulative meaning both of the entries will be executed.

Entering the command cron add will add a new task to the Scheduler. The general command format is:

```
cron add minutes hours dayOfMonth month weekday command -name taskName
   [[-active][-description ...][-user ... -password ...]
[-save]]
```

If the -active flag is included in the command line, then the task is immediately scheduled to run at the next scheduled time.

Parameters	Description
Minutes	A numeric value between 0 and 59 representing when within the hour the task should run. Optional * = all minutes in the hour (same as 0-59).
Hours	A numeric value between 0 and 23 (where 0=12am, 23=11pm) representing when within the day the task should run. Optional * = all hours in the day (same as 0-23).
dayOfMonth	A numeric value between 1 and 31 representing what day in the month the task should run. Optional * = all days in the month).
Month	A numeric value between 1 and 12 (where 1=January, 12=December) representing what month the task should run. Alternately the name of the months may also be used, for example: January[jan], February[feb], March[mar], and so on. Optional * = all months in the year (same as 1-12).
Weekday	A numeric value representing what day of the week the task should run, Sunday (0) through Saturday (6). Alternately the name of the weekday may also be used, for example: Sunday[sun], Monday[mon], Tuesday[tue], Wednesday[wed], and so on. Optional * = all days of the week (same as 0-6).
Command	The iSM command (for example, start listenerName, stop listenerName, and so on) that the task will execute. For more information, see the *iWay Service Manager User's Guide*.

Parameters	Description
-name	Enter a unique name to associate to the task. This name will be used when looking up the task later.
-description (optional)	Enter a brief description of the task.
-active (optional)	If set to true, the task is scheduled to run immediately. If the value is false, then the task is not scheduled to run. If it is missing or set to false (the default), the status of the task is set to UNSCHEDULED.
-user (optional)	If the task must be run with an alternate user ID, enter the ID.
-password (optional)	If the task must be run with an alternate user ID, enter the password of the alternate user. (Required if -user is specified, ignored if -user is not included.)
-save (optional)	Save the added scheduled task in the Schedule repository. This ensures that the schedule and any changes are persisted when iSM is recycled.
-skipHoliday (optional)	If 'Skip Holidays' flag set ('true'), then the days checked in the Schedule's calendar are skipped.

For example, if you want to execute a shell command to clear the error_log file located in the directory c:wwwapachelogs every day at midnight, issue the following:

```
cron add 0 0 * * * "!cmd /c set nada=;echo %nada% > c:logserror_log" -name
task1
```

To start an iSM listener called MonthEnd on the last business day of the month (will not start the listener up on Saturday or Sunday) at 8am:

```
cron add 0 8 @LBDOM * * "start MonthEnd" -name task2
```

To start an iSM listener called MonthEnd on the last day of the month at 8am:

```
cron add 0 8 @LD * * "start MonthEnd" -name task3
```

Note:

- Time parameters not entered (for example. -month, -day, -weekday, etc.) will default to the asterisk ('*') operator (meaning all possible values for that field).
- There are several ways of specifying multiple date/time values in a field:
  - The comma (,) operator specifies a list of values, for example: 1, 3, 4, 7, 8, and so on.
  - The dash (-) operator specifies a range of values, for example, 1-6 which is equivalent to 1, 2, 3, 4, 5, 6.
  - The values may also be combined as well, for example: 1-4, 6, 8-10 is equivalent to 1, 2, 3, 4, 6, 8, 9, 10.
- The asterisk (*) operator specifies all possible values for a field. For example, an asterisk in the hour time field would be equivalent to every hour (subject to matching other specified fields).
- The step operator (/) is valid only for the minutes and hours fields, which can be used to skip a given number of minutes or hours. For example, */3 in the minute time field is equivalent to 0, 3, 6, 9, and so on. While (*) specifies every minute, the */3 means only those minutes divisible by 3. When the step operator does not have an asterisk value (*) preceding the (/) specifier, this means starting now executes again in n minutes or hours rather than every n minutes or hours. For example, /2 in the minutes field indicates execute again in 2 minutes, /5 executes again in 5 minutes and so on.
- Commands and descriptions that consist of multiple words must be enclosed in double quotes ("). For example, start checkQueue or This task is run only at year's end.

Canceling a Task

The cron cancel command is used to remove a task or all tasks from the Scheduler of iSM. Any tasks that were canceled (but not saved) will resume as scheduled only when iSM restarts.

To cancel a task from the schedule of iSM, the general command format is:

```
cron cancel -name task [-save] | -all]
```

Parameter	Description
`-name`	Name of the task to cancel. The named task will be canceled and removed from the iSM schedule.

Parameter	Description
`-save`	The named task is removed from the Schedule repository of iSM and will not be available when iSM is restarted.
`-all`	All tasks currently in the Scheduler of iSM are immediately removed.

Suspending a Task

The cron suspend command is used to suspend the next scheduling a task (or all tasks) in the iSM Scheduler.

Note: Any tasks that were currently executing will complete execution but will not be rescheduled. The suspend command only prevents the task from being scheduled in the future.

To suspend a task within the schedule of iSM, the general command format is:

```
cron suspend [-name task]
```

Parameter	Description
`-name`	Optional name of the task to suspend. The named task scheduling will be suspended; the task will remain on the Schedule list of iSM with a status of SUSPENDED. To restart the task at its next regularly scheduled time use the cron resume command. 1. If -name is not supplied ALL SCHEDULED tasks will be suspended. 2. If the suspend command is issued while a task is running, the running task is not interrupted but will complete normally and will not be rescheduled. 3. The suspend command only prevents the task (or tasks) from being scheduled in the future.

Resuming a Task

The cron resume function resumes the scheduling of a suspended task (or all tasks) in the scheduler of iSM.

To resume a SUSPENDED task in the schedule of iSM, the general command format is:

```
cron resume [-name task]
```

Parameter	Description
`-name`	Optional name of the task to resume scheduling. The named task execution will be scheduled to start at its next regularly scheduled time. **Notes:** If -name is not supplied, all SUSPENDED tasks will be scheduled to execute at their next regularly scheduled time.

iWay Service Manager Schedule Command Console

The schedule command allows you to add, suspend, resume or cancel tasks during the current instance of iSM. Unless otherwise noted in the function, tasks that are scheduled through the command console will not be saved in the repository of iSM nor will they be carried over from one instance of iSM to the next when recycled.

Unlike the cron interface named parameters may be entered in any order (for example -hour may proceed -minute, -name may follow -command, and so on). The command line input is evaluated from left to right following the schedule command verb (add, list, cancel, or delete). Each parameter is terminated by the start of another named parameter or by a carriage return.

Note: Named parameters are parameters that are proceeded by a dash ('-') then the parameter name, for example, -name, -hour, and so on.

Listing a Schedule

Entering the command schedule or schedule list produces a listing of all currently scheduled iSM tasks. To list one or all of the currently scheduled iSM tasks the general command format is:

```
schedule [list [-name task]]
```

Parameter	Description
-name	Is the name or partial name of the task. The name can contain only part of a task name, for example, *t* to list all tasks that start with the letter *t*, *te* to list all tasks that start with the letters *te*, and so on.
-all	When the list command is issued without the all the -all parameter, only active tasks are listed. Tasks that are suspended are not displayed. Using the -all parameter lists both active and suspended tasks.

The following image shows an example of the schedule list output.

```
Enter command:>schedule list
  BeginningOfMonth
        parmMap=(min=0, hrs=0, desc=Schedule BeginningOfMonth listener to start at mid
night on the first day of every month, weekday=*, monthDay=1, month=*, active=false,
password=null, user=null, cmdToExec=start BeginningOfMonth), state=VIRGIN, lastTimeRa
n=N/A, nextTimeToRun=N/A
  ClearTempLog
        parmMap=(min=*/8, hrs=*, desc=Every day at 11:59 PM clear the log file in the
temp directory, weekday=*, monthDay=*, month=*, active=true, password=null, user=null
, cmdToExec=!cmd /c set x=;echo %x%>c:\temp\activity.log), state=SCHEDULED, lastTimeR
an=N/A, nextTimeToRun=Thu Apr 15 13:56:00 EDT 2010

Enter command:>
```

Entering the command schedule list -name <value> (in the following example case 'B*') produces a list of tasks that start with the value. The following image shows an example of the schedule list -name function output.

```
        parmMap=(min=*/8, hrs=*, desc=Every day at 11:59 PM clear the log file in the
temp directory, weekday=*, monthDay=*, month=*, active=true, password=null, user=null
, cmdToExec=!cmd /c set x=;echo %x%>c:\temp\activity.log), state=SCHEDULED, lastTimeR
an=N/A, nextTimeToRun=Thu Apr 15 13:56:00 EDT 2010

Enter command:>schedule list -name B
  BeginningOfMonth
        parmMap=(min=0, hrs=0, desc=Schedule BeginningOfMonth listener to start at mid
night on the first day of every month, weekday=*, monthDay=1, month=*, active=false,
password=null, user=null, cmdToExec=start BeginningOfMonth), state=VIRGIN, lastTimeRa
n=N/A, nextTimeToRun=N/A

Enter command:>
```

The schedule list displays the following information on the console.

Values	Description
`name`	Name of the task in the schedule.
`parmMap`	Parameter of the Task. Each Scheduler entry in the dictionary represents a job and follows a particular format as a series of fields, separated by spaces and/or tabs. Each field can have a single value or a series of values. ❑ **min.** A number between 0 and 59 that represents what minute of the hour the task should start execution. Zero (0) will start the task at the top of each hour. ❑ **hrs.** A number between 0 and 23 that represents what hour of the day the task should start execution. Zero (0) representing midnight (12 AM); 23 representing 11 PM. ❑ **desc.** Task description. ❑ **weekday.** A number between 0 (Sunday) and 6 (Saturday) that represents what day of the week that the task should run on. In addition to the numerical representation the weekday names or abbreviations may also be used, for example, Sun, Mon, Tue, and so on.

Values	Description
`parmMap` (continued)	❑ **month.** A number between 1 (January) and 12 (December) that represents what month that the task should run on. In addition to the numerical representation the month names or abbreviations may also be used, for example, Jan, Feb, Mar, and so on. ❑ **active.** If set to true, this task will be scheduled each and every time iSM is recycled. ❑ **password.** If this task is being executed with the credentials of a different user, enter the password of the user. ❑ **user.** If this task must be executed under the credentials of a different user, enter the User ID to use when executing this task. ❑ **cmdToExecute.** The command that the task will execute when the scheduled time comes. Any iSM command may be executed. Some iSM commands make more sense than others to schedule as tasks. ❑ **dependent.** Dependent command to run when duration timer expires. ❑ **monthDay.** A number between 1 and 31 that represents what day of the month that the task should run on. In addition to the numbers the following special values may also be entered: ❑ **@FMOM.** First Monday of the month, can be abbreviated as @FM. ❑ **@LFOM.** Last Friday of the month, can be abbreviated as @LF. ❑ **@LBDOM.** Last business day of the month. This value will return the last calendar workday (Monday through Friday) of the month. This value can be abbreviated as @LBD. ❑ **@LDOM.** Last day of the month, can be abbreviated as @LD.

Values	Description
`parmMap` (continued)	❑ **duration.** Time between the start of the cmdToExecute and when to start the dependent command. There are several ways of specifying multiple date and time values in a field: ❑ The comma (,) operator specifies a list of values (for example, 1, 3, 4, 7, 8). ❑ The dash (-) operator specifies a range of values, for example: 1 - 6, which is equivalent to 1, 2, 3, 4, 5, 6. ❑ The asterisk (*) operator specifies all possible values for a field. For example, an asterisk in the hour time field would be equivalent to every hour (subject to matching other specified fields). ❑ The slash (/) operator (called step), which can be used to skip a given number of values. For example: */3 in the hour time field is equivalent to 0, 3, 6, 9, 12, 15, 18, 21. So * specifies *every hour*, but the */3 means only those hours divisible by 3. The meaning of / specifier, however, means *when the modulo is zero* rather than *every*. If an * does not proceed the / (for example, /2, /5, and so on) it directs the scheduler to execute the command every n cycles where is the number that follows the step.
`state`	Current state of the task. This can be one of the following values: ❑ `VIRGIN`. Task has not been scheduled. This state is an indication that the active flag of the task is set to false. ❑ `SCHEDULED`. Task is scheduled to run. ❑ `RUNNING`. Task is currently running. ❑ `COMPLETE`. Task is complete but has not been rescheduled. ❑ `CANCELED`. Task has been canceled and not rescheduled. ❑ `ERROR`. Task ended in error.

Values	Description
`lastTimeRan`	Time of day that the task was last ran. A value of 'N/A' indicates that the task has not been run during this instance of iSM.
`nextTimeToRun`	When the task is scheduled to run again.

Adding a Task

The add function adds a task to the schedule of iSM. Tasks added using the command line are immediately added to the schedule. Tasks added using the command line will not be persisted to the repository of iSM unless the -save option is specified in the command line.

The general command line format is:

```
schedule add [parameters]
```

Parameters	Description
`-name`	Enter a unique name to associate to the task. This name will be used when looking up the task later. If -name is missing, iSM will generate a name for the task.
`-active`	A value of either true or false. If true, the task is scheduled to run immediately. If the value is false then the task is not scheduled to run. Optional, if omitted, a false value is assumed.
`-minute`	A numeric value between 0 and 59 representing when within the hour the task should run. Optional, if omitted, an * is assumed (all minutes in the hour).
`-hour`	A numeric value between 0 and 23 (where 0 = 12am, 23 = 11pm) representing when within the day the task should run. Optional, if omitted, an * is assumed (all hours in the day).
`-day`	A numeric value between 1 and 31 representing what day in the month the task should run. Optional, if omitted, an * is assumed (all days in the month).

Parameters	Description
`-month`	A numeric value between 1 and 12 (where 1=January, 12=December) representing what month the task should run. Alternately, the text name of the month, (for example, January[jan], February[feb], March[mar],...) can also be used. Optional, if omitted, an * is assumed (all months in the year).
`-weekday`	A numeric value between 0 and 6 (where 0=Sunday and 6=Saturday) representing what day of the week the task should run. Alternately the text name of the weekdays (for example, Sunday[sun], Monday[mon], Tuesday[tue],...) can also be used. Optional, if omitted, an * is assumed (all days in the week).
`-command`	The iSM command (for example, start listener, stop listener, and so on) that the task will execute. (For more information, see the *iWay Service Manager User's Guide*).
`-duration`	Length of time that the task will run prior to the Dependent Command. The format of duration [in seconds] is in the form [xxh][xxm]xx[s]. For example 04h30m45, which creates a duration of 4 hours, 30 minutes, and 45 seconds.
`-dependent`	The iSM command to be executed after the Duration Timer of the task has expired. (For example, start listener, stop listener, and so on) that the task will execute. (For more information, see the *iWay Service Manager User's Guide*).
`-user`	If the task must be ran with an alternate user ID, enter the ID of the alternate user for the value of this parameter.
`-password`	If the task must be ran with an alternate user ID, enter the password of the alternate user for the value of this parameter.
`-save`	Save the added scheduled task in the repository of iSM. This allows iSM to reschedule this task when the current instance is recycled.

Note:

1. Adding a task with the same name as a currently scheduled task will cause the previously scheduled task (with the same name) to be canceled and this new task to be scheduled in place of the old task.
2. Time parameters not entered (for example, -minute, -hour, -day, -month, -weekday) will default to the asterisk (*) operator (meaning all valid values for that field).
3. There are several ways of specifying multiple date/time values in a field: The comma (,) operator specifies a list of values, for example: 1, 3, 4, 7, 8. The dash (-) operator specifies a range of values, for example, 1 - 6, which is equivalent to 1, 2, 3, 4, 5, 6.
4. The asterisk (*) operator specifies all possible values for a field. For example, an asterisk in the hour time field would be equivalent to every hour (subject to matching other specified fields).
5. The slash (/) operator (called step), which can be used to skip a given number of values. For example, */3 in the hour time field is equivalent to 0, 3, 6, 9, 12, 15, 18, 21. So * specifies every hour, but the */3 means only those hours divisible by 3. The meaning of */ specifier, however, means '*when the modulo is zero* rather than *every*.

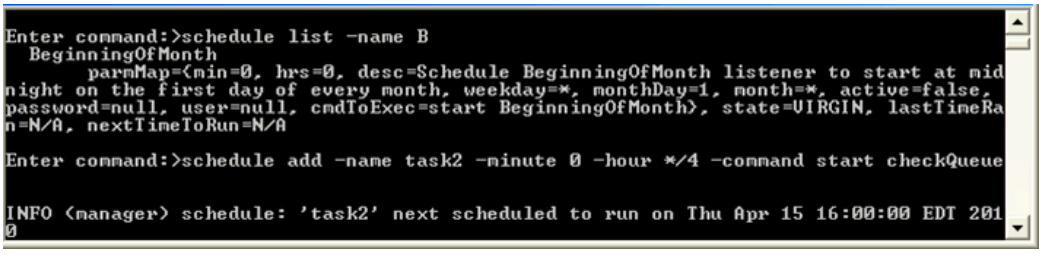
   It the * does not proceed the step (/) (for example, /2, /5, and so on) it directs the scheduler to execute the command every n cycles (minutes, hours, and so on) where n is the number that follows the step (/) character.

Adding a Task to Start a Listener

To add an iSM task named task2 that will start a listener named checkQueue at the top of the hour, every four hours every day of the week, every month of the year, enter the following command (all one line) following the Enter command:> prompt:

```
schedule add -name task2 -minute 0 -hour */4 -command start checkQueue
```

The following image shows the results of the command.

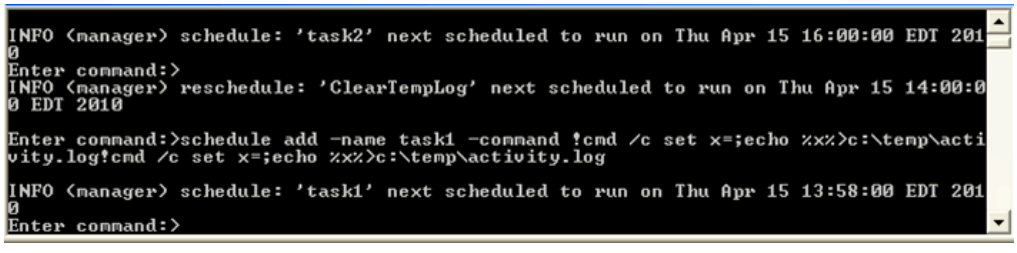


Adding a Task to Execute an External Command

To add an iSM task named task1 that will clear the file activity.log found in the subdirectory c:\temp every minute of every hour of every day of the week every month of the year, enter the following command (all one line) following the Enter command:> prompt:

```
schedule add -name task1 -command !cmd /c set x=;echo
 %x%>c:\temp\activity.log!cmd /c set x=;echo %x%>c:\temp\activity.log
```

The following image shows the results of the command.



Canceling a Task

The cancel function removes the named task from the schedule of iSM. The task, if currently processing, completes its processing cycle prior to being canceled.

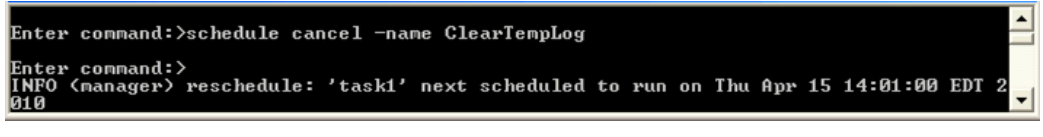
Canceling a task only removes it from the current instance of the schedule. When iSM is recycled, the task (if persisted in the iSM repository) will be rescheduled.

The general command line format is:

```
schedule cancel -name task
```

Parameter	Description
`-name`	Name of the task to cancel. The named task will be canceled and removed from the iSM schedule.

In the following image, the iSM task ClearTempLog has been canceled.



If the task that is canceled has a dependent task configured, the dependent task is canceled from the iSM schedule.

Suspending a Task

The suspend function suspends the scheduling of the next invocation of the task by the scheduler of iSM. The task, if currently processing, completes its processing cycle but will not be rescheduled.

The general command line format is:

```
schedule suspend [-name task]
```

Parameter	Description
`-name`	Optional name of the task to suspend. The named task execution will be suspended. The task will remain on the Schedule list of iSM with a status of SUSPENDED. To restart the task at its next regularly scheduled time, use schedule resume. **Notes:** 1. If -name is not supplied, ALL SCHEDULED tasks will be suspended. 2. If the suspend command is issued while a task is running, the running task is not interrupted but will complete normally; and will not be rescheduled. 3. The suspend command only prevents the task (or tasks) from being scheduled in the future.

If the task that is suspended has a dependent task configured, the dependent task is canceled from the iSM schedule.

Resuming a Task

The resume function resumes the scheduling of a suspended task in the scheduler of iSM. The named task will be scheduled to execute at its next regularly scheduled time.

The general command line format is:

```
schedule resume [-name task]
```

Parameter	Description
-name	Optional name of the task to resume scheduling. The named task execution will be scheduled to start at its next regularly scheduled time. If the -name parameter is missing, all tasks in the iSM schedule that are suspended will resume scheduled execution on their next regularly scheduled time.

If the task that is resumed has a dependent task configured, then the dependent task is only scheduled by the iSM that immediately follows the start of the configured primary task.

Command Line Schedule Examples

This section provides examples on how to configure various schedules using the command line.

Once a Year

To schedule the same task using the command line, use the following command:

```
schedule add -name RunOnNewYear -description Run once a year on New Year's
Day at
   12:01am -minute 01 -hour 0 -day 1 -month January -command run
script.xyz.scr -save
```

Once a Month

To schedule the same task using the command line, use the following command:

```
schedule add -name LDOBScript -description "On the last day of business run
script
   scr.xyz.scr" -minute 0 -hour 0 -day @LBDOM -command run script.xyz.scr -
save
```

Once a Week

To schedule the same task using the command line, use the following command:

```
schedule add -name RunOnMonday -description "Run the script script.xyz.scr
on
   Monday evening at 11:59pm" -minute 59 -hour 23 -weekday Monday -command
run script.xyz.scr -save
```

Daily

To schedule the same task using the command line, use the following command:

```
schedule add -name RunDaily -description "Run the script script.xyz twice
daily;
   once at 12pm (noon) and again at 6pm" -minute 0 -hour 12,18 -weekday
Mon,Tue,Wed,Thu,Fri
   -command run script.xyz.scr -save
```

Appendix A

iWay Service Manager Commands Reference

iWay Service Manager (iSM) provides a collection of commands to assist you when you are running iSM and help you identify and isolate problems during run-time processing. This section provides a reference for the available iSM commands.

Note: iSM commands are case-insensitive. However, names (for example, a channel name to be displayed) are case-sensitive.

Available iSM Commands

iSM commands fall into several categories, which are listed and described in the following table.

iSM Command	Description
Command Consoles Only (Telnet, SSH)	
showlog	Display traces on the command console.
hidelog	Do not display traces on the command console (default).
Investigate the Server (iSM) and JVM/OS	
show	Display desired information (see *show* command).
errors	Display prior error and warning traces.
Change Server (iSM) Characteristics	
set	Set options (see *set* command).
jdbc	Manage a JDBC data pooling provider.
xalog	Start and stop transaction log collection.
Channel Management	
info	Get channel information, such as the number of messages.

iSM Command	Description
quit	Terminate the server (iSM).
refresh	Stop and start a channel applying configuration changes.
start	Start a channel.
stats	Display channel performance statistics.
stop	Stop a channel.
Message Handling	
enqueue	Add a message to an internal queue.
flow	Run a named process flow.
Command Controls	
remote	Perform operations on another configuration/iIA.
run	Execute a command script.
say	Issue a message (used in scripts).
sleep	Pause (used in scripts).
spool	All output to a designated file.
Calendar and Scheduler	
calendar	Manage an iSM calendar provider.
cron	Use cron command syntax to schedule an activity.
schedule	Use iSM extended syntax to schedule an activity.
Miscellaneous	
bam	Manage iWay Business Activity Monitor (BAM) services.
copy	Copy a file.
diagzip	Generate a diagnostic .zip file for iWay Technical Support.

iSM Command	Description
gc	Run the Java garbage collector.
line	Separate information on the console.
package	Install or remove a configuration package.
time	Cause time to be displayed in traces.
tool	Execute s special tool, such as the Debugger.
type	Display a file or directory.

Calendar

The calendar command exports or imports data from a defined Calendar provider.

The calendar command currently supports the following syntax:

```
calendar
calendar -list[ configuration]
calendar -export[ configuration] calendarProvider[ destinationPath][ -
replace]
calendar -import[ configuration] sourcePath[ -replace]
```

Note: Parameters enclosed in brackets [] are optional parameters.

Issuing the command *calendar* without any parameters lists the currently defined Calendar providers.

Parameter	Description
`-list`	Lists the defined Calendar providers for an iSM configuration.
`configuration` (optional)	An iSM configuration name (for example, base). If not specified, then the current configuration is used.
`calendarProvider`	Name of a defined Calendar provider.
`-export`	Exports the Calendar provider to an XML file.

Parameter	Description
`destinationPath` (optional)	If specified, this is where the data file is created. If omitted, then the file is created in the following directory: *<iSM_Home>*`/etc/calendar` If the destination path contains embedded spaces, then the name the value should be enclosed in quotes.
`-import`	Imports a Calendar provider into the specified configuration.
`sourcePath`	Source path to the Calendar provider, which can be entered using one of the following formats: ❑ `file:///path` ❑ `ftp://[user:password@]server/path` ❑ `poolingProvider://host[:port]/configurationId/providerName` ❑ `configurationId/providerName`
`-replace` (optional)	Replaces the existing data file if the file exists. If the file exists and the -replace command is not specified, then the file will not be overwritten and an error is generated.

Cron

The *cron* command allows you to add, suspend, resume or cancel tasks during the current instance of iWay Service Manager (iSM). For more information on using the *cron* command, see *iWay Service Manager Cron Command Console* on page 32.

Copy

Copies a file or directory to a new location.

To use *copy*, issue the following command:

```
copy <from> <to>
```

where:

<from>

Is the path to a file or a directory.

<to>

Is the path to the target.

Note: File references in iSM use forward slash characters (/) for directory separators. Files with blanks or special characters in the name should be enclosed in quote characters (“ “).

Diagzip

The diagzip command creates a diagnostic .zip file. This file contains information useful to an iWay service when problems are encountered.

To use diagzip, issue the following command:

```
diagzip <outfile> [<comment>][[-config <name>][-comments<comments>][-
overwrite]]
```

The .zip file is written to the specified output file. To create a diagnostic .zip file for another configuration, use the *-config* switch, followed by the name of the desired configuration. By default, the diagnostic .zip file is created for the current configuration. if you omit the suffix, the zip suffix will be added.

Use the comments field to explain the reason that the diagnostic .zip file was created. This can be a simple string or the name of a file containing the explanation. If you use a file name, prefix the name with the @ character.

The following table lists and describes the switches that are supported by the *diagzip* command:

Switch	Description
`-comments`	This switch is followed by comments as described in the above sample. Comments can be the second positional parameter or can be denoted by a switch.
`-config`	This switch is followed by the configuration name.
`-overwrite`	By default, the command will not overwrite an existing file. You can use this switch to permit the command to overwrite an existing file

Enqueue

Enqueues a message to a named server internal or ordered queue. Often the enqueue function is used with a Scheduler to cause a channel to take some action at a specified time.

To use *enqueue*, issue the following command:

```
enqueue  <qname> [<input>] [switches]
```

where:

<qname>

Is the name of an existing internal or ordered queue. Although the queue must exist, its associated listener does not need to be active or in a running state.

<input>

The input to add to the queue. If this value is omitted, a signal document (as shown in the *flow* command) is passed. For more information, see *Flow* on page 58.

Note: The input can be contained in a file. To do so, append the *_file()* iFL function to reference the file path. For example:

```
enqueue myq _file('c:/data/myinput.xml')
```

switches

Is a switch you can specify, which provides specific instructions for processing and handling the input/message in the queue. The following table lists and describes the switches that are supported by the *enqueue* command:

Switch	Applies To	Description
`-xml`	internal and ordered queues	The input is in XML format and will be parsed accordingly. This is the default switch for the *enqueue* command.
`-json`	internal and ordered queues	The input is in JSON format and will be parsed accordingly.
`-flat`	internal and ordered queues	The input is in flat format and will not be parsed or manipulated.

Switch	Applies To	Description
`-priority`	internal queue	Indicates the priority of the message when the target is an internal (not ordered) queue. The default priority is 4.
`-key <key>`	ordered batch queue	The key that identifies this message group in the ordered batch queue.
`-delete`	ordered batch queue	Sends the delete batch (of <*key*>) group in the ordered batch queue.
`-end`	ordered batch queue	Sends an end of batch signal, causing the batch identified by the <*key*> value to be released for processing.
`-map <pairs>`	internal and ordered queues	Adds *token=value* pairs to the standard signal document if used, as the parameter map. The pairs will also be set as DOC level special registers (SREGs) in the execution environment. This must be the last switch specified on the line. In addition, all tokens that follow this switch are considered as *token=value* pairs. The equal (=) and comma (,) characters are optional.

The message (or end of batch signal) will be passed to the process flow as the identified queue is processed by its associated listener.

The following is an example of issuing the enqueue command.

```
Enter command:>enqueue iqueue1 -map one=111<signal protocol="enqueue"
timestamp="2015-04-20T13:26:29.509Z"
type="command" version="2">
    <parms count="1">
       <parm name="one">111</parm>
    </parms>
</signal>
```

Errors

Lists the last few errors traced by the server.

The following command is used for errors:

```
errors[reset]
```

If reset is specified, the error list will be cleared.

Flow

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

The process flow execution is not directly connected to the resources of iWay Service Manager (iSM). As a result, the use of published transforms, subflows, providers, or other resources that are external to the process flow itself may not be located.

To issue a *flow* command, enter the following:

```
flow <flowname> [-input filepath] [-xml|-json|-flat][-output filepath] [-
iwp exported path] [-commit] [-debug] [-deep] [-tree] [-expect filepath][-
map ...]
```

The following table lists and describes the parameters for the *flow* command.

Parameter	Description
flowname	Is the name of the process flow. The process flow must have been published to the system area of the configuration under which it is to be run.

Parameter	Description
-input filepath	A file containing the input to be supplied to the process flow. If parsed (non-flat), then the input must be in a form to be parsed for execution. The following switches determine the input type: ❑ **-xml.** The input is an XML document that is parsed into an internal form for execution. This is the default value. ❑ **-json.** The input is in JSON format that is parsed into an internal form for execution. ❑ **-flat.** The input is not parsed, but is presented as a string to the process flow.
-output filepath	The output of the process flow is presented in this file. If -output is not present, then the process flow output is not presented. iSM unique file naming is supported. For example: `-output /myarea/fileout/test###.txt`
-iwp filepath	Executes an unpublished process flow exported from iWay Integration Tools (iIT). In this case, the *flowname* operand is used only for documentation purposes and is ignored, although it must be provided.
-commit	Commits the named process flow. In addition, this parameter runs the process flow transactionally. If this parameter is omitted, then the process flow is not run under transactional control.
-debug	Sets the trace log to *debug* mode for this execution.
-deep	Sets the trace log to *deep* mode for this execution.
-tree	Sets the trace log to *tree* mode for this execution.
-expect filepath	If present, then the output will be compared to the expected output. This is often useful to validate a process flow against a prior version. The formats and layouts must match.

Parameter	Description
-map <pairs>	Adds *token=value* pairs to the standard signal document if used, as the parameter map.The pairs will also be set as DOC level special registers in the execution environment. This must be the last switch on the line, and all tokens that follow it are considered as *token=value* pairs. The equal (=) character and comma (,) characters are optional. For example: `-map a=b, c d`

Example

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

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

Because no input was specified in the command, a standard signal document will be passed to the process flow, which will have the following structure:

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

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

```
match
```

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

```
nomatch
```

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

```
Enter command:>flow passthru _file(c:/docs/flowin.xml) -expect c:/docs/
expect.xml
Flow 'passthru' OK, not committed
Unequal compare:
-- Lengths are not equal
Length expected=54, actual=53
Difference starts at char 46,  expected=c'b'/d'98', actual=c'<'/d'60'
Partial Expected: >aaaab</Test>
Partial Actual: >aaaa</Test>
Expected: <?xmlversion='1.0'encoding='UTF-8'?><Test>aaaab</Test>
Actual  : <?xmlversion='1.0'encoding='UTF-8'?><Test>aaaa</Test>
nomatch
Enter command:>
```

GC (Garbage Collector)

Runs the Java garbage collector. This command is not a library for memory issues, and should not be needed or used constantly.

Help

Displays a list of the available iSM commands.

- **help <command>**: Displays information on the specific command.
- **help <command> <subcommand>**: Displays information on the subcommand. The special command ifl presents information on the available iFL functions.

For example:

```
Enter command:>help ifl
```

```
Enter command:>help ifl
_add            _aes            _after          _all            _any
_attbyfname     _attcnt         _attflatof      _atthdr         _atthdric
_base64         _before         _cat            _ceil           _chaninfo
_concat         _cond           _contains       _count          _dadd
_dateadd        _datediff       _dateof         _datesub        _ddiv
_decode64       _deentity       _deflate        _div            _dmul
_docinfo        _dsub           _encode64       _encr           _endswith
_entity         _eval           _excel          _excelsheets    _exists
_exists1        _fetch          _file           _fileexists     _filegdg
_fileinfo       _flatof         _floor          _flow           _fmtdate
_fmtdec         _fmtint         _fn             _frombase64     _fromhex
_ftstamp        _getprin        _getschedulevalue _hasrole      _hasruleerr
_hasschemaerr   _hex            _hl7ack         _iadd           _idiv
_if             _imod           _imul           _indexof        _inflate
_inlist         _int            _intmask        _isdob          _isendpoint
_iseos          _iserror        _isflat         _isjson         _isnumber
_isreg          _isroot         _isscheduled    _isub           _iswellformed
_isxml          _iwexists       _iwxflat        _iwxpath        _jdbc
_jsonpath       _jsonptr        _lcase          _ldap           _left
_length         _lit            _lock           _log            _manifest
_match          _max            _md5            _min            _mod
_mod10          _mul            _murmurhash     _normalizespace _now
_pad            _parmof         _printable      _property       _propertymatch
_qval           _random         _regex          _replace        _restoredoc
_reverse        _right          _root           _round          _savedoc
_script         _scriptlist     _setreg         _setschedule    _sha1
_sha256         _sha384         _sql            _srcname        _sreg
_startswith     _sub            _substr         _sysinfo        _timer
_token          _tpa            _tpah           _tpaw           _tpid
_tpn            _tpp            _tpr            _tps            _tpsql
_tpt            _tpxmlsql       _treehash       _trim           _tstamp
_ucase          _unq            _url            _urldecode      _urlencode
_urlparse       _uuid           _xflat          _xflat1         _xml
_xpath          _xpath1         _xquery         _zcert          _zconfig


Enter command:>
```

A list of the parameters for any function can be obtained by entering its name.

```
Enter command:>help ifl _ldap
```

```
_ldap     Get attribute from LDAP repository
   1   filter      required    Selection expression
   2   attribute   required    Attribute to be accessed from the repository
   3   context     optional    Context to be applied to the search
```

Hidelog and Showlog

The *hidelog* and *showlog* commands toggle displaying or hiding the trace log. They are only available in the Telnet / SSH command consoles. These commands are not applicable to the main iSM command window.

Info

Reports on the status of each listener and channel.

The command `info [<name>]` is given to report on one or all channels. If the name is included, the configuration is reported.

```
Enter command:>info

                                completed   failed   active  workers    free
http          --active--            0         0        0        3         3
file          --active--            0         0        0        3         3
emittest      --active--            0         0        0        1         1
file1         --active--            2         0        0        1         1
ap            --active--            0         1        0        0         0
gw            --active--            2         0        0        1         1
```

In the sample shown above, two documents have been processed by the *file1* and the *gw* channels. These two documents completed processing successfully, which does not mean that they necessarily had no application errors. There is one failed document on the *ap* channel. This means that the message did not complete processing but rather failed to complete processing due to an execution, not an application, error. Examples of such errors are having no wired edge in a process file that the document could follow (this is usually an application design error) or a programming problem encountered during the execution path of the message. Since a message can be sent to many output destinations, this is specifically not the number of documents routed to a reply or an error destination.

The state of the channel can be:

Channel State	Description
active	The channel is processing messages or is waiting to process messages.
inactive	The channel is not marked to be started automatically and must be started manually.
config	The channel did not start due to a configuration error.

Channel State	Description
retry	The channel has not been able to initialize due to external conditions. It is scheduled for a startup after a defined period.
stopping	A stop command has been received and is being processed.
stopped	The channel is not active and is awaiting a start command.
license	The channel cannot start due to a detected license violation.

The active column is the number of messages currently being processed. Workers shows the number of subchannels available to process messages, and free shows those subchannels available to process workers.

Jdbc

The *jdbc* command manages the connection pool associated with a JDBC Provider.

The *jdbc* command uses the following format:

```
jdbc <providername> clear [-t <timeout>]
```

or

```
jdbc <providername> limits [-idle <maxidle>] [-active <maxactive>]
```

Clear

The *clear* option closes and removes connections from the connection pool.

- **provider name.** The name of the JDBC provider toward which the command is directed.
- **timeout.** The maximum time, in seconds. where the clear command blocks waiting for active connections to be returned to the pool. The default setting is to wait indefinitely.

Notes:

- Execution of this command drops all connections from the pool, but leaves the pool itself intact. Thus, when the next user attempts to borrow a connection from the pool, a new connection will be added to the pool for that user. This means that if a user wants the pool to remain empty for some period of time, channels that use the pool should be stopped before the command runs.
- The pool cannot be cleared completely until all active connections have been returned. If a timeout is not specified, this command will block until all connections have been returned to the pool and closed. If a component borrows a connection from the pool but does not return it-for example, XDSQL Agent using the JDBC provider along with the agent's own connection caching facility-the command would block indefinitely. Avoid this situation by stopping all channels that use the pool before executing this command.
- While the command is actually executing, callers will be unable to borrow connections from the pool. Any attempt will result in an SQL Exception with the message, *unable to get pooled connection from this provider*. Once all connections have been cleared, however, the pool will become available, as described above.

Limits

The *limits* option can reset the limits of the pool for the maximum number of idle connections and the maximum number of active connections.

- **provider name.** The name of the JDBC provider toward which the command is directed.
- **idle.** New value for maximum number of idle connections to retain in the pool.
- **active.** New value for maximum number of active connections allowed by the pool.

Notes:

- You must supply at least one of the *idle* or *active* options.
- The maximum number of idle connections must always be less than or equal to the maximum number of active connections.
- This command changes only the runtime values of these limits. When restarting iSM, the pool will initialize with the previously saved values.
- This command does not affect connections currently in the pool. After resetting the maximum number of idle connections, the pool can contain more idle connections than the new value, until these connections are borrowed and returned, reach their idle limit and are evicted, or are removed by the *clear* command.

Line

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

The *line* command uses the following format:

```
line [<count>] [-log]
```

where:

<count>

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

-log

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

For example:

```
Enter command:>line 2

---------------------------------------------

---------------------------------------------
```

Package

Packages encapsulate components to be applied into an existing iSM configuration or iWay Integration Application (iIA). Packages are built using the iSM Administration Console by selecting the specific components to be included. The *package* command enables packages to be managed through a command console or within a script.

To use *package*, issue the following command:

```
package  <action> <name> [-config <name>] [-directory <name>] [-version
number]
```

where:

<action>

Is the specific action to apply. Supported actions are listed and described in the following table.

Action	Description

`add`
or
`merge`
	Merges a package into the configuration.

`delete`
or
`unmerge`
	Removes a package from the configuration.

`list`	Lists the packages that are currently installed.

The following table lists and describes the switches that are supported by the *package* command:

Switch	Description
`-config`	Name of the iSM configuration or iIA. If omitted, then the current iSM configuration is addressed.
`-directory`	Location of packages if not in the standard install location. This switch applies only to the *merge* action. **Note:** Packages are built in a standard location in your installation by the iSM Administration Console. You can move your packages to an external directory for distribution if required.
`-version`	The version for the package. The default value is 1.

Pull

Takes settings from a named configuration to the current configuration. The command can import configuration-level special registers or providers. Optionally it can replace object of the same name in the target configuration.

When special registers are imported, the read-only iWay register such as iwayconfig cannot be replaced.

Example: pull register *filein* from configuration *other*:Pull other register filein

Quit

Terminates this server configuration. The quit command cannot be issued by a remote management console. All channels must be stopped for the quit command to be used.

Refresh

Stops and restarts the specified channel. Resources are released and reacquired, and any changes made to the configuration of the channel are applied.

To use *refresh*, issue the following command:

```
refresh <name>
```

where:

<name>

Is the name of the channel to be refreshed.

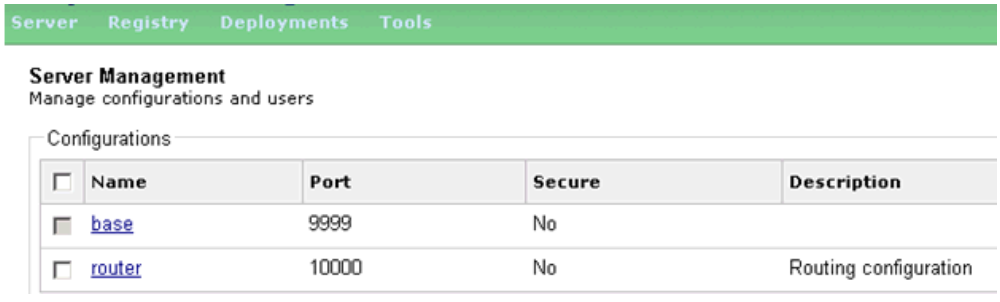
Remote

Establishes a connection to a remote configuration receiver for one command or for subsequent commands. The named [and possibly remote addressed] configuration must exist and be operational.

A single command can be executed by being declared directly on the remote command line following the `-execute` operand. In this case the connection is established, the command is executed, and then the session is disconnected. A common use of this command form is to use a single [local] scheduler to cause commands to be issued to other configurations or iWay Integration Applications (iIAs).

If a single command is not specified, then a link is opened for subsequent commands. A common use of this form is to run commands to control several activities on the addressed configuration or iIA. In such a use case, once established, until the receiver is reset to the current configuration, all commands are executed by the addressed remote configuration.

You can also remote to a configuration running on another installation. To do this, you need to know the URL to reach the command handler of the configuration or the URL of the master configuration on that host. Each configuration has a command port which is shown in the control console for that configuration. Usually, the port for the base configuration on an installation is 9999, but can be changed during the installation. As additional configurations are created, they are assigned port numbers increasing by a value of one (for example, 10000, 10001, and so on). Again, this can be changed when the configuration is created. The port is in the Server Management (Manage configurations and users) page, as shown in the following image:



Some commands cannot be executed by a remote configuration. These include spool and remote.

```
remote -reset | -display | -host URL [-config <configuration name>] [-user
user] [-password password] [-execute <command>]
```

Keyword	Use	Comments
`-display`	Show the current state of the remote connections.	
`-config <string>`	Address the named configuration.	The keyword -config is optional in which case the name should be the first operand.
`-host <url>`	URL of the iSM (*iwayhome*) host.	The default is the current configuration. The URL should be the address of the master configuration. The default port is 9999.

Keyword	Use	Comments
`-user <string>`	Authorized user on the remote configuration.	The default is the current user.
`-password <string>`	Authorized credential for this user on the remote configuration.	The default is the current password.
`-reset`	Disconnect from the remote target and return to the local command handler.	
`-execute <command>`	One command to be executed on the remote computer.	Surround multiple token command with quotes.

Examples:

1. This example establishes a remote session to a local configuration named *test* and starts several channels.

   ```
   ->remote test
   ->start chan1*
   ->start two*
   ->remote -reset
   ```

   Note: The `start chan1*` command starts all channels that begin with *chan1*.

2. This example stops a single channel on a remote configuration named *test* on the host *otherhost*.

   ```
   remote test -host otherhost  -exec 'stop one*'
   ```

Run

Runs a batch file (script) of iWay commands. Any commands can be in the command file, however information issued may not be found. All commands are "evaluated" using the iWay Functional Language, using the special registers of the manager only.

As the server starts, it looks in the configuration root for a file autocmd.txt. If found, it runs this command file.

Within a command file, the goto <label> command is available to skip forward. Labels end with colons. A goto command can be the subject of an IFL if().

The special token $cmdstatus$ is replaced with the value of the command status from the prior command. To use this token, the cmdstatus feature must be enabled. See set command.

A run file can contain comments, which begin with // in any column.

```
_if( _now('E')='Tue',goto tuesday)
start (chan1, chan2)     // the non-tuesday channels
goto everyday
tuesday:
start chan_tues
_if($cmdstatus = 0),goto end)
say command to start channel chan-tues failed with status $cmdstatus
everyday:
```

The following is issued when performing a run command:

```
run <scriptpath [-trace] [-loop <count>] [-map pairs…]
```

where:

`scriptpath`

Is the script to be run.

`-t[race]`

Is each line of the displayed scripted as it is executed.

`-l[oop]<count>`

Is the number of executions of each line. If the count exceeds the <count> value, the script is terminated. This is intended for prevent loops by uncontrolled GOTO's. The default is 50. To eliminate the loop counting, set the count to zero (0).

`-s[toponerrors]`

If set, the script will terminate on any detected errors.

`-map pairs`

This must be the last switch/parameter on the command line. All tokens that follow it are considered as token=value pairs. The = sign and commas are optional. These token/value pairs are treated a script parameters. To include the value of a parameter on a line of the script, designate the token as $token$.

For example:

```
run  myscript.txt -map name=Weekday
```

The script might contain the line:

```
say Script is $name$
```

with the resultant message emitted:

```
Script is Weekday
```

The Start command offers the *-doflow* switch to cause a defined channel failure startup flow to execute if the channel fails to start. You may wish to consider this in your startup script.

Say

Emits the remainder of the command line to the console and, if enabled, to the spool. The line to be emitted is evaluated for iFL and the $cmdstatus$ token is replaced with the prior command status.

Schedule

The *schedule* command allows you to add, suspend, resume or cancel scheduled tasks during the current instance of iWay Service Manager (iSM). For more information on using the *schedule* command, see *iWay Service Manager Schedule Command Console* on page 39.

Script

Scripts extend the capabilities of the iSM server by enabling users to offer services through interpreted programming languages, such as JavaScript and Python.

The Mozilla Rhino engine for the JavaScript programming language, is currently included as a feature in the iSM libraries. The iSM platform can use any script engines that comply with JSR 223. The website scripting.dev.java.net hosts an open project to maintain several script engines that conform to JSR 223 (like Python).

The site also links to engines maintained elsewhere. You can learn more about the embedded JavaScript technology engine by visiting the Mozilla Rhino website.

The script iSM command allows you to invoke scripts dynamically and review the results using the iSM console. The iSM console format is:

```
script location [-entry scriptFunction][ p1[ p2[ …pN]]]
```

Parameter	Description
`location`	The script file location or script input string. ❑ The script file location is the absolute path to the script file itself. When creating and saving a script file the extension of the file identifies the type of script it is. For example, if the extension is .js, then the Script type is JavaScript. If the extension is .py, then the Script type is Python. ❑ If the location fails to be an absolute file path then it is assumed to be a script input string. Script input strings are assumed to be JavaScript file and is processed as such. The input string must be enclosed in double quotes. All double quotes contained in the input string must be escaped with the backslash (/) character.
`scriptFunction`	Name of the function within the script to execute.
`p1 through pN`	The script parameters.

Imbedded Spaces

The command line is parsed on white space between elements, therefore the input `script c:/program files/scripts/f2.js -entry func1 par1 par2` will result in the following parameter array being processed by the script engine:

```
script
c:/program
files/scripts/f2.js
-entry
func1
par1
par2
```

If you want to pass an embedded space (for example, a path like c:/program files/scripts/f2.js) then the parameter needs to be quoted (either single or double), as shown below:

```
script "c:/program files/scripts/f2.js" -entry func1 par1 par2
```

This command line results in the following parameter array being processed by the script engine:

```
script
-file
c:/program files/scripts/f2.js
-entry
func1
par1
par2
```

This will load the script file c:/program files/scripts/f2.js and call the function func1(p1,p2) with parameters par1, par2.

If you want to embed a quote (either single or double) into a parameter the quote needs to be escaped by using the backslash (\) character.

For example, `script "function printVariableA(a) { return (\"the variable is:\\"\"+a+\"\\"\"); }" -entry printVariableA foo` produces the following output:

```
SCRIPT: results: 'the variable is:"foo"'
```

Likewise, the line `script "function printVariableA(a){return('the variable is:\\''a+a'\\'');}" -entry printVariableA foo` produces the following output:

```
SCRIPT: results: 'the variable is:'foo''
```

The (\\) is necessary to exit the escape character.

The same holds true for variables passed to the following script:

```
script "function printVariableA(a) { return('the variable is:\\''+a+'\
\''); }" -entry printVariableA "I pitty da\'fool"
```

The script above, produces the following output:

```
SCRIPT: results: 'the variable is:'I pitty da'fool''
```

To pass a Special Register to the script the command line could look like this:

```
script "function printVariableA(a) { return('the variable is:\\''+a+'\
\''); }" -entry printVariableA _SREG(engine)
```

It results in the following output:

```
SCRIPT: results: 'the variable is:'base''
```

To pass a Special Register to the script as a named variable, the command line could look like this:

```
script "print('the variable is:\\''+a+'\\'');" _CONCAT('a=',_SREG(engine))
```

It produces the following output:

```
SCRIPT: results: 'the variable is:'base''
```

Passing Variables, Arrays, and Stacks

To assign a value, an array, or stack to a variableName, use the following format:

Type	Format
single	variableName=value
array	variableName=val1,val2,val3,...
stack	variableName=valn,val(n-1),val(n-2),...val0

To pass value, an array, or stack to a script function as a parameter the format is:

Type	Format
single	value
array	val1,val2,val3,...
stack	valn,val(n-1),val(n-2),...val0

iWay Service Manager Objects

The scripting engine supports the following embedded iSM objects:

Variable Name	Description
XDDOC	Current XDDocument (available only when executed as an iFL).
XDSRM	Special Register Manager
XDMGR	Current XDManager

Accessing of the iSM objects is just the same as accessing any other built in JavaScript object.

For example, object.reference where object is the name of the built in object. Document, XDDOC, and reference is either a method or variable within the object.

XDDOC

The XDDOC object is valid only when the script engine is invoked as an iFL. it contains the current XML document and allows the script developer access to the XML document being processed. For example, the following JavaScript finds all of the elements with the name title and produces an html document containing an ordered list of 'titles' values:

```
importPackage(com.ibi.edaqm);
var xmlRoot=XDDOC.getRoot();
var nodeList=xmlRoot.findAllByName('title').toArray();
document.write('<h3>Titles:</h3><p>');
// build an ordered list for display
document.write('<ol>');
for(x in nodeList)
{
    document.write('<li>'+nodeList[x].getValue()+'</li>');
}
document.write('</ol>');
```

If you are using the XML document (bookstore.xml), then you would receive the following results:

```
<h3>Titles:<h3><p>
<ol>
<li>Everyday Italian</li>
<li>Harry Potter</li>
<li>XQuery Kick Start</li>
<li>Learning XML</li>
</ol>
```

The JavaScript imports the com.ibi.edaqm package that contains the definition of the XDDocument object. The getRoot method obtains the root element of the document. The method findAllByName obtains a list of all elements that are named 'title' the toArray method converts the list to a String array that the for(x in nodeList)statement may iterate through.

For more information, see the XDDocument description found in the *iWay Service Manager Programmer's Guide* for the methods and variables of the object.

XDSRM

The XDSRM object is available to the iSM Administration Console, as well as scripts invoked using the iWay Functional Language (iFL) interface. The XDSRM object allows the JavaScript developer access to the iSM Special Register (SREG) mappings. If invoked from the iSM Administration Console, then only those SREGs are assigned at the global level (set through the Server, Register Settings area of the iSM Administration Console).

When invoked as an iFL the object has reference to any SREG set within the current channel context such as the thread of execution in a process flow, upwards to the system manager. For example, the following JavaScript looks up the SREG mapping and iterates through the key set of the map to get the name and value of the SREG:

```
importPackage(com.ibi.edaqm);
document.write('<h2>Simple Ordered List of Special Registers:</h2><p>');
var map = XDSRM.getSpecialRegisters();
var keys = map.keySet().iterator();
// build an ordered list for display
document.write('<ol>');
while (keys.hasNext())
{
  var key = keys.next();
  document.write('<li>'+key+'='+XDSRM.lookupSpecialRegister(key,'???')+'</
li>');
}
document.write('</ol>');
```

The returned results would be an HTML document, as shown below:

```
<h2>Simple Ordered List of Special Registers:</h2><p>
<ol>
<li>iway.lastnode=Service</li>
<li>locale=en_us</li>
<li>iwayworkdir=C:/PROGRA~1/iway8/config/32CD7798534EECF2F41C303A0DF286
1</li>
<li>iwayconfig=32CD7798534EECF2F41C303A0DF28671</li>
<li>iway.pid=1592</li>
<li>name=ExecProcess</li>
<li>doclocation=config</li>
<li>val6=x=32CD7798534EECF2F41C303A0DF28671,
y=C:/PROGRA~1/iway8/config/8.0.0.SM</li>
<li>protocol=Lcl</li><li>iway.flowname=flowTest</li>
<li>engine=32CD7798534EECF2F41C303A0DF28671</li>
<li>iway.serverip=192.168.0.102</li><li>iwayversion=8.0.0.SM</li>
<li>iway.serverhost=informat-cde89b</li>
<li>iway.serverfullhost=192.168.0.102</li>
<li>iwayhome=C:/PROGRA~1/iWay8/</li>
<li>iway.channel=ExecProcess</li>
</ol>
```

Results will vary depending on your configuration.

The getSpecialRegisters method is used to obtain a HashMap of all SREGs. From the HashMap, the keySet method returns the key Set object, and the iterator method returns the iterator object for that Set. While there are key objects within the iterator, the key object is obtained from the iterator. You can use the lookupSpecialRegister method of the XDSRM object to look up the value associated with the key.

For more information, see the SREG description found in the *iWay Service Manager Programmer's Guide* for the methods and variables of the object.

XDMGR

The XDMGR object is available to the iSM Administration Console, as well as scripts invoked using the iWay Functional Language (iFL) interface. This object allows scripts to the iSM XDManager object allowing the JavaScript writer accesses to the iSM channels that are configured on the system. For example, the following JavaScript looks up the XDMaster mapping and iterates through the key set of the map to get the name and status of the Master:

```
importPackage(com.ibi.edaqm);
importPackage(java.util);
document.write('<h2>Simple Ordered List of Masters:</h2><p>');
var map = XDMGR.getMasters();
var keys = map.keySet().iterator();
// build an ordered list for display
document.write('<ol>');
while (keys.hasNext())
{
    var key = keys.next();
    var master = XDMGR.getMaster(key);
    document.write('<li>'+key+', state='+master.getStateName()+'</li>');
}
document.write('</ol>');
```

The returned results would be an HTML document, as shown below:

```
<h2>Simple Ordered List of Masters:</h2><p>
<ol>
<li>BAMChannel, state=inactive</li>
<li>FTPServerToDOCROOT, state=inactive</li>
<li>FILE2SFTP, state=inactive</li>
<li>SOAP1, state=inactive</li>
<li>IBILOCAL, state=inactive</li>
><li>IBIMVS, state=inactive</li>
</ol>'
```

Results vary depending on your configuration.

The getMasters method is used to obtain a HashMap of all XDMaster (Channel) objects. From the HashMap, the keySet method returns the key Set object, and the iterator method returns the iterator object for that Set. While there are key objects within the iterator, the key object is obtained from the iterator. You can use the getMaster method of the XDMGR object to look up the Master (Channel). The getStateName method is used to get the displayable state that is associated with that Master.

For more information on the XDManager and its uses, see the Javadoc document found in the sub directory .../etc/doc/javadoc/com/ibi/edaqm.

The accessed manager may be a special manager if used in a process flow test. Such a manager cannot supply information pertaining to the currently running system as a whole. Only the manager in control of the runtime system can provide such information as the state of all running channels.

The following is a list of rules and considerations:

1. Do not use a method unless it is documented in the JavaDoc.
2. The methods in these classes were developed for Java use. If needed, contact iWay Support for use of specific methods that may not be easily available.

3. Do not treat this as an opportunity to hack. The server can be placed into inappropriate states through injudicious use of non-reference methods.

Set

Sets a parameter.

❑ **set acl [<cfgname>/]<aclName> <list of roles> [-noverify|-verify] [-append|-replace|-remove]**

Security in iWay Service Manager (iSM) is managed by Access Control Lists (ACLs), which are described in the *iWay Service Manager Security Guide*. With ACLs, users are assigned roles that are associated with permissions. The *set acl* command sets roles for a specified ACL. The list of available ACLs can be seen by using the *show acl* command. The roles associated with the ACL are assigned to a user by values in an authentication realm. The following table lists and describes the switches that are supported by the *set acl* command:

Switch	Description
`-append`	Adds the role(s) to the ones already associated with the ACL. If a configuration is specified (for example, myconfig/cmdstart), then the configuration will be added if it does not exist in the security file.
`-replace`	Replace the existing roles with the new roles.
`-remove`	Removes the listed roles from the ACL.
`-verify`	Checks the integrity of the security file holding the roles. This is the default.
`-noverify`	Indicates not to check the integrity of the security file holding the roles.

Example:

In this example, the `cmdstart` and `cmdstop` permissions are added to the `operator` role in the `testsys` application.

```
set acl testsys/operator cmdstart cmdstop
```

The `operator` role is assigned to the user in the selected authentication realms, such as LDAP.

The configuration parameter can also be a regular expression if required. This enables the role to apply to a class of configurations based on a naming convention. For example, if development servers end with D, production servers with P, and quality assurance servers with Q, then a role might be assigned as follows:

```
set acl .Q/monitor  monitor -append
```

- **set active on | off**

  Sets the active status of the named channel/listener. An active channel starts automatically when the server is started, and starts in response to a start all command.

- **set appcmd name 'command' [parms]**

  Creates a *user* command. For more information, see *User Commands* on page 101.

- **set cmdstatus on | off**

  Causes the command handler to mark each subsequent command with a notice of success or failure. Commands that complete successfully are marked with IWAY0000 OK. Failures are marked with IWAY9999 Failed. The failing line is prefixed with [fail]. Users are cautioned that when using automated scripts, not to count on the exact 9999 number. That number may vary in the future. A value of 0000 will always denote success.

- **set display**

  Shows the current settings.

- **Set passive on | off -m name**

  Sets the passivation state of the named channel. A passivated channel remains started, but temporarily ceases to acquire messages until it is reactivated. This functionality is used to throttle performance. Only some protocols support passivation. Passive state is controlled internally by High Water Mark (HWM) detection in some listeners.

- **set pflowdebug on | off [-m <channelname>]**

  Debug nodes in process flows that normally execute only for process flows running in iIT tests if the Debug mode property is set for the process flow. During normal flow execution in a channel, the nodes are ignored. Using this command instructs the debug nodes to execute during a normal channel operation. This property can only be set through the command console and cannot be saved. Users are cautioned that enabling the process flow debug state can significantly impact system performance.

- **set policy [<cfgname>/]<policyName> <value> [-noverify]**

Sets the named policy to the required value. The -noverify flag avoids the check for signatures on the checking side, and can only be used with administrative authority.

- **set property <file> <name> <value> [-encrypt | -aes <key> [-keylength <length>]]**

  Sets the named property in the property file. The file name is expected to end with the *.properties* extension and this suffix will be supplied if required. The file will be created if it does not exist. The value is evaluated as iFL, which facilitates setting values that have been encrypted or taken from other portions of the system. The *-encrypt* keyword will encrypt the property value using an iSM salted masking algorithm.

  The *-aes* keyword uses the provided key to encrypt the value. Advanced Encryption Standard (AES) is a strong encryption standard. The *key* must be 16 characters or less, as it becomes a 128-bit key. The *keylength* for AES can be 128 (default), 192, or 256. To use key lengths greater than 128, the appropriate Java policy files must be available.

  For example:

  ```
  set property testproperty userpswd mypassword -aes iwaykey1
  ```

  If the key value is stored in a Special Register (SREG), which is the usual practice, named for the *secretkey* sample, the command would be structured as follows:

  ```
  set property testproperty userpswd mypassword -aes _sreg(secretkey)
  ```

  Either method would generate a properties file called *testproperty.properties*, which is structured as follows:

  ```
  #Saved by set property command
  #Fri Jan 24 14:52:29 EST 2014
  userpswd=A8vRBNezksAtoySgaFbOygkuMeYqmIy6v9GsIwU6K60\=
  ```

  The value can be read using standard iFL functions in the specific operand or command. For example:

  ```
  _aes('decrypt',_sreg(secretkey),_property('testproperty','userpswd'))
  ```

  This example generates *mypassword* for use in the process execution.

  For more information, see *Keeping Secret Values* in the *iWay Service Manager Security Guide*.

- **Set qa on|off**

  QA (Quality Assurance) mode is used during application testing to facilitate tests. For example, it causes timestamps in XML documents to be replaced with a constant to assist during message compares.

- **set register <name> [<value>][-switches] [-delete]**

  Sets a configuration register to a specified value. Configuration registers are those shown in the sregs command a Global manager. You cannot change the value of any of the built in registers, but you can add your own. The value of the register is ascertained when the register is accessed by the server; for example changing a value used to configure a channel will not affect that channel until it is restarted.

  Note: One register can depend on another. If the new register depends on the value of another register, that register must have been created before the dependent register. Also, you can start the iFL value with the defer character (`) to signify that the value is to be determined at the time of lookup. For typed registers, a constant value at set time is checked.

Value	Description
-secure	The value is to be encrypted. Use this for passwords.
-defer	The value is determined at lookup time. Use this for an iFL to be evaluated each time that the value is obtained.
-delete	Delete the register.
-string	Default. The register value is string.
-integer	The register value is an integer.
-float	The register value is a floating point number.
-double or -real	The register value is a double precision floating point number.

- **Set retryinterval duration [-save]**

  During channel startup, a failure (for example, MQ is unable to reach a queue manager) that prevents the channel from starting will cause the channel to be queued for retry in a set period. The duration is specified in seconds. If *-save* is used, then the configuration is permanently changed. The default duration is 30 seconds. The retry interval is a server property, and affects all channels. Channels that fail to start due to a configuration error will not be retried.

- **set time on|off**

  Adds the current time to all traces displayed at the console.

- **set <tracelevel> on|off [-save] [-master <name>]**

  Sets a trace level. The trace level change can be saved in the dictionary such that it becomes permanent. A setting can be applied to a single master/channel by name, but master-level settings cannot be saved.

  Note: The name of the console is "console". You can set trace levels for the console as `-master console`. By default, regardless of the specific trace level settings for other channels, during iSM startup, the console traces only error and warning level messages. You can change this behavior using the Console Settings page. For more information, see BAD XREF HERE "Console Settings.

- **set unique <pattern>[value]**

  Sets or resets the unique [file] name index. The unique file name pattern is used to generate unique names. For example, the following file name resets the value used for the pattern to zero:

  ```
  set unique c:/fileout/file###.xml
  ```

  ```
  set unique "c:/Program Files/config/base/output/ib####.txt" 50
  ```

  The set affects use of the pattern both for the unique file name generation (for example, in the file emitter) and in the _unq() iFL command. The value must fall within the modulus of the pattern size; for example, setting the value of the above pattern to 1000 is not accepted.

  This command might be scheduled to automatically reset the value at midnight each day (requires installation of the iSM scheduling component,).

  Note that the example with a pattern name enclosed the pattern in quotes. This is because it contained a blank in the pattern (file) name.

Shell

Also can be abbreviated ! per the common practice on some systems. The remainder of the command line is treated as an external command, and the server attempts to pass it to the operating system. The results of the attempt will depend upon the OS and the command.

Some operating systems do not support execution of internal commands such as *dir* or *copy*. Using the shell command, however, it is possible to use Windows internal commands through an invoked command shell. For example:

```
Enter command:>shell cmd /c dirVolume in drive C is OS
Volume Serial Number is C0FE-5175
Directory of C:\iway8\config\base
02/22/2013  03:01 PM    <DIR>          .
02/22/2013  03:01 PM    <DIR>          ..
10/24/2012  03:19 PM    <DIR>          asyncOutSOAP1
02/22/2013  03:01 PM    <DIR>          backups
02/22/2013  03:01 PM            75,600 base.xml
.
.
.
```

Show

The *show* command displays information about iWay Service Manager (iSM). This command supports minimum recognition to make the first operand easier to enter. It also supports subcommands to detail which information is to be shown.

The *show* command uses the following format:

```
show <option>
```

where:

<option>

Is a specific subcommand.

The *remote* command can be used to connect the command handler to other hosts (*iwayhome* instances on other computers) to investigate information on those systems.

The following sections describe the supported subcommands.

Classpath

Displays the members of the current classpath.

Console

Displays information about users who are logged on to the iWay Service Manager (iSM) Administration Console.

Configs

Shows information on reachable configurations on iWay Service Manager (iSM). For example:

```
* base          Port 9999  Server Uptime: 10 minutes      telnet port=23
Config1         [template]
inbound_config  Port 10004 Server is down
```

Descriptor

A descriptor is created when an application is deployed using iWay Integration Tools (iIT). It contains internal information, which is usually referenced by iWay Technical Support. The exact information that is returned by a descriptor command may vary between iWay releases.

You may be requested by iWay Technical Support to execute this command for troubleshooting or debugging purposes.

The *show descriptor* command uses the following format:

```
show descriptor app_name
```

where:

app_name

Is the name of the application that is deployed using iIT.

Examples:

```
show descriptor app1
```

```
show descriptor *
```

Note: An asterisk character (*) indicates the current configuration/application. Use with the remote command.

Errors

Display the prior ten error and warning messages.

The *show errors* command uses the following format:

```
show errors [-clear]
```

If *-clear* is used, then the table of messages is cleared.

Extensions

The *show extensions* command displays the extensions that are currently in use by iSM. In addition, this command displays any extensions that failed to load. Extensions are used add components such as protocols to iSM. This is a diagnostic command and may be requested by iWay support. For example:

```
Enter command>show extensions
iwrdbhwm
iwxalog
iwxaq
iwxbatch  (Not licensed)
iwxconfigservices
```

Exits

Displays information about loaded exits.

The *show exits* command now requires an exit type.

```
[-sort <sort type> -low <key> -high <key>] [-detail]
```

The following table lists and describes the supported exit types that can be called. Additional exit types may be supported in a future release.

Exit Type	Description
activity	Activity log drivers such as BAM or XDTimeLogger configured for use.
agents	Services (agents) available for configuration.
correlation	Correlation drivers configured for use.
listeners	Acquisition protocol handlers available for configuration.
persistent	Drivers such as activity log available for configuration.
preemitters	Preemitters available for configuration.
preparsers	Preparsers available for configuration.
replyTo	Reply protocol handlers available for configuration.
reviewers	Reviewers (called before parsers) available for configuration.

Note: If the detail switch is used, then additional information about the exit is displayed.

The following table lists and describes the supported sort types (ASCII order).

Sort Type	Description
name	The formal name of the component.

Sort Type	Description
description	The description (label) of the component as seen in configuration screens.

The following table lists and describes the supported sort filters (ASCII order).

Sort Filter	Description
low	The lowest sorted value (by sort type) to be included.
high	The highest sorted value (by sort type) to be included.

Options include, displaying information about services (agents), preparsers, listener protocols, reply protocols, correlation, xalog exits, and so on. These exits are classes and not instances. It is dependent upon what is available in the server, and not what has been used. The available exits depend on the services that are installed.

Flows

Displays information about flows that have executed under a channel (for example, the contents of the flow cache) and flows that are available for execution by the flow command. The *-sysonly* switch can be used to avoid seeing the channel flows, and to only have system executable flows presented.

iFL

Displays help for specified iFL functions.

The *show ifl* command uses the following format:

```
show ifl [function]
```

If the function name is omitted, then the list of available iFL functions is shown. Otherwise, information is provided for the specified function, where available.

iFLCache

As iFL is compiled for use, it is placed into a cache. This command shows the contents of the cache. Normally this command is used only by iWay Technical Services. You cannot affect the cache, as it is internal to iWay Service Manager (iSM).

The *show iflcache* command uses the following format:

```
show iflcache [maxitems] [filepath]
```

where:

maxitems

Indicates the maximum items to display.

filepath

Writes the contents of the display to the named file.

Jceproviders

Java cryptographic providers are used by iSM to provide security facilities. The loaded providers are used by the server in configuring available algorithms, and so on. For example:

```
show jceproviderJCE Providers available:
   SUN
   SunJSSE
   SunJCE
   SunMSCAPI
   BC
```

JVM

Displays the manufacturer and version of the JVM.

License

Displays information about the current license, including feature codes, identified license files, and so on.

Manifest

Displays the manifest for a specified .jar file. The default location is the current classpath.

Memory

Displays memory statistics. Interpretation of memory usage statistics are beyond the scope of this document. For display purposes, some of the repetitive portions of the examples have been omitted.

Memory alone provides a simple display.

```
memory max 65088K used 9336K, free 464K,
      nodes: total allocated 7441 namespace 3604
      Maps: funcs 0, xpath 0
      Heap: init 0K, used 8932K, commit 9336K, max 65088K
      Non-Heap: init 28864K, used 22852K, commit 32096K, max 118784K
      Garbage Collection Info
      Name: Copy   Collection count: 203   time: 454ms
      Name: MarkSweepCompact   Collection count: 0   time: 0ms
```

Memory detail yields a far more complete display.

```
memory max 65088K used 9336K, free 899K,
      nodes: total allocated 7441 namespace 3604
      Maps: funcs 0, xpath 0
      Heap: init 0K, used 8438K, commit 9336K, max 65088K
      Non-Heap: init 28864K, used 22860K, commit 32096K, max 118784K
      Garbage Collection Info
      Name: Copy   Collection count: 204   time: 455ms
         Memory Pools:
           Eden Space
           Survivor Space
      Name: MarkSweepCompact   Collection count: 0   time: 0ms
         Memory Pools:
           Eden Space
           Survivor Space
           Tenured Gen
           Perm Gen
           Perm Gen [shared-ro]
      Memory Pools Info
      Name: Code Cache
           Usage: init 192K, used 1613K, commit 1632K, max 32768K
           Peak Usage: init 192K, used 1613K, commit 1632K, max 32768K
           Type: Non-heap memory
           Memory Manager Names: CodeCacheManager
      Name: Eden Space
                -
                -
      Name: Perm Gen [shared-rw]
        Usage: init 12288K, used 5821K, commit 12288K, max 12288K
        Collection Usage: init 12288K, used 0K, commit 0K, max 12288K
        Peak Usage: init 12288K, used 5821K, commit 12288K, max 12288K
        Type: Non-heap memory
        Memory Manager Names: MarkSweepCompact
```

Java memory is under control of the system, and generally falls into four major categories.

Field	Definition
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. The value of init may be undefined (0) on some platforms.
used	Represents the amount of memory currently used.
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 (increase or decrease). The Java virtual machine may release memory to the system and committed could be less than init. Committed will always be greater than or equal to used.
max	Represents the maximum amount of memory (in bytes) that can be used for memory management. Its value may be undefined. The maximum amount of memory may change over time if defined. The amount of used and committed memory will always be less than or equal to max if max is defined. A memory allocation may fail if it attempts to increase the used memory such that used > committed even if used <= max would still be true (for example, when the system is low on virtual memory).

Packages

Packages are the primary method of adding components to iSM configurations, for example, iWay Integration Applications (iIAs) or applications. The *show packages* command displays packages and their contents for the iSM configuration, iIA, or application.

`show packages [name]`

where:

name

Is the name of the specific server (iSM instance) to be reported. If omitted, then the current server is assumed.

Policies

The *show policies* command shows the current security policies. For more information on security policies, see the *iWay Service Manager Security Guide*.

Pools

The server maintains resource pools for internal use. The *show pools* command lists the pools currently in use, along with some usage information.

Ports

The *show ports* command shows the currently assigned IP ports and the corresponding channel with which they are associated.

Providers

iWay Service Manager (iSM) is configured with resource providers, offering centralized configurable control over resources such as keystores, namespaces, LDAP directory access, and so on. The providers command lists which providers are currently being used. Providers that have not ever been accessed for any purposes are not shown, even if configured.

Queue

Displays the status of a managed queue when the information is available to iSM.

```
Masters:
ordered [Ordered]
  Name of Ordered Queue: ordered  [persistent]
    size: 78  added: 14390  removed: 14312  persisted: 14390  deleted: 0
```

The following table lists and describes the values that are returned.

Value Returned	Description
size	The number of messages currently on the queue.
added	The number of messages added to the queue.

Value Returned	Description
removed	The number of messages removed from the queue and passed to a process flow.
persisted	The number of messages written to the persistence queue. Note that if there were any messages on the persistence store when the listener was started, this value may not equal the number shown as added.
deleted	The number of messages in cancelled batches.

Registers

Special Registers (SREGs) are variables (name/value pairs) used throughout iWay Service Manager (iSM). Registers call into a hierarchy of definition, and come in several types. For example, each listener has its own SREGs, available to all message channels that it controls. A set of SREGs is made available to all users in iSM. The *show registers* command displays a list of available SREGs in iSM.

```
Special Registers:
    Global (manager):
        iwayhome=[SYS] 'c:/iway8/'
        engine=[SYS] 'base'
        iwayconfig=[SYS] 'base'
        doclocation=[SYS] 'config'
        iway.serverfullhost=[SYS] 'beck-2.ibi.com'
        console-master-port=[SYS] '9999'
        locale=[SYS] 'en_us'
        ibse-port=[CFG] '9000'
        iwayversion=[SYS] '8.0.SM'
        iwayworkdir=[SYS] 'c:/iway8/config/base'
        iway.serverip=[SYS] '172.19.20.239'
        iway.serverhost=[SYS] 'beck-2'
    SOAP1:
        protocol=[SYS] 'SOAP'
        name=[SYS] 'SOAP1'
    filein:
        protocol=[SYS] 'FILE'
        name=[SYS] 'filein'
```

Retryinterval

The *show retryinterval* command shows the current setting of the retry interval. The retry interval controls the time that iSM will delay before attempting to start a channel that is properly configured but was unable to start for some external reason, such as the inability to reach a designated remote address.

Routes

Routes are used by the channels to direct the processing of incoming messages. The routes currently defined for each channel are shown by this command.

Schedule

Currently defined schedules are displayed.

Sysprops

Java properties defined when starting the JVM are displayed. The classpath is not included, as it is available through the *show classpath* command. For example:

```
show sysprops IWAY61                                    ='c:/iway80'
IWAY8                                  ='c:/iway80'
IWAY80                                 ='c:/iway80'
asd                                    ='ddd'
awt.toolkit                            ='sun.awt.windows.WToolkit'
file.encoding                          ='Cp1252'
file.encoding.pkg                      ='sun.io'
.
.
.
java.vm.version                        ='24.45-b08'
javax.xml.parsers.SAXParserFactory
='org.apache.xerces.jaxp.SAXParserFactoryImpl'
javax.xml.stream.XMLInputFactory
='com.sun.xml.internal.stream.XMLInputFactoryImpl'
javax.xml.transform.TransformerFactory='net.sf.saxon.TransformerFactoryImpl'
line.separator                         ='\r\n'
os.arch                                ='amd64'
os.name                                ='Windows 7'
os.version                             ='6.1'
.
.
.
user.dir                               ='c:\iway80\config\base'
user.home                              ='C:\Users\Beck'
user.language                          ='en'
user.script                            =''
user.timezone                          ='America/New_York'
user.variant                           =''
Number of processors = 4
```

Sysvars

Displays the values of the Java system variables. You can restrict the display to one or a set by entering the “starting value” for the variables to be shown.

The following display shows all variables that begin with user:

```
show sysvars user
```

Threads

Displays thread information.

- **show threads [monitor | dump]**: Command given to display thread information.
- **show threads (no operands)**: Each channel is shown. Under the manager, there is one thread for each channel name [called a listener] (SOAP1 and filein). Each has a set of threads for its message channel. For example, the filein listener there are three message channels, called W.filein.1 and so forth. The state of each thread is shown.

```
SOAP1:
Thr-HSOAP1  [runnable]
Thr-FSOAP1  [timed_waiting]
filein:
W.filein.1  [timed_waiting]
W.filein.2  [timed_waiting]
W.filein.3  [timed_waiting]
manager:
console0  [blocked]
console1  [runnable]
console2  [blocked]
console3  [blocked]
console4  [blocked]
SOAP1  [timed_waiting]
    filein  [timed_waiting]
```

- **show thread monitor [on|off]** controls a check of deadlocked threads. Use of this command can significantly affect system performance. To monitor deadlocks, first issue the `thread monitor on` command. Once issued, the thread deadlock monitor is activated. Then issue thread monitor to check for deadlocks. When you are finished with the testing, set the monitor off.
- **show thread dump** issues a trace dump of every thread except the console thread. It is useful for tracking down problems, but is of limited use outside of iWay development.

Unique

Displays the values of the unique naming patterns currently in use. `Show unique` displays the following:

```
C:/fileout/test###.txt = 77
C:/outfiles/hippaout###.txt = 343
```

This means there are two unique naming patterns in use. The current setting value is shown above.

Note: Use set unique to reset values.

Version

Displays the iSM version. A version consists of the main service level version, such as 8.0.0 and a build number, such as 1191. The build number is useful information to provide iWay Software Customer Support when an issue is being resolved. This number is the lowest build number of all iWay .jar files on the classpath.

Below the main version display is listed any iWay .jar files in the Java class path that have a build number greater than the main build number or have a build number that cannot be determined. These .jar files usually contain upgrade software patches, along with their build number and some build information.

```
Version 8.0.0-SNAPSHOT.1191
iwcore.jar          1191   PLATO      Sun 06/11/2017 03:42 PM EDT
```

This display shows that the iSM version in use is 8.0.0, and that two changes have been applied. The first is a general upgrade for one component and the second is possibly a special patch made. The exact format of the build information may vary based on the machine name and time of build. Additional information on any of the .jar files is available through the *manifest* command.

The following table lists and describes the switches that are supported by the *version* command:

Switch	Description
`all`	Displays the version for all iWay .jar files, not only those with build version numbers greater than the minimum version number.
`debug`	Adds additional information, including the name of the .jar file that is identified as the base version .jar file.

Use the *version all* command to display the versions of all iWay .jar files without filtering the date.

Xalog

Information on currently defined activity log drivers is displayed. For example:

```
show xalogXALog Exits(1):Benchmarker: XDLogBase [anyRuntimeParm=false,
canCancel=false,
hasDomainError=false, name=xalog, runParms={events=false, skip=0,
delim=comma, file=c:\fileout\timer###.txt}, runtimeParmArray=[false, false,
false, false]]
```

Sleep

Pauses the command script for the designated time period. The time is in seconds, specified to a precision of hundredths.

```
Sleep 2   sleep for two seconds
Sleep 1.5 sleep for one half second
```

Spool

The spool holds a record of commands and their responses. By default, spooling is disabled.

- **spool <file> [append]**: Starts the spool. If the file exists, it is deleted and the spool is restarted. If the append token is used, the spool does not delete the file, but rather appends to it. As a protection, you cannot create a spool with suffix .xml, .properties, and so on. iWay suggests using file names that end in .txt, .log, .spool, and so on.
- **spool off**: Pauses spooling. You might want to do this to avoid recording repetitive commands.
- **spool on**: Resumes spooling.
- **spool close**: Terminates and closes the spool.
- **spool**: Displays the state of the spool.

The spool applies only to the immediate command connection. The spool command cannot be used in remote mode.

Absolute spool file locations are accepted as entered. Relative spool file locations are considered to be off the log directory, which is normally configured for the server in the Log Settings console page. The default is <home>/config/<configuration>/log.

Start

Starts one or more (named) channels. The start is asynchronous, which means that the start is triggered by the command, but the channel(s) cannot be expected to be active by the time the command returns.

The following list describes the formats.

- **start [all] [switches]**: Starts all active channels.
- **start <name> [-protocol]**: Starts one listener/channel or one protocol. The protocol name must be that of a valid protocol installed in the system.
- **start (<name> ,<name>*) [switches]**: Start one or more specific listeners/channels or protocols.

When a channel name is specified, you may add an asterisk character (*) to the end of the channel name to start all channels that begin with the specified name. In the following example, all channels that begin with *chan1* will be started.

```
start chan1*
```

The switches control how the names are interpreted. The following table lists and describes the switches that are supported by the *start* command:

Switch	Description
`-protocol`	The names are interpreted as protocol names, such as MSMQ or File. This allows starting of all listeners/channels of a specified protocol.
`-pulse`	Attempt to start a pulsable channel for a single access/poll cycle. Some protocols such as file can be pulsed. Others, such as HTTP, cannot be pulsed. An attempt to start a nonpulsable channel will result in an error. Pulsing can only be used for starting specific, named channels.
`-active`	The start applies to channels marked active. Active channels start when the start all is issued (by default at the initialization of the server). This is the default for the start command.
`-inactive`	The operation applies to inactive channels. Inactive channels do not start when the start all command is issued. This allows channels to be "reserved" outside of normal operation.
`-both`	The operation starts both active and inactive channels.
`-doflow`	Run the channel startup failure process flow if defined for the listener in the event that the channel does not properly start.

Switch	Description
`-noflow`	Do not run the channel startup failure process flow if defined for the listener. This is the default.

The following examples show usage of the *start* command:

```
start chan1
```

```
start mq -protocol -doflow
```

```
start (mq, file) -protocol -doflow
```

Stats

Reports statistics on the current activity of the server. The Ehrlang coefficient is shown as constant until enough information is available to compute a valid value other than 1.0. The number per second computation assumes a steady input, and as it is based on wall clock time, is for guidance only.

```
in secs     count      low     high     mean variance std.dev.   ehr  n/sec
SOAP1
        wall:     0   0.0000   0.0000   0.0000   0.0000   0.0000
-
        cpu :         0.0000   0.0000   0.0000   0.0000   0.0000
-
        user:         0.0000   0.0000   0.0000   0.0000   0.0000
-
filein
        wall:     3   0.0000   0.0160   0.0103   0.0001   0.0073 const
196.77
        cpu :         0.0000   0.0156   0.0052   0.0001   0.0074
const
        user:         0.0000   0.0156   0.0052   0.0001   0.0074 const
```

stats -reset: Sets the statistics counters to zero. If a channel name is specified, then only that channel is reset.

Stop

Stops one or more [named] listeners/channels.

- **stop [all]**: Stops all listeners/channels currently started.
- **stop <name> [-protocol]**: Stops one listener/channel or protocol.
- **stop (<name> [,<name>*) [-protocol]**: Stops one or more specific listeners/channels or protocols. The protocol name must be that of a valid protocol installed in the system.

When a channel name is specified, you may add an asterisk character (*) to the end of the channel name to stop all channels that begin with the specified name. In the following example, all channels that begin with *chan1* will be stopped.

```
stop chan1*
```

Time

Displays the current time on the command line. The time can be displayed in GMT or in local time:

Command	Description
Time	Displays the time in GMT.
time -gmt	Displays the time in GMT.
time -local	Displays the time in the local time of the server.

Tool

The server provides a setoff tool available to users for special purposes. Usually these are more complex than simple commands, and each tool can maintain its own dialog with the user. An example of such a tool is testfuncs, which enables a user to test an iFL expression against a specific document or set of special registers.

Each tool is document separately from this document.

Type

Displays the contents of a directory or a file.

To use *type*, issue the following command:

```
type <filename> [-hex] [-from <lowaddr>] [-to <highaddr>]
```

where:

<filename>

Is the absolute path or relative path to your current configuration. If a directory name is given, then the directory list is displayed. For example:

```
Enter command:>type c:/docs/a.xml

<sign> equals </sign>
```

The following table lists and describes the switches that are supported by the *type* command:

Switch	Description
-hex	Displays the file contents in hex. For example: **Enter command:>**type c:/docs/a.xml -hex 000000 3C 73 69 67 6E 3E 20 65 71 75 61 6C 73 20 3C 2F <sign> equals </sign> 000016 73 69 67 6E 3E 0D 0A
-from	Displays only *from* the line (non-hex) or byte offset (hex) value in decimal.
-to	Displays only *to* the line (non-hex) or byte offset(hex) value in decimal.

User Commands

An application command operates as shorthand for a more complex command. An application command can be created by:

```
set appcmd <name> <text> [<token=value>*]
```

where the <text> portion is issued as a command in response to the entry of the <name>. For example, to say "hello, world" when the user types "bigbang":

```
set appcmd bigbang 'say hello, world'
Enter command:>bigbang
hello, world
```

Note the quotes surrounding the command. This is required.

Appcmds are global to the server, and can be created in the startup script. Once created, they can be accessed from the main command window (if any) or from a Telnet command console.

Parameters can be included as token=value pairs. The value is included in the executed command within $name$ brackets.

```
set appcmd bigbang  'say hello, $where$'
```

To execute the command, enter the following:

```
Enter command:>bigbang where=mars
hello, mars
```

You can also add a default value, by including token=value pairs on the definition line:

```
set appcmd bigbang  'say hello, $where$'  where=world

Enter command:>bigbang where=mars
hello, mars
Enter command:>bigbang
hello, world
```

For example:

1. An application keeps counters for items read and items successfully indexed into a data base. The counters are in metric registers myapp.read and myapp.index. A command is desired to report the current counters.

   ```
   set appcmd myapp "say read: _sreg(myapp.read,0), indexed
   _sreg(myapp.indexed,0)"
   Enter command >myapp
   read: 15, indexed: 13
   ```

2. A user wants to encrypt passwords in AES, under the key stored in the pre-defined special register aeskey. The password needs to be surrounded in recognition tags, and it will be stored in the properties file pswds.properties. The command is shown on two lines in the example, but it must be on a single, continuous line.

   ```
   set appcmd enc  "set property c:/pswds $pswd$
    _concat('aes(',_aes('enc',sreg(aeskey),$pswd$),')')"
   Enter command:>enc pswd=dick
   The password properties file now contains the first entry
   #Saved by set command
   #Wed Jun 02 16:15:20 EDT 2010
   dick=aes(gM7ghnyG4DkLaJrs6R/jhQ\=\=)
   ```

Legal and Third-Party Notices