Oracle GlassFish Server
Reference Manual
Release 3.1.2
E24938-01

February 2012
This reference manual describes administration commands and utility commands that are available with Oracle GlassFish Server 3.1.2. This reference manual also describes concepts that are related to GlassFish Server administration.

The available options, arguments, and operands for each command are provided in accordance with standard rules of command syntax, along with availability attributes, diagnostic information, and cross-references to other manual pages and reference material with relevant information.

This reference manual is for all users of Oracle GlassFish Server.
Contents

Preface ......................................................................................................................... xv

1 Oracle GlassFish Server 3.1.2 Section 1: asadmin Utility Subcommands

add-library .................................................................................................................. 1-3
add-resources ............................................................................................................ 1-3
apply-http-lb-changes ............................................................................................. 1-3
backup-domain ......................................................................................................... 1-3
change-admin-password ............................................................................................ 1-3
change-master-broker ............................................................................................... 1-3
change-master-password ........................................................................................... 1-3
collect-log-files ....................................................................................................... 1-3
configure-jms-cluster .............................................................................................. 1-3
configure-lb-weight ................................................................................................ 1-3
configure-ldap-for-admin ...................................................................................... 1-3
copy-config .............................................................................................................. 1-3
create-admin-object ............................................................................................... 1-3
create-application-ref ............................................................................................ 1-3
create-audit-module ............................................................................................... 1-3
create-auth-realm ................................................................................................... 1-3
create-backup-config ............................................................................................. 1-3
create-cluster ......................................................................................................... 1-3
create-connector-connection-pool ......................................................................... 1-3
create-connector-resource .................................................................................... 1-3
create-connector-security-map ............................................................................. 1-3
create-connector-work-security-map ..................................................................... 1-3
create-custom-resource ........................................................................................ 1-3
create-domain ......................................................................................................... 1-3
create-file-user ....................................................................................................... 1-3
create-http ................................................................................................................ 1-3
create-http-health-checker ..................................................................................... 1-3
create-http-lb .......................................................................................................... 1-3
create-http-lb-config ............................................................................................. 1-3
create-http-lb-ref ................................................................................................... 1-3
create-http-redirect ............................................................................................... 1-3
create-iiop-listener ................................................................................................ 1-3
create-iiop-redirect ............................................................................................... 1-3
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>create-instance</td>
<td>1-3</td>
</tr>
<tr>
<td>create-jacc-provider</td>
<td>1-3</td>
</tr>
<tr>
<td>create-javamail-resource</td>
<td>1-3</td>
</tr>
<tr>
<td>create-jdbc-connection-pool</td>
<td>1-3</td>
</tr>
<tr>
<td>create-jdbc-resource</td>
<td>1-3</td>
</tr>
<tr>
<td>create-jmsdest</td>
<td>1-3</td>
</tr>
<tr>
<td>create-jms-host</td>
<td>1-3</td>
</tr>
<tr>
<td>create-jms-resource</td>
<td>1-3</td>
</tr>
<tr>
<td>create-jndi-resource</td>
<td>1-3</td>
</tr>
<tr>
<td>create-jvm-options</td>
<td>1-3</td>
</tr>
<tr>
<td>create-lifecycle-module</td>
<td>1-3</td>
</tr>
<tr>
<td>create-local-instance</td>
<td>1-3</td>
</tr>
<tr>
<td>create-message-security-provider</td>
<td>1-3</td>
</tr>
<tr>
<td>create-network-listener</td>
<td>1-3</td>
</tr>
<tr>
<td>create-node-config</td>
<td>1-3</td>
</tr>
<tr>
<td>create-node-dcom</td>
<td>1-3</td>
</tr>
<tr>
<td>create-node-ssh</td>
<td>1-3</td>
</tr>
<tr>
<td>create-password-alias</td>
<td>1-3</td>
</tr>
<tr>
<td>create-profiler</td>
<td>1-3</td>
</tr>
<tr>
<td>create-protocol</td>
<td>1-3</td>
</tr>
<tr>
<td>create-protocol-filter</td>
<td>1-3</td>
</tr>
<tr>
<td>create-protocol-finder</td>
<td>1-3</td>
</tr>
<tr>
<td>create-resource-adapter-config</td>
<td>1-3</td>
</tr>
<tr>
<td>create-resource-ref</td>
<td>1-3</td>
</tr>
<tr>
<td>create-schedule</td>
<td>1-3</td>
</tr>
<tr>
<td>create-service</td>
<td>1-3</td>
</tr>
<tr>
<td>create-ssl</td>
<td>1-3</td>
</tr>
<tr>
<td>create-system-properties</td>
<td>1-3</td>
</tr>
<tr>
<td>create-threadpool</td>
<td>1-3</td>
</tr>
<tr>
<td>create-transport</td>
<td>1-3</td>
</tr>
<tr>
<td>create-virtual-server</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-admin-object</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-application-ref</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-audit-module</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-auth.realm</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-backup-config</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-cluster</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-config</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-connector-connection-pool</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-connector-resource</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-connector-security-map</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-connector-work-security-map</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-custom-resource</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-domain</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-file-user</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-http</td>
<td>1-3</td>
</tr>
<tr>
<td>delete-http-health-checker</td>
<td>1-3</td>
</tr>
</tbody>
</table>
enable-backup-config ................................................................. 1-3
enable-http-lb-application ....................................................... 1-3
enable-http-lb-server .............................................................. 1-3
enable-monitoring ................................................................. 1-3
enable-secure-admin .............................................................. 1-3
enable-secure-admin-internal-user .......................................... 1-3
enable-secure-admin-principal ............................................... 1-3
export ..................................................................................... 1-3
export-http-lb-config .............................................................. 1-3
export-sync-bundle ............................................................... 1-3
flush-connection-pool ............................................................ 1-3
flush-jmsdest ........................................................................ 1-3
freeze-transaction-service ..................................................... 1-3
generate-jvm-report ............................................................... 1-3
get ....................................................................................... 1-3
get-client-stubs ................................................................. 1-3
get-health ........................................................................... 1-3
import-sync-bundle ............................................................... 1-3
install-node ........................................................................ 1-3
install-node-dcom ................................................................ 1-3
install-node-ssh .................................................................. 1-3
jms-ping .............................................................................. 1-3
list ....................................................................................... 1-3
list-admin-objects ............................................................... 1-3
list-application-refs ............................................................. 1-3
list-applications .................................................................. 1-3
list-audit-modules .............................................................. 1-3
list-auth-realms .................................................................. 1-3
list-backup-configs ............................................................... 1-3
list-backups ....................................................................... 1-3
list-clusters ........................................................................ 1-3
list-commands ................................................................... 1-3
list-components .................................................................. 1-3
list-configs ......................................................................... 1-3
list-connector-connection-pools ........................................... 1-3
list-connector-resources ..................................................... 1-3
list-connector-security-maps ............................................... 1-3
list-connector-work-security-maps ....................................... 1-3
list-containers .................................................................... 1-3
list-custom-resources .......................................................... 1-3
list-domains ....................................................................... 1-3
list-file-groups ................................................................... 1-3
list-file-users ...................................................................... 1-3
list-http-lb-configs ............................................................. 1-3
list-http-lbs ...................................................................... 1-3
list-http-listeners .............................................................. 1-3
list-iiop-listeners ............................................................... 1-3
ping-node-dcom ................................................................. 1-3
ping-node-ssh ................................................................. 1-3
recover-transactions ...................................................... 1-3
redeploy .......................................................... 1-3
remove-library ............................................................. 1-3
restart-domain .............................................................. 1-3
restart-instance ............................................................ 1-3
restart-local-instance .................................................. 1-3
restore-domain ............................................................ 1-3
resume-domain ............................................................. 1-3
rollback-transaction ..................................................... 1-3
rotate-log ................................................................. 1-3
run-script ................................................................. 1-3
set ................................................................. 1-3
set-log-attributes ......................................................... 1-3
set-log-levels ............................................................... 1-3
setup-ssh ................................................................. 1-3
set-web-context-param .................................................. 1-3
set-web-env-entry ........................................................ 1-3
show-component-status ................................................ 1-3
start-cluster ............................................................... 1-3
start-database .............................................................. 1-3
start-domain ............................................................... 1-3
start-instance .............................................................. 1-3
start-local-instance ..................................................... 1-3
stop-cluster ............................................................... 1-3
stop-database .............................................................. 1-3
stop-domain ............................................................... 1-3
stop-instance ............................................................. 1-3
stop-local-instance ...................................................... 1-3
suspend-domain .......................................................... 1-3
undeploy ................................................................. 1-3
unfreeze-transaction-service ......................................... 1-3
uninstall-node ............................................................. 1-3
uninstall-node-dcom .................................................... 1-3
uninstall-node-ssh ....................................................... 1-3
unset ................................................................. 1-3
unset-web-context-param ............................................. 1-3
unset-web-env-entry ................................................... 1-3
update-admin-server-coordinates ................................ 1-3
update-admin-server-local-coordinates .......................... 1-3
update-connector-security-map .................................... 1-3
update-connector-work-security-map .............................. 1-3
update-file-user .......................................................... 1-3
update-node-config ..................................................... 1-3
update-node-dcom ....................................................... 1-3
update-node-ssh .......................................................... 1-3
update-password-alias.................................................................................................................. 1-3
uptime ................................................................................................................................................. 1-3
validate-dcom ....................................................................................................................................... 1-3
validate-multicast .............................................................................................................................. 1-3
verify-domain-xml ............................................................................................................................... 1-3
version .................................................................................................................................................. 1-3

1 Oracle GlassFish Server 3.1.2 Section 1M: Utility Commands

appclient .............................................................................................................................................. 1-3
asadmin .................................................................................................................................................. 1-3
capture-schema ................................................................................................................................. 1-3
package-appclient ............................................................................................................................... 1-3

1 Oracle GlassFish Server 3.1.2 Section 5ASC: GlassFish Server Concepts

application ............................................................................................................................................ 1-3
configuration ....................................................................................................................................... 1-3
domain .................................................................................................................................................. 1-3
dotted-names ....................................................................................................................................... 1-3
instance ............................................................................................................................................... 1-3
logging .................................................................................................................................................. 1-3
monitoring .......................................................................................................................................... 1-3
passwords ............................................................................................................................................ 1-3
resource ............................................................................................................................................... 1-3
security ................................................................................................................................................ 1-3

1 Oracle GlassFish Server 3.1.2 Section 5GFP: Events

generic-probe ..................................................................................................................................... 1-3
glassfish:connector-pool:applications:connectionAcquiredEvent ....................................................... 1-3
glassfish:connector-pool:applications:connectionReleasedEvent ...................................................... 1-3
glassfish:connector-pool:applications:connectionUsedEvent ............................................................. 1-3
glassfish:connector-pool:applications:decrementConnectionUsedEvent .............................................. 1-3
glassfish:deployment:lifecycle:applicationDeployedEvent .................................................................... 1-3
glassfish:deployment:lifecycle:applicationUndeployedEvent ............................................................... 1-3
glassfish:ejb:bean:beanCreatedEvent ................................................................................................... 1-3
glassfish:ejb:bean:beanDestroyedEvent ............................................................................................... 1-3
glassfish:ejb:bean:containerEnteringEvent ........................................................................................... 1-3
glassfish:ejb:bean:containerLeavingEvent ............................................................................................ 1-3
glassfish:ejb:bean:messageDeliveredEvent .......................................................................................... 1-3
glassfish:ejb:bean:methodEndEvent ..................................................................................................... 1-3
glassfish:ejb:bean:methodReadyAddEvent ............................................................................................ 1-3
glassfish:ejb:bean:methodReadyRemoveEvent ..................................................................................... 1-3
glassfish:ejb:bean:methodStartEvent .................................................................................................. 1-3
glassfish:ejb:cache:beanPassivatedEvent .............................................................................................. 1-3
glassfish:ejb:cache:expiredSessionsRemovedEvent .............................................................................. 1-3
glassfish:ejb:pool:objectAddedEvent .................................................................................................... 1-3
glassfish:ejb:pool:objectAddFailedEvent ............................................................................................. 1-3
glassfish:jdbc:connection-pool:connectionTimedOutEvent ........................................ 1-3
glassfish:jdbc:connection-pool:connectionUsedEvent ........................................ 1-3
glassfish:jdbc:connection-pool:connectionValidationFailedEvent ...................... 1-3
glassfish:jdbc:connection-pool:decrementConnectionUsedEvent ....................... 1-3
glassfish:jdbc:connection-pool:incrementNumConnFreeEvent ......................... 1-3
glassfish:jdbc:connection-pool:incrementNumConnFreeEvent ......................... 1-3
glassfish:jdbc:connection-pool:potentialConnLeakEvent ................................ 1-3
glassfish:jdbc-pool:applications:connectionAcquiredEvent .......................... 1-3
glassfish:jdbc-pool:applications:connectionReleasedEvent .......................... 1-3
glassfish:jdbc-pool:applications:decrementConnectionUsedEvent .................. 1-3
glassfish:jdbcrasqltracing:traceSqlEvent ...................................................... 1-3
glassfish:jdbcrasqltracing:statementCache:statementCacheHitEvent ............... 1-3
glassfish:jdbcrasqltracing:statementCache:statementCacheMissEvent .............. 1-3
glassfish:jdbcrasqltracing:statementleak:potentialStatementLeakEvent .......... 1-3
glassfish:jersey:server:requestEnd ............................................................. 1-3
glassfish:jersey:server:requestStart .......................................................... 1-3
glassfish:jsf:faces-servlet:requestEnd ......................................................... 1-3
glassfish:jsf:faces-servlet:requestStart ....................................................... 1-3
glassfish:kernel:connection-queue:connectionAcceptedEvent ....................... 1-3
glassfish:kernel:connection-queue:connectionClosedEvent .......................... 1-3
glassfish:kernel:connection-queue:connectionConnectedEvent ..................... 1-3
glassfish:kernel:connection-queue:onTaskDequeuedEvent ............................ 1-3
glassfish:kernel:connection-queue:onTaskQueueOverflowEvent .................... 1-3
glassfish:kernel:connection-queue:setMaxTaskQueueSizeEvent ..................... 1-3
glassfish:kernel:connections-keep-alive:decrementCountConnectionsEvent ....... 1-3
glassfish:kernel:connections-keep-alive:incrementCountConnectionsEvent ........ 1-3
glassfish:kernel:connections-keep-alive:incrementCountHitsEvent ................ 1-3
glassfish:kernel:connections-keep-alive:setMaxCountRequestsEvent ............... 1-3
glassfish:kernel:connections-keep-alive:setTimeoutInSecondsEvent ............... 1-3
glassfish:kernel:connections-keep-alive:subHeapSizeEvent ......................... 1-3
glassfish:kernel:file-cache:addHeapSizeEvent .............................................. 1-3
glassfish:kernel:file-cache:addMappedMemorySizeEvent ................................ 1-3
glassfish:kernel:file-cache:countContentHitEvent ....................................... 1-3
glassfish:kernel:file-cache:countContentMissEvent .................................... 1-3
glassfish:kernel:file-cache:countHitEvent ................................................ 1-3
glassfish:kernel:file-cache:countInfoHitEvent ........................................... 1-3
glassfish:kernel:file-cache:countInfoMissEvent ......................................... 1-3
glassfish:kernel:file-cache:incOpenCacheEntriesEvent .................................. 1-3
glassfish:kernel:file-cache:subHeapSizeEvent .......................................... 1-3
glassfish:kernel:file-cache:subMappedMemorySizeEvent ............................... 1-3
glassfish:kernel:thread-pool:maxNumberOfThreadsReachedEvent ........................................ 1-3
glassfish:kernel:thread-pool:setCoreThreadsEvent ........................................................... 1-3
glassfish:kernel:thread-pool:setMaxThreadsEvent ............................................................ 1-3
glassfish:kernel:thread-pool:threadAllocatedEvent .............................................................. 1-3
glassfish:kernel:thread-pool:threadDispatchedFromPoolEvent .......................................... 1-3
glassfish:kernel:thread-pool:threadReleasedEvent ............................................................... 1-3
glassfish:kernel:thread-pool:threadReturnedToPoolEvent .................................................... 1-3
glassfish:orb:inboundconnection:inboundConnectionClosed ............................................... 1-3
glassfish:orb:inboundconnection:inboundConnectionOpened .............................................. 1-3
glassfish:orb:outboundconnection:outboundConnectionClosed ............................................ 1-3
glassfish:orb:outboundconnection:outboundConnectionOpened .......................................... 1-3
glassfish:security:ejb:policyDestructionEvent ................................................................. 1-3
glassfish:security:ejb:policy:policyCreationEvent ............................................................ 1-3
glassfish:security:ejb:securityManagerCreationEvent ....................................................... 1-3
glassfish:security:ejb:securityManagerDestructionEvent .................................................... 1-3
glassfish:security:realm:realmAddedEvent ........................................................................ 1-3
glassfish:security:realm:realmRemovedEvent .................................................................... 1-3
glassfish:security:web:policyConfigurationCreationEvent ............................................... 1-3
glassfish:security:web:policyConfigurationDestructionEvent .......................................... 1-3
glassfish:security:web:securityManagerCreationEvent ....................................................... 1-3
glassfish:security:web:securityManagerDestructionEvent .................................................... 1-3
glassfish:transaction:transaction-service:activated .............................................................. 1-3
glassfish:transaction:transaction-service:committed ............................................................ 1-3
glassfish:transaction:transaction-service:deactivated ......................................................... 1-3
glassfish:transaction:transaction-service:freeze .................................................................. 1-3
glassfish:transaction:transaction-service:rolledback .......................................................... 1-3
glassfish:web:http-service:requestEndEvent ....................................................................... 1-3
glassfish:web:jsp:jspDestroyedEvent .................................................................................. 1-3
glassfish:web:jsp:jspErrorEvent ........................................................................................ 1-3
glassfish:web:jsp:jspLoadedEvent ...................................................................................... 1-3
glassfish:web:jsp:jspReloadedEvent ..................................................................................... 1-3
glassfish:webservices:servlet-109:endedEvent .................................................................... 1-3
glassfish:webservices:servlet-109:startedEvent .................................................................. 1-3
glassfish:webservices:servlet-ri:endedEvent ...................................................................... 1-3
glassfish:webservices:servlet-ri:startedEvent ................................................................... 1-3
glassfish:web:session:afterServiceEvent ........................................................................... 1-3
glassfish:web:session:beforeServiceEvent ........................................................................ 1-3
glassfish:web:session:sessionDestroyedEvent .................................................................... 1-3
glassfish:web:session:sessionExpiredEvent ...................................................................... 1-3
glassfish:web:session:sessionPassivatedEndEvent ............................................................... 1-3
glassfish:web:session:sessionPassivatedStartEvent ............................................................. 1-3
glassfish:web:session:sessionPersistedEndEvent

glassfish:web:session:sessionPersistedStartEvent

glassfish:web:session:sessionRejectedEvent

glassfish:web:web-module:webModuleStartedEvent

glassfish:web:web-module:webModuleStoppedEvent
Preface

Both novice users and those familiar with Oracle GlassFish Server can use online man pages to obtain information about the product and its features. A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

Overview

The following contains a brief description of each man page section and the information it references:

- Section 1 describes, in alphabetical order, the asadmin utility subcommands.
- Section 1M describes GlassFish Server utility commands.
- Section 5ASC describes concepts that are related to GlassFish Server administration.
- Section 5GFP describes events that provide monitoring data for GlassFish Server.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no Bugs section.

Name
This section gives the names of the commands or functions documented, followed by a brief description of what they do.

Synopsis
This section shows the syntax of commands or functions.

The following special characters are used in this section:

[ ]
Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.

| Separator. Only one of the arguments separated by this character can be specified at a time.

Description
This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss options or cite examples.
Options
This section lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the Synopsis section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.

Operands
This section lists the command operands and describes how they affect the actions of the command.

Examples
This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command-line entry and machine response is shown. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the Synopsis, Description, Options, and Usage sections.

Exit Status
This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion, and values other than zero for various error conditions.

See Also
This section lists references to other man pages, in-house documentation, and outside publications.

Notes
This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here.

Bugs
This section describes known bugs and, wherever possible, suggests workarounds.

Documentation Accessibility
For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

Access to Oracle Support
Oracle customers have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.
This section describes, in alphabetical order, the subcommands of the `asadmin(1M)` utility.
add-library

adds one or more library JAR files to GlassFish Server

Synopsis

add-library [--help] [--type={common|ext|app}] [--upload={false|true}] library-file-path [library-file-path ... ]

Description

The add-library subcommand adds one or more library archive files to GlassFish Server.

The --type option specifies the library type and the GlassFish Server directory to which the library is added.

The library-file-path operand is the path to the JAR file that contains the library to be added. To specify multiple libraries, specify multiple paths separated by spaces.

Note: The library archive file is added to the DAS. For common and extension libraries, you must restart the DAS so the libraries are picked up by the server runtime. To add the libraries to other server instances, synchronize the instances with the DAS by restarting them.

This subcommand is supported in remote mode only.

Options

--help
-?

Displays the help text for the subcommand.

--type

Specifies the library type and the GlassFish Server directory to which the library is added. Valid values are as follows:

common

Adds the library files to the Common class loader directory, domain-dir/lib. This is the default.

ext

Adds the library files to the Java optional package directory, domain-dir/lib/ext.

app

Adds the library files to the application-specific class loader directory, domain-dir/lib/applibs.

For more information about these directories, see "Class Loaders" in Oracle GlassFish Server Application Development Guide.
--upload
Specifies whether the subcommand uploads the file to the DAS. In most situations, this option can be omitted.

Valid values are as follows:

false
The subcommand does not upload the file and attempts to access the file through the specified file name. If the DAS cannot access the file, the subcommand fails.

For example, the DAS might be running as a different user than the administration user and does not have read access to the file. In this situation, the subcommand fails if the --upload option is false.

true
The subcommand uploads the file to the DAS over the network connection.

The default value depends on whether the DAS is on the host where the subcommand is run or is on a remote host.

- If the DAS is on the host where the subcommand is run, the default is false.
- If the DAS is on a remote host, the default is true.

If a directory filepath is specified, this option is ignored.

Operands

library-file-path
The paths to the archive files that contain the libraries that are to be added. You can specify an absolute path or a relative path.

If the --upload option is set to true, this is the path to the file on the local client machine. If the --upload option is set to false, this is the path to the file on the server machine.

Examples

Example 1 Adding Libraries
This example adds the library in the archive file mylib.jar to the application-specific class loader directory on the default server instance.

asadmin> add-library --type app /tmp/mylib.jar
Command add-library executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

remove-library(1), list-libraries(1)
asadmin(1M)
"Class Loaders" in Oracle GlassFish Server Application Development Guide
add-resources

creates the resources specified in an XML file

Synopsis

add-resources [--help] [--target target] [--upload={false|true}] xml-file-name

Description

The add-resources subcommand creates the resources named in the specified XML file. The resources that can be created with this subcommand are listed in See Also in this help page.

The --target option specifies the target for which you are creating the resources. If this option specifies the domain, the resources are added only to the configuration of the domain administration server (DAS). If this option specifies any other target, the resources are added to the configuration of the DAS and references are added to the resources from the specified target.

The xml-file-name operand is the path to the XML file that contains the resources to be created. The DOCTYPE must be specified as http://glassfish.org/dtds/glassfish-resources_1_5.dtd in the resources.xml file.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target for which you are creating the resources.
Valid values are as follows:

server
Creates the resources for the default server instance server and is the default value.

domain
Creates the resources for the domain.

cluster-name
Creates the resources for every server instance in the cluster.

instance-name
Creates the resources for a particular GlassFish Server instance.

--upload
Specifies whether the subcommand uploads the file to the DAS. In most situations, this option can be omitted.
Valid values are as follows:

**false**
The subcommand does not upload the file and attempts to access the file through the specified file name. If the DAS cannot access the file, the subcommand fails. For example, the DAS might be running as a different user than the administration user and does not have read access to the file. In this situation, the subcommand fails if the `--upload` option is false.

**true**
The subcommand uploads the file to the DAS over the network connection. The default value depends on whether the DAS is on the host where the subcommand is run or is on a remote host.

- If the DAS is on the host where the subcommand is run, the default is false.
- If the DAS is on a remote host, the default is true.

### Operands

**xml-file-name**
The path to the XML file that contains the resources that are to be created. You can specify an absolute path, only the file name, or a relative path.

- If you specify an absolute path, the XML file can be anywhere.
- If you specify only the file name, the XML file must reside in the `domain-dir/config` directory on the DAS host. This requirement must be met even if you run the subcommand from another host.
- If you specify a relative path, the XML file must be in the relative directory.

An example XML file follows.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC
"-//GlassFish.org//DTD GlassFish Application Server 3.1 Resource Definitions//EN"
'http://glassfish.org/dtds/glassfish-resources_1_5.dtd'>
<resources>
  <jdbc-connection-pool name="SPECjPool" steady-pool-size="100"
    max-pool-size="150" max-wait-time-in-millis="60000"
    pool-resize-quantity="2" idle-timeout-in-seconds="300"
    is-isolation-level-guaranteed="true"
    is-connection-validation-required="false"
    connection-validation-method="auto-commit"
    fail-all-connections="false"
    datasource-classname="oracle.jdbc.pool.OracleDataSource">
    <property name="URL" value="jdbc:oracle:thin:@iasperfsol12:1521:specdb"/>
    <property name="User" value="spec"/>
    <property name="Password" value="spec"/>
    <property name="MaxStatements" value="200"/>
    <property name="ImplicitCachingEnabled" value="true"/>
  </jdbc-connection-pool>
  <jdbc-resource enabled="true" pool-name="SPECjPool"
    jndi-name="#jdbc/SPECjDB"/>
</resources>
```
Examples

Example 1 Adding Resources
This example creates resources using the contents of the XML file resource.xml.

```
asadmin> add-resources resource.xml
Command : Connector connection pool jms/testQFactoryPool created.
Command : Administered object jms/testQ created.
Command : Connector resource jms/testQFactory created.
Command : Resource adapter config myResAdapterConfig created successfully
Command : JDBC connection pool DerbyPoolA created successfully.
Command : JDBC resource jdbc/__defaultA created successfully.
Command add-resources executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jdbc-connection-pool(1), create-jdbc-resource(1),
create-jms-resource(1), create-jndi-resource(1), create-javamail-resource(1),
create-custom-resource(1), create-connector-resource(1),
create-connector-work-security-map(1), create-admin-object(1),
create-resource-adapter-config(1)
asadmin(1M)
apply-http-lb-changes

applies load balancer configuration changes to the load balancer

Synopsis

apply-http-lb-changes [--help] 
  lb-name

Description

Use the apply-http-lb-changes subcommand to apply the changes in the load
balancer configuration to the physical load balancer. The load balancer must already
exist. To create a physical load balancer, use the create-http-lb subcommand.

This subcommand is only applicable to Oracle GlassFish Server. This subcommand is
not applicable to GlassFish Server Open Source Edition.

Context

The Load Balancer distributes the workload among multiple Oracle GlassFish Server
instances, increasing the overall throughput of the system. The Load Balancer also
enables requests to failover from one server instance to another. For HTTP session
information to persist, configure HTTP session persistence.

Note: The Load Balancer Plugin is only available with Oracle
GlassFish Server, and is not available with GlassFish Server Open
Source Edition. For GlassFish Server Open Source Edition, it is
possible to use the mod_jk module to configure load balancing on
the Apache HTTP server.

For more information about configuring load balancing with GlassFish Server, refer to
the online help in the GlassFish Server Administration Console.

Options

--help
-?
Displays the help text for the subcommand.

Operands

lb-name
The name of the load balancer to which changes are applied. The load balancer must
already exist. You can create it with the create-http-lb subcommand.

Examples

Example 1 Using the apply-http-lb-changes subcommand
This example applies configuration changes to a load balancer named mylb.
asadmin> apply-http-lb-changes mylb
Command apply-http-lb-changes executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-http-lb(1), create-http-lb-config(1)
asadmin(1M)
backup-domain

performs a backup on the domain

Synopsis

backup-domain [--help]
[--long={false|true}]
[--description description-text]
[--domaindir domain-root-dir]
[--backupdir backup-directory]
[--backupconfig backup-config-name]
[domain_name]

Description

The backup-domain subcommand backs up files under the named domain.

This subcommand is supported in local mode only in GlassFish Server Open Source
Edition, and is supported in local mode and remote mode in Oracle GlassFish Server.

In GlassFish Server Open Source Edition, the domain to be backed up must be
stopped.

In Oracle GlassFish Server, the domain to be backed up must be stopped or be
suspended using the suspend-domain(1) subcommand.

Options

--help
-?
Displays the help text for the subcommand.

--long
-l
Displays detailed information about the backup operation.

The default value is false.

--description
Specifies a description to store in the backup file. The description is displayed as part
of the information about a backup file.

The default value has this form:

domain-name backup created on YYYY_MM_DD by user user-name

--domaindir
Specifies the domain root directory, the parent directory of the domain to back up.

The default value is as-install/domains.

--backupdir
Specifies the directory under which the backup file is to be stored.
The default value is `as-install/domains/domain-dir/backups`. If the domain is not in the default location, the location is `domain-dir/backups`.

`--backupconfig`
(Supported only in Oracle GlassFish Server.) The name of the domain backup configuration in the backup directory under which the backup file is to be stored.

**Operands**

`domain-name`
Specifies the name of the domain to be backed up.
This operand is optional if only one domain exists in the GlassFish Server installation.

**Exit Status**

0
subcommand executed successfully
1
error in executing the subcommand

**See Also**

`restore-domain(1), list-backups(1), suspend-domain(1), resume-domain(1)`
change-admin-password

changes the administrator password

Synopsis

change-admin-password [--help]
[--domaindir domain-root-dir [--domain_name domain-name]]

Description

The change-admin-password subcommand modifies the administrator password. The change-admin-password subcommand is interactive because the subcommand prompts the user for the old administrator password, for the new administrator password, and for confirmation of the new administrator password. The new password must contain at least 8 characters.

If the only user is an anonymous user without a password, this subcommand fails.

If a blank password is provided, this subcommand fails if secure administration is enabled.

For security purposes, create a password-protected user account with administrator privileges. To create this account, use the create-file-user(1) or the Administration Console. After creating this user account, remove the anonymous user to restrict access to GlassFish Server settings.

If more than one administrator is configured for GlassFish Server, you must run the asadmin command with the --user option to change the password for that user. For more information, see the examples in this help page.

This subcommand is supported in local mode and remote mode. If the --domaindir or --domain_name option is specified, the change-admin-password subcommand operates in local mode. If neither option is specified, the change-admin-password subcommand first attempts to operate in remote mode. If neither option is specified and the DAS is not running, the change-admin-password subcommand operates in local mode, using the default values for the --domaindir and --domain_name options.

Options

--help
-?
Displays the help text for the subcommand.

--domaindir

Specifies the parent directory of the domain specified in the --domain_name option. When this option is used, the change-admin-password subcommand operates in local mode.

--domain_name

Specifies the domain of the admin user.

This option is not required if the directory specified by the --domaindir option contains only one domain.
Examples

Example 1  Changing the Administrator Password For a Single User in Multimode
asadmin --user admin
asadmin> change-admin-password
Please enter the old admin password>
Please enter the new admin password>
Please enter the new admin password again>
Command change-admin-password executed successfully.

Example 2  Changing the Administrator Password For a Single User in Single Mode
asadmin --user admin change-admin-password
Please enter the old admin password>
Please enter the new admin password>
Please enter the new admin password again>
Command change-admin-password executed successfully.

Exit Status

0
command executed successfully

1
command failed

See Also

create-file-user(1), delete-password-alias(1), list-password-aliases(1),
update-password-alias(1)
asadmin(1M)
change-master-broker

changes the master broker in a Message Queue cluster providing JMS services for a GlassFish Server cluster.

Synopsis

change-master-broker [--help] clustered-instance-name

Description

The change-master-broker subcommand changes the master broker in a Message Queue cluster that is the JMS provider for a GlassFish Server cluster. By default, the master broker is the one associated with the first instance configured in the GlassFish Server cluster.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

--help
-?
Displays the help text for the subcommand.

Operands

clustered-instance-name
The name of the server instance whose Message Queue broker is to become the master broker of the Message Queue cluster. This server instance must be an instance in a GlassFish Server cluster.

Examples

Example 1 Changing the master broker
The following subcommand changes the Message Queue master broker to the one for the clustinst3 clustered instance.

asadmin> change-master-broker clustinst3
Command change-master-broker executed successfully.

Exit Status

0
subcommand executed successfully
1
error in executing the subcommand
See Also

asadmin(1M)
change-master-password

changes the master password

Synopsis

change-master-password [--help] [--nodedir node-dir] [--domaindir domain-dir] [--savemasterpassword=(false|true)] [domain-name|node-name]

Description

The change-master-password subcommand is used to modify the master password. The change-master-password subcommand is interactive in that the user is prompted for the old master password, as well as the new master password. This subcommand will not work unless the server is stopped. In a distributed environment, this command must run on each machine in the domain.

Options

--help
-?
Displays the help text for the subcommand.

--nodedir
The name of the directory containing the node instance for which the password will be changed. If this option is omitted, the change is applied to the entire domain.

--domaindir
The name of the domain directory used for this operation. By default, the --domaindir option is $AS_DEF_DOMAINS_PATH, which is an environment variable defined in the file asenv.bat or asenv.conf.

--savemasterpassword
This option indicates whether the master password should be written to the file system. This is necessary so that the start-domain(1) command can start the server without having to prompt the user.

The default is false.

Caution:  Saving the master password on disk is extremely insecure and should be avoided.

Note: If the --savemasterpassword option is not set, the master password file, if it exists, will be deleted.

Operands

domain-name|node-name
This name of the domain or node for which the password will be changed. If there is only a single domain, this is optional.
Examples

Example 1 Changing the Master Password
This example shows how to changed the master password for the domain44ps domain.

asadmin> change-master-password domain44ps
Please enter the new master password>
Please enter the new master password again>
Master password changed for domain44ps

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-password-alias(1), list-password-aliases(1), start-domain(1),
update-password-alias(1)

asadmin(1M)
collect-log-files

creates a ZIP archive of all available log files

Synopsis

collect-log-files [--help] [--target target] [--retrieve={false|true}] [retrievefilepath]

Description

The `collect-log-files` subcommand collects all available log files for the domain administration server (DAS), the specified cluster, or the specified GlassFish Server instance and creates a single ZIP archive of the log files. If a cluster is specified, the ZIP archive also contains the log file for the DAS.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--retrieve
Specifies whether the ZIP archive is created in a directory other than the default directory.

By default the ZIP archive is created in the `domain-dir/collected-logs` directory. The ZIP file names are constructed from the specified `target` and timestamp, as follows:

log_yyyy-mm-dd_hh-min-sec.zip

Possible values are as follows:

false
The ZIP archive will be created in the default directory. If omitted, the --retrieve option defaults to false.

true
The ZIP archive will be created in the directory that the `retrievefilepath` operand specifies. If `retrievefilepath` is omitted, the ZIP archive will be created in the default directory.

--target
Specifies the target for which log files will be collected.

Possible values are as follows:

server
The log files will be collected for the DAS (default).

instance-name
The log files will be collected for the specified GlassFish Server instance.
cluster-name
The log files will be collected for the specified cluster and the DAS.

Operands

retrievelfilepath
The name of the directory in which the ZIP archive will be created. If this operand is omitted, the ZIP archive will be created in the default directory. If the --retrieve option is false, this operand is ignored.

Examples

Example 1 Collecting Log Files for the Default Server
This example generates a ZIP archive from the log files for the default server.

```
asadmin> collect-log-files
Command collect-log-files executed successfully.
```

Example 2 Collecting Log Files for a Cluster
This example generates a ZIP archive from the log files for a cluster named cluster1 and the two server instances running in the cluster.

```
asadmin> collect-log-files --target cluster1
Log files are downloaded for instance1.
Log files are downloaded for instance2.
Command collect-log-files executed successfully.
```

Example 3 Collecting Log Files in a Directory Other Than the Default for a Cluster
This example generates a ZIP archive from the log files for a cluster named cluster1 and its two server instances, and saves the archive in a directory named /space/output.

```
asadmin> collect-log-files --target cluster1 --retrieve true /space/output
Log files are downloaded for instance1.
Log files are downloaded for instance2.
Created Zip file under /space/output/log_2010-12-15_15-55-54.zip.
Command collect-log-files executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand
See Also

list-log-attributes(1), list-log-levels(1), rotate-log(1), set-log-attributes(1),
set-log-levels(1)
asadmin(1M)

"Administering the Logging Service" in Oracle GlassFish Server Administration Guide
configure-jms-cluster

configures the Message Queue cluster providing JMS services to a GlassFish Server cluster

Synopsis

configure-jms-cluster [--help]
[-c clustertype={conventional|enhanced}]
[-c configstoretype={masterbroker|shareddb}]
[-c messagestoretype={file|jdbc}]
[-d dbvendor database-vendor]
[-d dbuser database-user]
[-d dburl database-url]
[-p property property-list]
cluster-name

Description

The configure-jms-cluster configures the Message Queue cluster providing JMS services to a GlassFish Server cluster.

This subcommand should be used before the GlassFish Server cluster is started for the first time. Otherwise, follow the instructions in "Administering the Java Message Service (JMS)" in Oracle GlassFish Server Administration Guide.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

--help
-?
Displays the help text for the subcommand.

--clustertype
The type of Message Queue cluster to configure. The value conventional specifies a conventional cluster, and the value enhanced specifies an enhanced, high-availability cluster. For information about these cluster types of Message Queue clusters, see "Broker Clusters" in Oracle GlassFish Server Message Queue Technical Overview.

The default value is conventional.

If enhanced is specified, the configstoretype and messagestoretype options are ignored.

--configstoretype
The type of data store for configuration data in a conventional cluster. The value masterbroker specifies the use of a master broker to store and manage the configuration data. The value shareddb specifies the use of a shared database to store the configuration data.

The default value is masterbroker.

This option is ignored if clustertype is set to enhanced.
--messagestoretype

The type of data store for message data in brokers in a conventional cluster. The value file specifies a file store. The value jdbc specifies a JDBC store.

The default value is file.

This option is ignored if clustertype is set to enhanced.

--dbvendor
--dbuser
--dburl

The database vendor, user, and access url of the JDBC database to use in any of these situations:
- When clustertype is set to enhanced
- When configstoretype is set to sharedb
- When messagestoretype is set to jdbc

For information about supported vendors and the formats of access urls for each vendor, see "JDBC-Based Persistence" in Oracle GlassFish Server Message Queue Administration Guide.

---

**Note:** To specify the password of the JDBC database user, use the --passwordfile utility option of the asadmin(1M) command after adding the entry AS_ADMIN_JMSDBPASSWORD to the password file.

---

--property

A list of additional database-vendor-specific properties to configure the JDBC database for use by the Message Queue cluster. Specify properties as a colon (:) separated list of property names and values in the form:

```
prop1name=prop1value:prop2name=prop2value
```

Operands

**cluster-name**

The name of the GlassFish Server cluster for which the Message Queue cluster is to provide JMS services.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

asadmin(1M)
configure-lb-weight

sets load balancing weights for clustered instances

Synopsis

configure-lb-weight [--help] --cluster cluster_name
        instance-name=weight[:instance-name=weight]

Description

The configure-lb-weight subcommand assigns weight to the server instances in a
cluster. Weights can be used for HTTP, RMI/IIOP and JMS load balancing. For the
HTTP load balancer, the weights are used only if the load balancer's policy is set to
weighted-round-robin. The load balancer policy is set in the create-http-lb-ref
subcommand or set subcommand.

Use the weight to vary the load going to different instances in the cluster. For example,
if an instance is on a machine with more capacity, give it a higher weight so that more
requests are sent to that instance by the load balancer. The default weight is 100. If all
instances have the default weight, the load balancer performs simple round robin load
balancing.

Note: This subcommand is only applicable to Oracle GlassFish
Server. This subcommand is not applicable to GlassFish Server

Options

--help
-?
Displays the help text for the subcommand.

--cluster
The name of the cluster.

Operands

instance-name=weight
The name of the instance and the weight you are assigning it. The weight must be an
integer. The pairs of instances and weights are separated by colons. For example
instance1=1:instance2=4 means that for every five requests, one goes to instance1
and four go to instance2. A weight of 1 is the default.

Examples

Example 1 Assigning Load Balancer Weights to Cluster Instances
The following subcommand assigns weights of 1, 1, and 2 to instances i1, i2, and i3 in
the cluster1 cluster.

asadmin> configure-lb-weight --cluster cluster1 i1=i1:i2=i1:i3=2
Command configure-lb-weight executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-http-lb-ref(1) create-cluster(1)
asadmin(1M)
configure-ldap-for-admin

configures the authentication realm named admin-realm for the given LDAP

Synopsis

configure-ldap-for-admin [--help]

Description

The configure-ldap-for-admin subcommand configures the authentication realm named admin-realm for the given LDAP. The configure-ldap-for-admin subcommand is interactive. The subcommand prompts the user for the basedn and ldap-group options.

This command is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Examples

Example 1 Configuring the LDAP Authentication Realm

asadmin> configure-ldap-for-admin
Enter the value for the basedn option>
Enter the value for the ldap-group option>
The LDAP Auth Realm admin-realm was configured correctly in admin server's configuration.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

change-admin-password(1), create-auth-realm(1) create-auth-realm(1),
list-auth-realms(1)

asadmin(1M)
copy-config

copies an existing named configuration to create another configuration

Synopsis

copy-config [--help]
[--systemproperties {name=value}[;name=value]*]
source-configuration-name destination-configuration-name

Description

The copy-config subcommand creates a named configuration in the configuration of the domain administration server (DAS) by copying an existing configuration. The new configuration is identical to the copied configuration, except for any properties that you specify in the --systemproperties option.

The default-config configuration is copied when a standalone server instance or standalone cluster is created.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--systemproperties
Optional attribute name-value pairs for the configuration. These properties override port settings in the configuration.

The following properties are available:

**ASADMIN_LISTENER_PORT**
This property specifies the port number of the HTTP port or HTTPS port through which the DAS connects to the instance to manage the instance. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**HTTP_LISTENER_PORT**
This property specifies the port number of the port that is used to listen for HTTP requests. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**HTTP_SSL_LISTENER_PORT**
This property specifies the port number of the port that is used to listen for HTTPS requests. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**IIOP_LISTENER_PORT**
This property specifies the port number of the port that is used for IIOP connections. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.
**IIOP_SSL_LISTENER_PORT**
This property specifies the port number of the port that is used for secure IIOP connections. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**IIOP_SSL_MUTUALAUTH_PORT**
This property specifies the port number of the port that is used for secure IIOP connections with client authentication. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**JAVA_DEBUGGER_PORT**
This property specifies the port number of the port that is used for connections to the Java Platform Debugger Architecture (JPDA) ([http://java.sun.com/javase/technologies/core/toolsapis/jpda/](http://java.sun.com/javase/technologies/core/toolsapis/jpda/)) debugger. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**JMS_PROVIDER_PORT**
This property specifies the port number for the Java Message Service provider. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**JMX_SYSTEM_CONNECTOR_PORT**
This property specifies the port number on which the JMX connector listens. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**OSGI_SHELL_TELNET_PORT**
This property specifies the port number of the port that is used for connections to the Apache Felix Remote Shell ([http://felix.apache.org/site/apache-felix-remote-shell.html](http://felix.apache.org/site/apache-felix-remote-shell.html)). This shell uses the Felix shell service to interact with the OSGi module management subsystem. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**Operands**

**source-configuration-name**
The name of the configuration that you are copying.

**destination-configuration-name**
The name of the configuration that you are creating by copying the source configuration.

The name must meet the following requirements:

- The name may contain only ASCII characters.
- The name must start with a letter, a number, or an underscore.
- The name may contain only the following characters:
  - Lowercase letters
  - Uppercase letters
  - Numbers
  - Hyphen
  - Period
The name must be unique in the domain and must not be the name of another named configuration, a cluster, a GlassFish Server instance, or a node.

The name must not be domain, server, or any other keyword that is reserved by GlassFish Server.

Examples

Example 1 Copying a Configuration
This example copies the default-config configuration to the pmdsaconfig configuration, overriding the settings for the following ports:

- HTTP listener port
- HTTPS listener port

```
asadmin> copy-config
--systemproperties HTTP_LISTENER_PORT=2000:HTTP_SSL_LISTENER_PORT=3000
default-config pmdsaconfig
```

Command copy-config executed successfully.

Exit Status

```
0
command executed successfully

1
error in executing the command
```

See Also

default-config(1), list-configs(1)

asadmin(1M)
create-admin-object

adds the administered object with the specified JNDI name for a resource adapter

Synopsis

create-admin-object [--help] [--target target]  
--restype restype  
[--classname classname]  
--raname raname  
[--enabled={true|false}]  
[--description description]  
[--property name=value[:name=value]*)  

Description

The create-admin-object subcommand creates the administered object with the specified JNDI name and the interface definition for a resource adapter. This subcommand is supported in remote mode only.

Options

--help  
-?  
Displays the help text for the subcommand.

--target

Specifies the target on which you are creating the administered object. Valid values are as follows:

server
Creates the administered object for the default server instance server and is the default value.

configuration_name
Creates the administered object for the named configuration.

cluster_name
Creates the administered object for every server instance in the cluster.

instance_name
Creates the administered object for a particular server instance.

---

Note: The resource is always created for the domain as a whole, but the resource-ref for the resource is only created for the specified --target. This means that although the resource is defined at the domain level, it is only available at the specified target level. Use the create-resource-ref subcommand to refer to the resource in multiple targets if needed.

---
--restype
Specifies the interface definition for the administered object. The resource type must be an interface definition that is specified in the ra.xml file of the resource adapter.

--classname
Specifies the class name of the administered object. Required if multiple administered objects use the same interface definition.

--raname
Specifies the name of the resource adapter associated with this administered object.

--enabled
Specifies if this object is enabled. Default is true.

--description
Text string describing the administered object.

--property
Description of the name/values pairs for configuring the resource. Dependent on the resource adapter. For JMS properties, see create-jms-resource(1) for JMS destination resources.

Operands

jndi_name
JNDI name of the administered object to be created.

Examples

Example 1 Creating an Administered Object
In this example, jmsra is a system resource adapter with the admin object interfaces, javax.jms.Queue and javax.jms.Topic.

asadmin> create-admin-object --restype javax.jms.Queue
--raname jmsra --description "sample administered object"
--property Name=sample_jmsqueue jms/samplequeue
Command create-admin-object executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-admin-object(1), list-admin-objects(1), create-resource-ref(1)
asadmin(1M)
create-application-ref

creates a reference to an application

Synopsis


Description

The create-application-ref subcommand creates a reference from a cluster or an unclustered server instance to a previously deployed application element (for example, a Java EE application, a Web module, or an enterprise bean module). This effectively results in the application element being deployed and made available on the targeted instance or cluster.

The target instance or instances making up the cluster need not be running or available for this subcommand to succeed. If one or more instances are not available, they will receive the new application element the next time they start.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target for which you are creating the application reference. Valid values are

- server- Specifies the default server instance as the target for creating the application reference. server is the name of the default server instance and is the default value for this option.

- cluster_name- Specifies a particular cluster as the target for creating the application reference.

- instance_name- Specifies a particular stand-alone server instance as the target for creating the application reference.

--virtualservers
Specifies a comma-separated list of virtual server IDs on which to deploy. This option applies only to Web modules (either standalone or in a Java EE application). If this option is not specified, the application is deployed to all virtual servers except the administrative server, __asadmin.

--enabled
Indicates whether the application should be enabled (that is, loaded). This value will take effect only if the application is enabled at the global level. The default is true.
--lbenabled
Controls whether the deployed application is available for load balancing. The default is true.

Operands

reference_name
The name of the application or module, which can be a Java EE application, Web module, EJB module, connector module, application client module, or lifecycle module.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter or number. It can contain alphanumeric characters plus underscore (_), dash (-), and period (.) characters. If the --enabled option is set to false, you can create references to multiple disabled versions by using an asterisk (*) as a wildcard character. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.

Examples

Example 1 Creating an Application Reference
The following example creates a reference to the Web module MyWebApp on the unclustered server instance NewServer.

```
asadmin> create-application-ref --target NewServer MyWebApp
Command create-application-ref executed successfully.
```

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-application-ref(1), list-application-refs(1)
asadmin(1M)

Oracle GlassFish Server Application Deployment Guide
create-audit-module

adds an audit module

Synopsis

create-audit-module [--help]
--classname classname
[--property {name=value}[[:name=value]*]
[--target target]
audit_module_name

Description

The create-audit-module subcommand adds the named audit module for the Java class that implements the audit capabilities. Audit modules collect and store information on incoming requests (from, for example, servlets and EJB components) and outgoing responses.

This subcommand is supported in remote mode only.

Options

--classname
The name of the Java class that implements this audit module. If not specified, this option defaults to com.sun.enterprise.security.Audit.

--help
-?
Displays the help text for the subcommand.

--property
Optional keyword-value pairs that specify additional properties for the audit module.
Audit module properties that are defined by GlassFish Server are as follows:

    auditOn
    If true, specifies that the audit module is loaded and called by the GlassFish Server audit library at audit points.

Other available properties are determined by the implementation of the audit module.

--target
Specifies the target on which you are creating the audit module. Valid values are as follows:

    server
    Creates the audit module for the default server instance server and is the default value.

    configuration_name
    Creates the audit module for the named configuration.

    cluster_name
    Creates the audit module for every server instance in the cluster.
instance_name
Creates the audit module for a particular server instance.

Operands

audit_module_name
The name of this audit module.

Examples

Example 1 Creating an audit module
asadmin> create-audit-module
--classname com.sun.appserv.auditmodule
--property defaultuser=admin:Password=admin sampleAuditModule
Command create-audit-module executed successfully

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-audit-module(1), list-audit-modules(1)
asadmin(1M)
create-auth-realm

adds the named authentication realm

Synopsis

create-auth-realm --classname realm_class [--help]
[--property{name=value}[:name=value]...]
[--target target_name] auth_realm_name

Description

The create-auth-realm subcommand adds the named authentication realm.
This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target on which you are creating the realm. Valid values are

server
Creates the realm on the default server instance. This is the default value.

configuration_name
Creates the realm in the specified configuration.

cluster_name
Creates the realm on all server instances in the specified cluster.

instance_name
Creates the realm on a specified server instance.

--classname
Java class which implements this realm. These include
com.sun.enterprise.security.auth.realm.file.FileRealm,
com.sun.enterprise.security.auth.realm.certificate.CertificateRealm,
com.sun.enterprise.security.auth.realm.jdbc.JDBCRealm,
com.sun.enterprise.security.auth.realm.ldap.LDAPRealm,
com.sun.enterprise.security.auth.realm.ldap.PamRealm, and
com.sun.enterprise.security.auth.realm.solaris.SolarisRealm, or a custom
realm.

--property
Optional attribute name-value pairs for configuring the authentication realm.
Authentication realms require provider-specific properties, which vary based on
implementation.

The following properties are common to all of the supported realms, which include
jaas-context
Specifies the Java Authentication and Authorization Service (JAAS) context.

assign-groups
(Optional) If this property is set, its value is taken to be a comma-separated list of
group names. All clients who present valid certificates are assigned membership
to these groups for the purposes of authorization decisions in the web and EJB
containers.

Specific to each realm, you can specify the following properties.

- You can specify the following properties for FileRealm:

  file
  Specifies the file that stores user names, passwords, and group names. The
default is domain-dir/config/keyfile.

- You can specify the following properties for CertificateRealm:

  LoginModule
  Specifies the name of a JAAS LoginModule to use for performing
  authentication. To use a JAAS LoginModule, you must first create an
  implementation of the javax.security.auth.spi>LoginModule interface, and then
  plug the module into a jaas-context. For more information, see “Custom
  Authentication of Client Certificate in SSL Mutual Authentication” in Oracle
  GlassFish Server Security Guide.

- You can specify the following properties for JDBCRealm:

  datasource-jndi
  Specifies the jndi-name of the jdbc-resource for the database.

  user-table
  Specifies the name of the user table in the database.

  user-name-column
  Specifies the name of the user name column in the database's user table.

  password-column
  Specifies the name of the password column in the database's user table.

  group-table
  Specifies the name of the group table in the database.

  group-name-column
  Specifies the name of the group name column in the database's group table.

  db-user
  (Optional) Allows you to specify the database user name in the realm instead
  of the jdbc-connection-pool. This prevents other applications from looking
  up the database, getting a connection, and browsing the user table. By default,
  the jdbc-connection-pool configuration is used.

  db-password
  (Optional) Allows you to specify the database password in the realm instead
  of the jdbc-connection-pool. This prevents other applications from looking
up the database, getting a connection, and browsing the user table. By default, the jdbc-connection-pool configuration is used.

**group-table**
Specifies the name of the group table in the database.

**digest-algorithm**
(Optional) Specifies the digest algorithm. The default is SHA-256. You can use any algorithm supported in the JDK, or none.

---

**Note:** In versions of GlassFish Server prior to 3.1.2, the default algorithm was MD5. If you have applications that depend on the MD5 algorithm, you can override the default SHA-25 algorithm by using the asadmin set subcommand:

```
asadmin> set
server.security-service.property.default-digest-algorithm=MD5
```

You can use the asadmin get subcommand to determine what algorithm is currently being used:

```
asadmin> get
server.security-service.property.default-digest-algorithm
```

Also note that, to maintain backward compatibility, if an upgrade is performed from GlassFish Server v2.x or v3.0.x to GlassFish Server 3.1.2, the default algorithm is automatically set to MD5 in cases where the digest algorithm had not been explicitly set in the older GlassFish Server version.

---

**digestrealm-password-enc-algorithm**
(Optional) Specifies the algorithm for encrypting passwords stored in the database.

---

**Note:** It is a security risk not to specify a password encryption algorithm.

---

**encoding**
(Optional) Specifies the encoding. Allowed values are Hex and Base64. If digest-algorithm is specified, the default is Hex. If digest-algorithm is not specified, by default no encoding is specified.

**charset**
(Optional) Specifies the charset for the digest algorithm.

- You can specify the following properties for LDAPRealm:

**directory**
Specifies the LDAP URL to your server.

**base-dn**
Specifies the LDAP base DN for the location of user data. This base DN can be at any level above the user data, since a tree scope search is performed. The smaller the search tree, the better the performance.
create-auth-realm

search-filter
(Optional) Specifies the search filter to use to find the user. The default is:
uid=%s (%s expands to the subject name).

group-base-dn
(Optional) Specifies the base DN for the location of groups data. By default, it is same as the base-dn, but it can be tuned, if necessary.

group-search-filter
(Optional) Specifies the search filter to find group memberships for the user. The default is uniquemember=%d (%d expands to the user elementDN).

group-target
(Optional) Specifies the LDAP attribute name that contains group name entries. The default is CN.

search-bind-dn
(Optional) Specifies an optional DN used to authenticate to the directory for performing the search-filter lookup. Only required for directories that do not allow anonymous search.

search-bind-password
(Optional) Specifies the LDAP password for the DN given in search-bind-dn.

Operands

auth_realm_name
A short name for the realm. This name is used to refer to the realm from, for example, web.xml.

Examples

Example 1  Creating a New Authentication Realm
This example creates a new file realm.

asadmin> create-auth-realm
--classname com.sun.enterprise.security.auth.realm.file.FileRealm
--property file=${com.sun.aas.instanceRoot}/config/
admin-keyfile:jaas-context=fileRealm file
admin-keyfile:jaas-context=fileRealm
Command create-auth-realm executed successfully

Where file is the authentication realm created.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-auth-realm(1), list-auth-realms(1)
asadmin(1M)
create-backup-config

creates a new domain backup configuration

Synopsis

create-backup-config [--help]
  [--schedule schedule-name]
  [--backupdir backup-directory]
  [--recyclelimit recycle-limit]
  [--configonly={false|true}]
  [--activebackupenabled={false|true}]
  [--autobackupenabled={true|false}]
  backup-config-name

Description

The create-backup-config subcommand creates a new domain backup configuration for automatically backing up a domain. Automatic backups can include the entire domain directory or just the config subdirectory in the domain directory.

This subcommand is supported in remote mode only.

Options

--help
  -?
  Displays the help text for the subcommand.

--schedule
  Specifies the name of the schedule to use for automatically backing up the domain.
  The default value is the predefined schedule daily.

--backupdir
  Specifies the base path where backup files are stored. GlassFish Server appends the subpath domain-dir/backup-config-name to the base path specified.
  This option has no default value. If no value is provided, backup files are stored in domain-dir/backups/backup-config-name.

--recyclelimit
  Specifies how many backup files to keep before deleting the oldest one to create a new backup.
  The default value is 25.

--configonly
  Specifies whether the backup is to contain only the configuration files for the domain. When true, only files in the config subdirectory of the domain directory are backed up. When false, all files in the domain directory are backed up, including deployed applications.
  The default value is false.
--activebackupenabled
Specifies whether scheduled backups can be performed without suspending the domain. When false, the domain is suspended before the backup operation and resumed after the backup operation completes. This suspension ensures that the content of the domain directory does not change during the backup operation.

The default value is false.

--autobackupenabled
Specifies whether automatic backups are enabled. When true, GlassFish Server automatically backs up the domain as indicated by the schedule specified in the domain backup configuration.

The default value is true.

Operands

backup-config-name
Specifies the name of the domain backup configuration to create. This name must be unique across all domain backup configurations in the domain.

Examples

Example 1 Creating an Automatic Monthly Backup
This example creates a domain backup configuration named monthly-backup to backup the domain directory every month, according to the monthly schedule. At most, twelve backup files of the domain will be retained.

asadmin> create-backup-config --schedule monthly
--recyclelimit 12 monthly-backup
Command create-backup-config executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-backup-config(1), disable-backup-config(1), enable-backup-config(1), list-backup-configs(1)
asadmin(1M)
create-cluster

creates a GlassFish Server cluster

Synopsis

create-cluster [--help] [--config config-name]
[--systemproperties (name=value)[;name=value]*]
[--properties (name=value)[;name=value]*]
[--gmsenabled={true|false}]
[--multicastport multicast-port]
[--multicastaddress multicast-address]
[--bindaddress bind-address]
[--hosts hadb-host-list]
[--haagentport port-number]
[--haadminpassword password]
[--haadminpasswordfile file-name] [--devicesize devicesize ]
[--haproperty (name=value)[;name=value]*]
[--autohadb=false] [--portbase port-number]
cluster-name

Description

The create-cluster subcommand creates a GlassFish Server cluster. Initially the cluster contains no GlassFish Server instances, applications, or resources.

A cluster requires a reference to the named configuration that defines the configuration of all instances that are added to the cluster. The configuration can be specified in the command to create the cluster, but is not required. If no configuration is specified, the subcommand creates a configuration that is named cluster-name-config for the cluster. The cluster that is created is a standalone cluster because the cluster's configuration is not shared with any other clusters or standalone instances.

To add instances to the cluster, set the --cluster option to the name of the cluster when using either of the following subcommands:

- create-instance(1)
- create-local-instance(1)

To delete server instances from the cluster at any time, use one of the following subcommands:

- delete-instance(1)
- delete-local-instance(1)

To associate applications and resources with all instances in the cluster, set the --target option to the name of the cluster when performing the following operations:

- Deploying applications by using the deploy(1) subcommand
- Creating resources by using subcommands such as create-jdbc-resource(1)
- Creating references to applications that are already deployed in other targets by using the create-application-ref(1) subcommand
Creating references to resources that are already created in other targets by using the `create-resource-ref(1)` subcommand

This subcommand is supported in remote mode only.

**Options**

---help
-?
Displays the help text for the subcommand.

--config
Specifies the named configuration that the cluster references. The configuration must exist and must not be named default-config or server-config. Specifying the --config option creates a shared cluster. If this option is omitted, a standalone cluster is created.

--systemproperties
Defines system properties for the configuration that is created for the cluster. These properties override the property values in the default-config configuration. The following properties are available:

**ASADMIN_LISTENER_PORT**
This property specifies the port number of the HTTP port or HTTPS port through which the DAS connects to the instance to manage the instance. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**HTTP_LISTENER_PORT**
This property specifies the port number of the port that is used to listen for HTTP requests. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**HTTP_SSL_LISTENER_PORT**
This property specifies the port number of the port that is used to listen for HTTPS requests. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**IIOP_LISTENER_PORT**
This property specifies the port number of the port that is used for IIOP connections. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**IIOP_SSL_LISTENER_PORT**
This property specifies the port number of the port that is used for secure IIOP connections. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**IIOP_SSL_MUTUALAUTH_PORT**
This property specifies the port number of the port that is used for secure IIOP connections with client authentication. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**JAVA_DEBUGGER_PORT**
This property specifies the port number of the port that is used for connections to the Java Platform Debugger Architecture (JPDA) ([http://java.sun.com/javase/technologies/core/toolsapis/jpda/](http://java.sun.com/javase/technologies/core/toolsapis/jpda/)) debugger.
Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**JMS_PROVIDER_PORT**
This property specifies the port number for the Java Message Service provider. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**JMX_SYSTEM_CONNECTOR_PORT**
This property specifies the port number on which the JMX connector listens. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**OSGI_SHELL_TELNET_PORT**
This property specifies the port number of the port that is used for connections to the Apache Felix Remote Shell (http://felix.apache.org/site/apache-felix-remote-shell.html). This shell uses the Felix shell service to interact with the OSGi module management subsystem. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

```
--properties
```
Defines properties for the cluster. The following properties are available:

**GMS_DISCOVERY_URI_LIST**
The locations of GlassFish Server instances in the cluster to use for discovering the cluster. This property is required only if the Group Management Service (GMS) is not using multicast for broadcasting messages.

Valid values for this property are as follows:

- A comma-separated list of uniform resource identifiers (URIs). Each URI must locate a GlassFish Server instance or the DAS. This format is required if multiple GlassFish Server instances are running on the same host.

  The format of each URI in the list is as follows:

  `scheme: //host-name-or -IP-address:port`

  - `scheme` is the URI scheme, which is tcp.
  - `host-name-or -IP-address` is the host name or IP address of the host on which the instance is running.
  - `port` is the port number of the port on which the instance listens for messages from GMS. The system property `GMS_LISTENER_PORT-classname` must be set for the instance. For information about how to set this system property for an instance, see "Discovering a Cluster When Multicast Transport Is Unavailable" in Oracle GlassFish Server High Availability Administration Guide.

- A comma-separated list of IP addresses or host names on which the DAS or the instances are running. The list can contain a mixture of IP addresses and host names. This format can be used only if one clustered instance is running on each host. The value of the `GMS_LISTENER_PORT` property must be unique for each cluster in a domain.

- The keyword `generate`. This format can be used only if one instance in a cluster is running on each host and the DAS is running on a separate host. Multiple instances on the same host cannot be members of the same cluster.
The value of the `GMS_LISTENER_PORT` property must be unique for each cluster in a domain.

**GMS_LISTENER_PORT**

The port number on which the cluster listens for messages from GMS. The default value is a reference to the `GMS_LISTENER_PORT-cluster-name` system property. By default, this system property is not set. In this situation, GMS selects a free port from the range that is defined by the properties `GMS_TCPSTARTPORT` and `GMS_TCPENDPORT`. By default, this range is 9090-9200. In most situations, the default behavior should suffice.

However, if GMS is not using multicast for broadcasting messages, the `GMS_LISTENER_PORT` property must specify a port number that is valid for all GlassFish Server instances in the cluster. To use the default value to meet this requirement, use a system property to set the port number individually for each instance.

For example, use the `create-system-properties` subcommand to create the system property `GMS_LISTENER_PORT-cluster-name` for the DAS. Then, for each instance in the cluster, set the `GMS_LISTENER_PORT-cluster-name` system property to the port number on which the instance listens for messages from GMS. The default value of the `GMS_LISTENER_PORT` property for the cluster references this system property.

**GMS_LOOPBACK**

Specifies whether an instance may receive from itself application-level messages that the instance broadcasts to the cluster.

Possible values are as follows:

- `false`
  - The instance may not receive messages from itself (default).

- `true`
  - The instance may receive messages from itself. Use this setting for testing an instance when the instance is the only instance in a cluster.

**GMS_MULTICAST_TIME_TO_LIVE**

The maximum number of iterations or transmissions that a multicast message for the following types of events can experience before the message is discarded:

- Group discovery
- Member heartbeats
- Membership changes

To match the configuration of the network on which the DAS and clustered instances are deployed, set this value as low as possible. To determine the lowest possible value for your system, use the `validate-multicast(I)` subcommand.

A value of 0 ensures that multicast messages never leave the host from which they are broadcast.

A value of 1 might prevent the broadcast of messages between hosts on same subnet that are connected by a switch or a router.

The default is 4, which ensures that messages are successfully broadcast to all cluster members in networks where hosts are connected by switches or routers.
**GMS_TCPENDPORT**
The highest port number in the range from which GMS selects a free port if the
GMS_LISTENER_PORT-\text{cluster-name} system property is not set. The default is 9200.

**GMS_TCPSTARTPORT**
The lowest port number in the range from which GMS selects a free port if the
GMS_LISTENER_PORT-\text{cluster-name} system property is not set. The default is 9090.

**--gmsenabled**
Specifies whether GMS is enabled for the cluster.
Possible values are as follows:

\textbf{true}
GMS is enabled for the cluster (default).

When GMS is enabled for a cluster, GMS is started in each server instance in the
cluster and in the DAS. The DAS participates in each cluster for which this option
is set to \textbf{true}.

\textbf{false}
GMS is disabled for the cluster.

**--multicastaddress**
The address on which GMS listens for group events. This option must specify a
multicast address in the range 224.0.0.0 through 239.255.255.255. The default is
228.9.XX.YY, where XX and YY are automatically generated independent values
between 0 and 255.

**--multicastport**
The port number of communication port on which GMS listens for group events. This
option must specify a valid port number in the range 2048-49151. The default is an
automatically generated value in this range.

**--bindaddress**
The Internet Protocol (IP) address of the network interface to which GMS binds. This
option must specify the IP address of a local network interface. The default is all public
network interface addresses.

On a multihome machine, this option configures the network interface that is used for
the GMS. A multihome machine possesses two or more network interfaces.

To specify an address that is valid for all GlassFish Server instances in the cluster, use a
system property to set the address individually for each instance.

For example, use the \texttt{create-system-properties} subcommand to create the system
property \texttt{GMS-BIND-INTERFACE-ADDRESS-\text{cluster-name}}. Then set the \texttt{--bindaddress}
option of this subcommand to \texttt{${GMS-BIND-INTERFACE-ADDRESS-\text{cluster-name}}\texttt{}} to
specify the system property. Finally, for each instance in the cluster, set the
\texttt{GMS-BIND-INTERFACE-ADDRESS-\text{cluster-name}} system property to the required network
interface address on the instance’s machine.

**--hosts**
Do not specify this option. This option is retained for compatibility with earlier
releases. If you specify this option, a syntax error does not occur. Instead, the
subcommand runs successfully and displays a warning message that the option is
ignored.
--haagentport
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

--haadminpassword
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

--haadminpasswordfile
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

--devicesize
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

--haproperty
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

--autohadb
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

--portbase
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

Operands

cluster-name
The name of the cluster.

The name must meet the following requirements:

■ The name may contain only ASCII characters.
■ The name must start with a letter, a number, or an underscore.
■ The name may contain only the following characters:
  - Lowercase letters
  - Uppercase letters
  - Numbers
The name must be unique in the domain and must not be the name of another cluster, a named configuration, a GlassFish Server instance, or a node.

The name must not be `domain`, `server`, or any other keyword that is reserved by GlassFish Server.

**Examples**

**Example 1  Creating a Cluster**
This example creates a cluster that is named `ltscluster` for which port 1169 is to be used for secure IIOP connections. Because the `--config` option is not specified, the cluster references a copy of the named configuration `default-config` that is named `ltscluster-config`.

```bash
asadmin> create-cluster
   --systemproperties IIOP_SSL_LISTENER_PORT=1169
   ltscluster
Command create-cluster executed successfully.
```

**Example 2  Creating a Cluster With a List of URIs for Discovering the Cluster**
This example creates a cluster that is named `tcpcluster`. In this example, GMS is not using multicast for broadcasting messages and multiple instances reside on the same host. Therefore, the `GMS_DISCOVERY_URI_LIST` property is set to the locations of the GlassFish Server instances to use for discovering the cluster. These instances reside on the host whose IP address is `10.152.23.224` and listen for GMS events on ports `9090`, `9091`, and `9092`.

To distinguish colon (`:`) characters in URIs from separators in a property list, colons in URIs are escaped with single quote characters (`'`) and backslash (`\`) characters. For more information about escape characters in options for the `asadmin` utility, see the `asadmin(1M)` help page.

This example assumes that the port on which each instance listens for GMS messages is set independently for the instance through the `GMS_LISTENER_PORT-tcpcluster` system property. For information about how to set the port on which an instance listens for GMS messages, see "Discovering a Cluster When Multicast Transport Is Unavailable" in Oracle GlassFish Server High Availability Administration Guide.

```bash
asadmin> create-cluster --properties GMS_DISCOVERY_URI_LIST=
tcp'://'10.152.23.224'::'9090,
tcp'://'10.152.23.224'::'9091,
tcp'://'10.152.23.224'::'9092 tcpcluster
Command create-cluster executed successfully.
```

**Example 3  Creating a Cluster With a List of IP Addresses for Discovering the Cluster**
This example creates a cluster that is named `ipcluster`. In this example, GMS is not using multicast for broadcasting messages and only one clustered instance resides on each host. Therefore, the `GMS_DISCOVERY_URI_LIST` property is set to the IP addresses of the hosts where instances to use for discovering the cluster are running. The cluster listens for messages from GMS on port `9090`.

```bash
asadmin> create-cluster --properties GMS_DISCOVERY_URI_LIST=
Tcp'://10.152.23.224'::'9090,
Tcp'://10.152.23.224'::'9091,
Tcp'://10.152.23.224'::'9092 ipcluster
Command create-cluster executed successfully.
```
Example 4  Creating a Cluster With a Generated List of Instances for Discovering the Cluster

This example creates a cluster that is named gencluster. In this example, GMS is not using multicast for broadcasting messages, one instance in the cluster is running on each host and the DAS is running on a separate host. Therefore, the GMS_DISCOVERY_URI_LIST property is set to the keyword generate to generate a list of instances to use for discovering the cluster. The cluster listens for messages from GMS on port 9090.

asadmin> create-cluster --properties 'GMS_DISCOVERY_URI_LIST=generate:
GMS_LISTENER_PORT=9090' gencluster
Command create-cluster executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-application-ref(1), create-instance(1), create-jdbc-resource(1),
create-local-instance(1), create-resource-ref(1), delete-cluster(1),
delete-instance(1), delete-local-instance(1), deploy(1), list-clusters(1),
start-cluster(1), stop-cluster(1), validate-multicast(1)

asadmin(1M)

"Discovering a Cluster When Multicast Transport Is Unavailable" in Oracle GlassFish Server High Availability Administration Guide

Apache Felix Remote Shell
(http://felix.apache.org/site/apache-felix-remote-shell.html), Java Platform Debugger Architecture (JPDA)
(http://java.sun.com/javase/technologies/core/toolsapis/jpda/)
create-connector-connection-pool

adds a connection pool with the specified connection pool name

Synopsis

create-connector-connection-pool [--help] [--target=target] 
--raname raname
--connectiondefinition connectiondefinitionname 
[--steadypoolsize steadypoolsize] 
[--maxpoolsize maxpoolsize] 
[--maxwait maxwait] 
[--poolsize poolsize] 
[--idletimeout idletimeout] 
[--isconnectvalidatereq={false|true}] 
[--failconnection={false|true}] 
[--leaktimeout=timeout] 
[--leakreclaim={false|true}] 
[--creationretryattempts=attempts] 
[--creationretryinterval=interval] 
[--lazyconnectionenlistment={false|true}] 
[--lazyconnectionassociation={false|true}] 
[--associatewiththread={false|true}] 
[--matchconnections={true|false}] 
[--maxconnectionusagecount=count] 
[--validateatmostonceperiod=interval] 
[--transactionsupport transactionsupport] 
[--descrip[ition description] 
[--ping {false|true}] 
[--pooling {true|false}] 
[--property (name=value)[;name=value]*]

Description

The create-connector-connection-pool subcommand defines a pool of connections to an enterprise information system (EIS). The named pool can be referred to by multiple connector resources. Each defined pool is instantiated at server startup, and is populated when accessed for the first time. If two or more connector resources point to the same connector connection pool, they are using the same pool of connections at run time. There can be more than one pool for a connection definition in a single resource adapter.

A connector connection pool with authentication can be created either by using a --property option to specify user, password, or other connection information, or by specifying the connection information in the XML descriptor file.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.
--associatewiththread
Specifies whether a connection is associated with the thread to enable the thread to reuse the connection. If a connection is not associated with the thread, the thread must obtain a connection from the pool each time that the thread requires a connection. Possible values are as follows:

false
A connection is not associated with the thread (default).

true
A connection is associated with the thread.

--connectiondefinition
The name of the connection definition.

--creationretryattempts
Specifies the maximum number of times that the server retries to create a connection if the initial attempt fails.
Default value is 0, which specifies that the server does not retry to create the connection.

--creationretryinterval
Specifies the interval, in seconds, between successive attempts to create a connection. If --creationretryattempts is 0, the --creationretryinterval option is ignored. Default value is 10.

--description
Text providing descriptive details about the connector connection pool.

--failconnection
If set to true, all connections in the pool are closed if a single validation check fails. This parameter is mandatory if the --isconnectvalidatereq option is set to true. Default value is false.

--idletimeout
The maximum time that a connection can remain idle in the pool. After this amount of time, the pool can close this connection. Default value is 300.

--isconnectvalidatereq
If the value is set to true, the connections will be checked to see if they are usable, before they are given out to the application. Default value is false.

--lazyconnectionenlistment
Specifies whether a resource to a transaction is enlisted only when a method actually uses the resource. Default value is false.

--lazyconnectionassociation
Specifies whether a physical connection should be associated with the logical connection only when the physical connection is used, and disassociated when the transaction is completed. Such association and dissociation enable the reuse of physical connections. Possible values are as follows:

false
A physical connection is associated with the logical connection even before the physical connection is used, and is not disassociated when the transaction is completed (default).
true
A physical connection is associated with the logical connection only when the physical connection is used, and disassociated when the transaction is completed. The `--lazyconnectionenlistment` option must also be set to true.

**--leakreclaim**
Specifies whether leaked connections are restored to the connection pool after leak connection tracing is complete. Possible values are as follows:

**false**
Leaked connections are not restored to the connection pool (default).

**true**
Leaked connections are restored to the connection pool.

**--leaktimeout**
Specifies the amount of time, in seconds, for which connection leaks in a connection pool are to be traced.

If connection leak tracing is enabled, you can use the Administration Console to enable monitoring of the JDBC connection pool to get statistics on the number of connection leaks. Default value is 0, which disables connection leak tracing.

**--matchconnections**
Specifies whether a connection that is selected from the pool should be matched with the resource adaptor. If all connections in the pool are identical, matching between connections and resource adapters is not required. Possible values are as follows:

**true**
A connection should be matched with the resource adaptor (default).

**false**
A connection should not be matched with the resource adaptor.

**--maxconnectionusagecount**
Specifies the maximum number of times that a connection can be reused.

When this limit is reached, the connection is closed. Default value is 0, which specifies no limit on the number of times that a connection can be reused.

**--maxpoolsize**
The maximum number of connections that can be created to satisfy client requests. Default value is 32.

**--maxwait**
The amount of time, in milliseconds, that a caller must wait before a connection is created, if a connection is not available. If set to 0, the caller is blocked indefinitely until a resource is available or until an error occurs. Default value is 60000.

**--ping**
A pool with this attribute set to true is contacted during creation (or reconfiguration) to identify and warn of any erroneous values for its attributes. Default value is false.

**--pooling**
When set to false, this attribute disables connection pooling. Default value is true.
--poolresize
Quantity by which the pool will scale up or scale down the number of connections.
Scale up: When the pool has no free connections, pool will scale up by this quantity.
Scale down: All the invalid and idle connections are removed, sometimes resulting in
removing connections of quantity greater than this value. The number of connections
that is specified by --steadypoolsize will be ensured. Possible values are from 0 to
MAX_INTEGER. Default value is 2.

--property
Optional attribute name/value pairs for configuring the pool.

  LazyConnectionEnlistment
  Deprecated. Use the equivalent option. Default value is false.

  LazyConnectionAssociation
  Deprecated. Use the equivalent option. Default value is false.

  AssociateWithThread
  Deprecated. Use the equivalent option. Default value is false.

  MatchConnections
  Deprecated. Use the equivalent option. Default value is false.

--rename
The name of the resource adapter.

--steadypoolsize
The minimum and initial number of connections maintained in the pool. Default value is 8.

--target
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

--transactionsupport
Indicates the level of transaction support that this pool will have. Possible values are XATransaction, LocalTransaction and NoTransaction. This attribute can have a value lower than or equal to but not higher than the resource adapter's transaction support attribute. The resource adapter's transaction support attribute has an order of values, where XATransaction is the highest, and NoTransaction the lowest.

--validateatmostonceperiod
Specifies the time interval in seconds between successive requests to validate a connection at most once. Setting this attribute to an appropriate value minimizes the number of validation requests by a connection. Default value is 0, which means that the attribute is not enabled.

Operands

  poolname
  The name of the connection pool to be created.
Examples

Example 1 Creating a Connector Connection Pool
This example creates a new connector connection pool named jms/qConnPool.

```bash
asadmin> create-connector-connection-pool --rename jmsra
--connectiondefinition javax.jms.QueueConnectionFactory
--steadypoolsize 20
--maxpoolsize 100 --poolresize 2 --maxwait 60000 jms/qConnPool
Command create-connector-connection-pool executed successfully
```

Exit Status

0  
subcommand executed successfully

1  
error in executing the subcommand

See Also

delete-connector-connection-pool(1), list-connector-connection-pools(1),
ping-connection-pool(1)
asadmin(1M)
create-connector-resource

registers the connector resource with the specified JNDI name

Synopsis

create-connector-resource [--help]
--poolname connectorConnectionPoolName
[--enabled={true|false}]
[--description description]
[--objecttype objecttype]
[--property (name=value)[:name=value]*]
[--target target]
jndi_name

Description

The create-connector-resource subcommand registers the connector resource with the specified JNDI name.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--poolname
The name of the connection pool. When two or more resource elements point to the same connection pool element, they use the same pool connections at runtime.

--enabled
This option determines whether the resource is enabled at runtime. The default value is true.

--objecttype
Defines the type of the connector resource. Default is user. Allowed values are:

    system-all
    A system resource for all server instances and the domain administration server (DAS).

    system-admin
    A system resource only for the DAS.

    system-instance
    A system resource for all server instances only.

    user
    A user resource.

--description
Text providing details about the connector resource.
create-connector-resource

--property
Optional attribute name value pairs for configuring the resource.

--target
This option specifies the ending location of the connector resources. Valid targets are:

  server
  Creates the connector resource in the default server instance. This is the default value.

  domain
  Creates the connector resource in the domain.

  cluster_name
  Creates the connector resource in every server instance in the cluster.

  instance_name
  Creates the connector resource in the specified server instance.

Note: The resource is always created for the domain as a whole, but the resource-ref for the resource is only created for the specified --target. This means that although the resource is defined at the domain level, it is only available at the specified target level. Use the create-resource-ref subcommand to refer to the resource in multiple targets if needed.

Operands

  jndi_name
  The JNDI name of this connector resource.

Examples

Example 1  Creating a Connector Resource
This example creates a connector resource named jms/qConnFactory.

  asadmin> create-connector-resource --poolname jms/qConnPool
          --description "sample connector resource" jms/qConnFactory
  Command create-connector-resource executed successfully

Example 2  Using the create-connector-resource subcommand
This example shows the usage of this subcommand.

  asadmin> create-connector-resource --target server --poolname jms/qConnPool
          --description "sample connector resource" jms/qConnFactory
  Command create-connector-resource executed successfully

Where jms/qConnFactory is the sample connector resource that is created.

Exit Status

  0
  subcommand executed successfully
1
error in executing the subcommand

See Also

delete-connector-resource(1), list-connector-resources(1),
create-resource-ref(1)

asadmin(1M)
create-connector-security-map

creates a security map for the specified connector connection pool

Synopsis

create-connector-security-map [--help]
--poolname connector_connection_pool_name
[--principals principal-name1,[principal-name2]...]
[--usergroups user-group1,[user-group2]...]
[--mappedusername user-name]
[--target target]
mapname

Description

The create-connector-security-map subcommand creates a security map for the specified connector connection pool. If the security map is not present, a new one is created. This subcommand can also map the caller identity of the application (principal or user group) to a suitable enterprise information system (EIS) principal in container-managed authentication scenarios. The EIS is any system that holds the data of an organization. It can be a mainframe, a messaging system, a database system, or an application. One or more named security maps can be associated with a connector connection pool. The connector security map configuration supports the use of the wild card asterisk (*) to indicate all users or all user groups.

To specify the EIS password, you can add the AS_ADMIN_MAPPEDPASSWORD entry to the password file, then specify the file by using the --passwordfile asadmin utility option.

For this subcommand to succeed, you must have first created a connector connection pool using the create-connector-connection-pool subcommand.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--poolname
Specifies the name of the connector connection pool to which the security map belongs.

--principals
Specifies a list of backend EIS principals. More than one principal can be specified using a comma-separated list. Use either the --principals or --usergroups options, but not both in the same command.

--usergroups
Specifies a list of backend EIS user group. More than one user groups can be specified using a comma separated list. Use either the --principals or --usergroups options, but not both in the same command.
--mappedusername
Specifies the EIS username.

--target
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

Operands

mapname
The name of the security map to be created.

Examples

Example 1 Creating a Connector Security Map
This example creates securityMap1 for the existing connection pool named connector-pool1.

asadmin> create-connector-security-map --poolname connector-pool1
--principals principal1,principal2 --mappedusername backend-username securityMap1
Command create-connector-security-map executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-connector-security-map(1), list-connector-security-maps(1),
update-connector-security-map(1)
asadmin(1M)
create-connector-work-security-map

creates a work security map for the specified resource adapter

Synopsis

create-connector-work-security-map [--help] --raname raname
[--principalsmap eis-principal1=principal_name1[, eis-principal2=principal_name2]*]
[--groupsmap eis-group1=server-group1[, eis-group2=server-group2]*]
[--description description]
mapname

Description

The create-connector-work-security-map subcommand maps the caller identity of the work submitted by the resource adapter EIS principal or EIS user group to a suitable principal or user group in the GlassFish Server security domain. One or more work security maps may be associated with a resource adapter. The connector work security map configuration supports the use of the wild card asterisk (*) to indicate all users or all user groups.

The enterprise information system (EIS) is any system that holds the data of an organization. It can be a mainframe, a messaging system, a database system, or an application.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--description
Text providing descriptive details about the connector work security map.

--groupsmap
Specifies a map of the backend EIS user group to the GlassFish Server user group. Use a comma-separated list to specify more than one mapping. Use either the --principalsmap option or the --groupsmap option, but not both.

--principalsmap
Specifies a map of the backend EIS principal to the GlassFish Server principal. Use a comma-separated list to specify more than one mapping. Use either the --principalsmap option or the --groupsmap option, but not both.

--raname
Indicates the connector module name, which is the name of the resource adapter.

Operands

mapname
The name of the work security map to be created.
Examples

Example 1 Creating a Connector Work Security Map (Principal)
This example creates connector work security map workSecurityMap1 that maps the backend EIS principal to the GlassFish Server principal.

```
asadmin create-connector-work-security-map --raname my-resource-adapter
--principalmap eis-principal-1=server-principal-1,eis-principal-2=server-principal-2,eis-principal-3=server-principal-1
workSecurityMap1
Command create-connector-work-security-map executed successfully.
```

Example 2 Creating a Connector Work Security Map (Group)
This example creates connector work security map workSecurityMap2 that maps the backend EIS user group to the GlassFish Server user group.

```
asadmin create-connector-work-security-map --raname my-resource-adapter
--groupmap eis-group-1=server-group-1,eis-group-2=server-group-2,eis-group-3=server-group-1 workSecurityMap2
Command create-connector-work-security-map executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-connector-work-security-map(1), list-connector-work-security-maps(1),
update-connector-work-security-map(1)
asadmin(1M)
**create-custom-resource**

creates a custom resource

**Synopsis**

create-custom-resource [--help] --restype type --factoryclass classname
[--enabled={true|false}] [--description text]
[--property (name=value)[:name=value]*] jndi-name
[--target target]

**Description**

The *create-custom-resource* subcommand creates a custom resource. A custom resource specifies a custom server-wide resource object factory that implements the `javax.naming.spi.ObjectFactory` interface.

This subcommand is supported in remote mode only.

**Options**

--help

-?

Displays the help text for the subcommand.

--target

The target on which the custom resource you are creating will be available. Valid values are:

**server**

The resource will be available on the default server instance and all domains hosted on the instance. This is the default value.

**domain**

The resource will be available on the specified domain only.

**cluster_name**

The resource will be available on every server instance in the cluster.

**instance_name**

The resource will be available on the specified server instance only.

---

**Note:** The resource is always created for the domain as a whole, but the resource-ref for the resource is only created for the specified --target. This means that although the resource is defined at the domain level, it is only available at the specified target level. Use the *create-resource-ref* subcommand to refer to the resource in multiple targets if needed.
--restype
The type of custom resource to be created. Specify a fully qualified type definition, for example javax.naming.spi.ObjectFactory. The resource type definition follows the format, xxx.xxx.

--factoryclass
Factory class name for the custom resource. This class implements the javax.naming.spi.ObjectFactory interface.

--enabled
Determines whether the custom resource is enable at runtime. Default is true.

--description
Text providing details about the custom resource. This description is a string value and can include a maximum of 250 characters.

--property
Optional attribute name/value pairs for configuring the resource.

Operands

jndi-name
The JNDI name of this resource.

Examples

Example 1 Creating a Custom Resource
This example creates a custom resource.

asadmin> create-custom-resource --restype topic
--factoryclass com.imq.topic mycustomresource
Command create-custom-resource executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

deleate-custom-resource(1), list-custom-resources(1), create-resource-ref(1)
asadmin(1M)
create-domain

creates a domain

Synopsis

create-domain [--help]
[--adminport adminport]
[--instanceport instanceport]
[--portbase portbase]
[--profile profile-name]
[--template template-name]
[--domaindir domaindir]
[--savemasterpassword={false|true}]
[--usemasterpassword={false|true}]
[--domainproperties (name=value)[:name=value]*]
[--keytooloptions (name=value)[:name=value]*]
[--savelogin={false|true}]
[--checkports={true|false}]
[--nopassword={false|true}]
domain-name

Description

The create-domain subcommand creates a GlassFish Server domain. A domain in GlassFish Server is an administrative namespace that complies with the Java Platform, Enterprise Edition (Java EE) standard. Every domain has a configuration, which is stored in a set of files. Any number of domains, each of which has a distinct administrative identity, can be created in a given installation of GlassFish Server. A domain can exist independently of other domains.

Any user who has access to the asadmin utility on a given system can create a domain and store its configuration in a folder of the user’s choosing. By default, the domain configuration is created in the default directory for domains. You can override this location to store the configuration elsewhere.

If domain customizers are found in JAR files in the as-install/modules directory when the create-domain subcommand is run, the customizers are processed. A domain customizer is a class that implements the DomainInitializer interface.

The create-domain subcommand creates a domain with a single administrative user specified by the asadmin utility option --user. If the --user option is not specified, and the --nopassword option is set to true, the default administrative user, admin, is used. If the --nopassword option is set to false (the default), a username is required. In this case, if you have not specified the user name by using the --user option, you are prompted to do so.

You choose an appropriate profile for the domain, depending on the applications that you want to run on your new domain. You can choose the developer, cluster, or enterprise profile for the domain you create.

This subcommand is supported in local mode only.
Options

--help
-?
Displays the help text for the subcommand.

--adminport
The HTTP port or the HTTPS port for administration. This port is the port in the URL that you specify in your web browser to manage the domain, for example, http://localhost:4949. The --adminport option cannot be used with the --portbase option. The default value is 4848.

The --adminport option overrides the domain.adminPort property of the --domainproperties option.

--instanceport
The domain provides services so that applications can run when deployed. This HTTP port specifies where the web application context roots are available for a web browser to connect to. This port is a positive integer and must be available at the time of domain creation. The --instanceport option cannot be used with the --portbase option. The default value is 8080.

The --instanceport option overrides the domain.instancePort property of the --domainproperties option.

--portbase
Determines the number with which port assignments should start. A domain uses a certain number of ports that are statically assigned. The portbase value determines where the assignment should start. The values for the ports are calculated as follows:

- Administration port: portbase + 48
- HTTP listener port: portbase + 80
- HTTPS listener port: portbase + 81
- JMS port: portbase + 76
- IIOP listener port: portbase + 37
- Secure IIOP listener port: portbase + 38
- Secure IIOP with mutual authentication port: portbase + 39
- JMX port: portbase + 86
- JPDA debugger port: portbase + 9
- Felix shell service port for OSGi module management: portbase + 66

When the --portbase option is specified, the output of this subcommand includes a complete list of used ports.

The --portbase option cannot be used with the --adminport, --instanceport, or the --domainproperties option.

--profile
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.
The file name, including a relative or absolute path, of a domain configuration template to use for creating the domain. If a relative path is specified, the subcommand appends the path to the `as-install/lib/templates` directory to locate the file. If it is an absolute pathname, the subcommand locates the file in the specified path.

This option enables domains of different types to be created and custom domain templates to be defined.

The directory where the domain is to be created. If specified, the path must be accessible in the filesystem. If not specified, the domain is created in the default domain directory, `as-install/domains`.

Setting this option to `true` allows the master password to be written to the file system. If this option is `true`, the `--usemasterpassword` option is also `true`, regardless of the value that is specified on the command line. The default value is `false`.

A master password is really a password for the secure key store. A domain is designed to keep its own certificate (created at the time of domain creation) in a safe place in the configuration location. This certificate is called the domain's SSL server certificate. When the domain is contacted by a web browser over a secure channel (HTTPS), this certificate is presented by the domain. The master password is supposed to protect the store (a file) that contains this certificate. This file is called `keystore.jks` and is created in the configuration directory of the domain created. If however, this option is chosen, the master password is saved on the disk in the domain's configuration location. The master password is stored in a file called `master-password`, which is a Java JCEKS type keystore. The reason for using the `--savemasterpassword` option is for unattended system boots. In this case, the master password is not prompted for when the domain starts because the password will be extracted from this file.

It is best to create a master password when creating a domain, because the master password is used by the `start-domain` subcommand. For security purposes, the default setting should be `false`, because saving the master password on the disk is an insecure practice, unless file system permissions are properly set. If the master password is saved, then `start-domain` does not prompt for it. The master password gives an extra level of security to the environment.

Specifies whether the key store is encrypted with a master password that is built into the system or a user-defined master password.

If `false` (default), the keystore is encrypted with a well-known password that is built into the system. Encrypting the keystore with a password that is built into the system provides no additional security.

If `true`, the subcommand obtains the master password from the `AS_ADMIN_MASTERPASSWORD` entry in the password file or prompts for the master password. The password file is specified in the `--passwordfile` option of the `asadmin(1M)` utility.

If the `--savemasterpassword` option is `true`, this option is also `true`, regardless of the value that is specified on the command line.

Setting the optional name/value pairs overrides the default values for the properties of the domain to be created. The list must be separated by the colon `:` character. The
--portbase options cannot be used with the --domainproperties option. The following properties are available:

**domain.adminPort**
This property specifies the port number of the HTTP port or the HTTPS port for administration. This port is the port in the URL that you specify in your web browser to manage the instance, for example, `http://localhost:4949`. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

The `domain.adminPort` property is overridden by the `--adminport` option.

**domain.instancePort**
This property specifies the port number of the port that is used to listen for HTTP requests. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

The `domain.instancePort` property is overridden by `--instanceport` option.

**domain.jmxPort**
This property specifies the port number on which the JMX connector listens. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**http.ssl.port**
This property specifies the port number of the port that is used to listen for HTTPS requests. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**java.debugger.port**
This property specifies the port number of the port that is used for connections to the Java Platform Debugger Architecture (JPDA) ([http://java.sun.com/javase/technologies/core/toolsapis/jpda/](http://java.sun.com/javase/technologies/core/toolsapis/jpda/)) debugger. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**jms.port**
This property specifies the port number for the Java Message Service provider. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**orb.listener.port**
This property specifies the port number of the port that is used for IIOP connections. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**orb.mutualauth.port**
This property specifies the port number of the port that is used for secure IIOP connections with client authentication. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**orb.ssl.port**
This property specifies the port number of the port that is used for secure IIOP connections. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.
**osgi.shell.telnet.port**
This property specifies the port number of the port that is used for connections to the Apache Felix Remote Shell (http://felix.apache.org/site/apache-felix-remote-shell.html). This shell uses the Felix shell service to interact with the OSGi module management subsystem. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**--keytooloptions**
Specifies an optional list of name-value pairs of keytool options for a self-signed server certificate. The certificate is generated during the creation of the domain. Each pair in the list must be separated by the colon (:) character.

Allowed options are as follows:

- **CN**
  Specifies the common name of the host that is to be used for the self-signed certificate. This option name is case insensitive.
  
  By default, the name is the fully-qualified name of the host where the create-domain subcommand is run.

**--savelogin**
If set to true, this option saves the administration user name and password. Default value is false. The username and password are stored in the .asadminpass file in user’s home directory. A domain can only be created locally. Therefore, when using the --savelogin option, the host name saved in .asadminpass is always localhost. If the user has specified default administration port while creating the domain, there is no need to specify --user, --passwordfile, --host, or --port on any of the subsequent asadmin remote commands. These values will be obtained automatically.

---

**Note:** When the same user creates multiple domains that have the same administration port number on the same or different host (where the home directory is NFS mounted), the subcommand does not ask if the password should be overwritten. The password will always be overwritten.

**--checkports**
Specifies whether to check for the availability of the administration, HTTP, JMS, JMX, and IIOP ports. The default value is true.

**--nopassword**
Specifies whether the administrative user will have a password. If false (the default), the password is specified by the AS_ADMIN_PASSWORD entry in the asadmin password file (set by using the --passwordfile option). If false and the AS_ADMIN_PASSWORD is not set, you are prompted for the password.

If true, the administrative user is created without a password. If a user name for the domain is not specified by using the --user option, and the --nopassword option is set to true, the default user name, admin, is used.
Operands

domain-name
The name of the domain to be created. The name may contain only ASCII characters and must be a valid directory name for the operating system on the host where the domain is created.

Examples

Example 1 Creating a Domain
This example creates a domain named domain4.

asadmin> create-domain --adminport 4848 domain4
Enter admin user name [Enter to accept default "admin" / no password]>
Using port 4848 for Admin.
Using default port 8080 for HTTP Instance.
Using default port 7676 for JMS.
Using default port 3700 for IIOP.
Using default port 8181 for HTTP_SSL.
Using default port 3820 for IIOP_SSL.
Using default port 3920 for IIOP_MUTUALAUTH.
Using default port 8686 for JMX_ADMIN.
Using default port 6666 for OSGI_SHELL.
Distinguished Name of the self-signed X.509 Server Certificate is:
[CN=sr1-usca-22,OU=GlassFish,O=Oracle Corp.,L=Redwood Shores,ST=California,C=US]
No domain initializers found, bypassing customization step
Domain domain4 created.
Domain domain4 admin port is 4848.
Domain domain4 allows admin login as user "admin" with no password.
Command create-domain executed successfully.

Example 2 Creating a Domain in an Alternate Directory
This example creates a domain named sampleDomain in the /home/someuser/domains directory.

asadmin> create-domain --domaindir /home/someuser/domains --adminport 7070 --instanceport 7071 sampleDomain
Enter admin user name [Enter to accept default "admin" / no password]>
Using port 7070 for Admin.
Using port 7071 for HTTP Instance.
Using default port 7676 for JMS.
Using default port 3700 for IIOP.
Using default port 8181 for HTTP_SSL.
Using default port 3820 for IIOP_SSL.
Using default port 3920 for IIOP_MUTUALAUTH.
Using default port 8686 for JMX_ADMIN.
Using default port 6666 for OSGI_SHELL.
Enterprise ServiceDistinguished Name of the self-signed X.509 Server Certificate is:
[CN=sr1-usca-22,OU=GlassFish,O=Oracle Corp.,L=Redwood Shores,ST=California,C=US]
No domain initializers found, bypassing customization step
Domain sampleDomain created.
Domain sampleDomain admin port is 7070.
Domain sampleDomain allows admin login as user "admin" with no password.
Command create-domain executed successfully.
Example 3 Creating a Domain and Saving the Administration User Name and Password

This example creates a domain named `myDomain` and saves the administration username and password.

```
asadmin> create-domain --adminport 8282 --savelogin=true myDomain
Enter the admin password [Enter to accept default of no password]>
Enter the master password [Enter to accept default password "changeit"]>
Using port 8282 for Admin.
Using default port 8080 for HTTP Instance.
Using default port 7676 for JMS.
Using default port 3700 for IIOP.
Using default port 8181 for HTTP_SSL.
Using default port 3820 for IIOP_SSL.
Using default port 3920 for IIOP_MUTUALAUTH.
Using default port 8686 for JMX_ADMIN.
Using default port 6666 for OSGI_SHELL.
Enterprise Service Distinguished Name of the self-signed X.509 Server Certificate is:
[CN=sr1-usca-22,OU=GlassFish,O=Oracle Corp.,L=Redwood Shores,ST=California,C=US]
No domain initializers found, bypassing customization step
Domain myDomain created.
Domain myDomain admin port is 8282.
Domain myDomain allows admin login as user "admin" with no password.
Login information relevant to admin user name [admin] for this domain [myDomain] stored at
[/home/someuser/.asadminpass] successfully.
Make sure that this file remains protected.
Information stored in this file will be used by asadmin commands to manage this domain.
Command create-domain executed successfully.
```

Example 4 Creating a Domain and Designating the Certificate Host

This example creates a domain named `domain5`. The common name of the host that is to be used for the self-signed certificate is `trio`.

```
asadmin> create-domain --adminport 9898 --keytooloptions CN=trio domain5
Enter the admin password [Enter to accept default of no password]>
Enter the master password [Enter to accept default password "changeit"]>
Using port 9898 for Admin.
Using default port 8080 for HTTP Instance.
Using default port 7676 for JMS.
Using default port 3700 for IIOP.
Using default port 8181 for HTTP_SSL.
Using default port 3820 for IIOP_SSL.
Using default port 3920 for IIOP_MUTUALAUTH.
Using default port 8686 for JMX_ADMIN.
Using default port 6666 for OSGI_SHELL.
Distinguished Name of the self-signed X.509 Server Certificate is:
[CN=trio,OU=GlassFish,O=Oracle Corp.,L=Redwood Shores,ST=California,C=US]
No domain initializers found, bypassing customization step
Domain domain5 created.
Domain domain5 admin port is 9898.
Domain domain5 allows admin login as user "admin" with no password.
Command create-domain executed successfully.
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

login(1), delete-domain(1), start-domain(1), stop-domain(1), list-domains(1)
asadmin(1M)
Apache Felix Remote Shell
(http://felix.apache.org/site/apache-felix-remote-shell.html), Java Platform
Debugger Architecture (JPDA)
(http://java.sun.com/javase/technologies/core/toolsapis/jpda/)
create-file-user

creates a new file user

Synopsis

create-file-user [--help] [--authrealmname auth_realm_name]
|--target target
|--groups user_groups[:user_groups]*

Description

The create-file-user subcommand creates an entry in the keyfile with the specified
username, password, and groups. Multiple groups can be created by separating them
with a colon (:). If auth_realm_name is not specified, an entry is created in the keyfile
for the default realm. If auth_realm_name is specified, an entry is created in the keyfile
using the auth_realm_name.

You can use the --passwordfile option of the asadmin(1M) command to specify the
password for the user. The password file entry must be of the form AS_ADMIN_USERPASSWORD=user-password.

If a password is not provided, this subcommand fails if secure administration is
enabled and the user being created is an administrative user.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This is the name of the target on which the command operates. The valid targets are
config, instance, cluster, or server. By default, the target is the server.

--groups
This is the group associated with this file user.

--authrealmname
The name of the realm in which the new user is created. If you do not specify this
option, the user is created in the "file" realm.

Operands

user_name
This is the name of file user to be created.
Examples

Example 1 Creating a User in the File Realm
This example creates a file realm user named sample_user. It is assumed that an authentication realm has already been created using the create-auth-realm subcommand.

```
asadmin> create-file-user
--groups staff:manager
--authrealmname auth-realm1 sample_user
Command create-file-user executed successfully
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-auth-realm(1), delete-file-user(1), list-file-users(1),
update-file-user(1), list-file-groups(1)
asadmin(1M)
create-http

sets HTTP parameters for a protocol

Synopsis

cREATE-HTTP  [--help]
--default-virtual-server  virtual-server
[--request-timeout-seconds  timeout]
[--timeout-seconds  timeout]
[--max-connection  max-keepalive]
[--dns-lookup-enabled={false|true}]
[--servername  server-name]
[--target  target]
protocol-name

Description

The create-http subcommand creates a set of HTTP parameters for a protocol, which
in turn configures one or more network listeners. This subcommand is supported in
remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--default-virtual-server
The ID attribute of the default virtual server for the associated network listeners.

--request-timeout-seconds
The time in seconds at which the request times out. If you do not set this option, the
request times out in 30 seconds.

--timeout-seconds
The maximum time in seconds for which a keep alive connection is kept open. A value
of 0 or less means keep alive connections are kept open indefinitely. The default is 30.

--max-connection
The maximum number of HTTP requests that can be pipelined until the connection is
closed by the server. Set this property to 1 to disable HTTP/1.0 keep-alive, as well as
HTTP/1.1 keep-alive and pipelining. The default is 256.

--dns-lookup-enabled
If set to true, looks up the DNS entry for the client. The default is false.

--servername
Tells the server what to put in the host name section of any URLs it sends to the client.
This affects URLs the server automatically generates; it doesn't affect the URLs for
directories and files stored in the server. This name should be the alias name if your
server uses an alias. If a colon and port number are appended, that port will be used in
URLs that the server sends to the client.
create-http

--target
Creates the set of HTTP parameters only on the specified target. Valid values are as follows:

server
Creates the set of HTTP parameters on the default server instance. This is the default value.

configuration-name
Creates the set of HTTP parameters in the specified configuration.

cluster-name
Creates the set of HTTP parameters on all server instances in the specified cluster.

standalone-instance-name
Creates the set of HTTP parameters on the specified standalone server instance.

Operands

protocol-name
The name of the protocol to which this HTTP parameter set applies.

Examples

Example 1 Using the create-http Subcommand
The following command creates an HTTP parameter set for the protocol named http-1:

asadmin> create-http --timeout-seconds 60 --default-virtual-server server http-1
Command create-http executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-http(1), create-network-listener(1), create-protocol(1),
create-virtual-server(1)
asadmin(1M)
create-http-health-checker

creates a health-checker for a specified load balancer configuration

Synopsis

create-http-health-checker [--help] [--url "/"]
[--interval 30] [--timeout 10]
[--config config_name] target

Description

The create-http-health-checker subcommand creates a health checker for a specified load balancer configuration. A health checker is unique for the combination of target and load balancer configuration.

Note: This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

Options

--help
-?
Displays the help text for the subcommand.

--url
The URL to ping to determine whether the instance is healthy.

--interval
The interval in seconds the health checker waits between checks of an unhealthy instance to see whether it has become healthy. The default value is 30 seconds. A value of 0 disables the health checker.

--timeout
The interval in seconds the health checker waits to receive a response from an instance. If the health checker has not received a response in this interval, the instance is considered unhealthy.

--config
The load balancer configuration for which you create the health-checker. If you do not specify a configuration, the subcommand creates a health checker for every load balancer configuration associated with the target. If no configuration references the target, the subcommand fails.

Operands

target
Specifies the target to which the health checker applies.

Valid values are:
- `cluster_name`- The name of a target cluster.
- `instance_name`- The name of a target server instance.

**Examples**

**Example 1 Creating a Health Checker for a Load Balancer Configuration**

This example creates a health checker for a load balancer configuration named `mycluster-http-lb-config` on a cluster named `mycluster`.

```bash
asadmin> create-http-health-checker --config mycluster-http-lb-config mycluster
```

Command `create-http-health-checker` executed successfully.

**Exit Status**

- 0: subcommand executed successfully
- 1: error in executing the subcommand

**See Also**

- `delete-http-health-checker(1)`
- `asadmin(1M)`
create-http-lb

creates a load balancer

Synopsis

create-http-lb [-help] --devicehost device_host_or_IP_address --deviceport device_port
[--sslproxyhost proxy_host]
[--sslproxyport proxy_port] [--target target] [--lbpolicy lbpolicy]
[--lbpolicymodule lb_policy_module] [--healthcheckerurl url]
[--healthcheckerinterval 10] [--healthcheckertimeout 10]
[--lbenableallinstances=true] [--lbenableallapplications=true] [--lbweight
instance=weight:['instance=weight']*) [--responsetimeout 60] [--httpsrouting=false]
[--reloadinterval60] [--monitor=false] [--routecookie=true]
[--property (name=value)['name=value']*)
] load_balancer_name

Description

Use the create-http-lb subcommand to create a load balancer, including the load balancer configuration, target reference, and health checker. A load balancer is a representation of the actual load balancer device, defined by its device host and port information. Once you’ve created the load balancer, you can automatically apply changes made to the load balancer configuration without running export-http-lb-config and manually copying the generated load balancer configuration file to the web server instance.

Note: This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

Options

--help
-?
Displays the help text for the subcommand.

--devicehost
The device host or the IP address of the load balancing device. This host or IP is where the physical load balancer will reside.

--deviceport
The port used to communicate with the load balancing device. It must be SSL enabled.

--sslproxyhost
The proxy host used for outbound HTTP.

--sslproxyport
The proxy port used for outbound HTTP.
--target
Specifies the target to which the load balancer applies.
Valid values are:

- cluster_name- Specifies that requests for this cluster will be handled by the load balancer.
- stand-alone_instance_name- Specifies that requests for this stand-alone instance will be handled by the load balancer.

--lbpolicy
The policy the load balancer follows to distribute load to the server instances in a cluster. Valid values are round-robin, weighted-round-robin, and user-defined. If you choose user-defined, specify a load balancer policy module with the lbpolicymodule option. If you choose weighted-round-robin, assign weights to the server instances using the configure-lb-weight subcommand. The default is round-robin.

--lbpolicymodule
If your target is a cluster and the load balancer policy is user-defined, use this option to specify the full path and name of the shared library of your load balancing policy module. The shared library needs to be in a location accessible by the web server.

--healthcheckerurl
The URL to ping to determine whether the instance is healthy.

--healthcheckerinterval
The interval in seconds the health checker waits between checks of an unhealthy instance to see whether it has become healthy. The default value is 10 seconds. A value of 0 disables the health checker.

--healthcheckertimeout
The interval in seconds the health checker waits to receive a response from an instance. If the health checker has not received a response in this interval, the instance is considered unhealthy. The default value is 10 seconds.

--lbenableallinstances
Enables all instances in the target cluster for load balancing. If the target is a server instance, enables that instance for load balancing.

--lbenableallapplications
Enables all applications deployed to the target cluster or instance for load balancing.

--lbweight
The name of the instance and the weight you are assigning it. The weight must be an integer. The pairs of instances and weights are separated by colons. For example instance1=1:instance2=4 means that for every five requests, one goes to instance1 and four go to instance2. A weight of 1 is the default.

--responsetimeout
The time in seconds within which a server instance must return a response. If no response is received within the time period, the server is considered unhealthy. If set to a positive number, and the request is idempotent, the request is retried. If the request is not idempotent, an error page is returned. If set to 0 no timeout is used. The default is 60.
create-http-lb

--httpsrouting
If set to true, HTTPS requests to the load balancer result in HTTPS requests to the server instance. If set to false, HTTPS requests to the load balancer result in HTTP requests to the server instance. The default is false.

--reloadinterval
The time, in seconds, that the load balancer takes to check for an updated configuration. When detected, the configuration file is reloaded. The default value is 60 seconds. A value of 0 disables reloading.

--monitor
If set to true, monitoring of the load balancer is switched on. The default value is false.

--routecookie
This option is deprecated. The value is always true.

--property
Optional attribute name/value pairs for configuring the load balancer.

Operands

lb_name
The name of the new load balancer. This name must not conflict with any other load balancers in the domain.

Examples

Example 1 Creating a Load Balancer
This example creates a load balancer named mylb.

asadmin> create-http-lb
--devicehost host1 --deviceport 5555 mylb

Command create-http-lb executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-http-lb(1), list-http-lbs(1), create-http-lb-config(1)
asadmin(1M)
create-http-lb-config

creates a configuration for the load balancer

Synopsis

create-http-lb-config [--help] [--responsetimeout 60] [httpsrouting=false] [--reloadinterval 60] [--monitor=false] [--property (name=value)[,name=value]*] --target target | config_name

Description

Use the create-http-lb-config subcommand to create a load balancer configuration. This configuration applies to load balancing in the HTTP path. After using this subcommand to create the load balancer configuration file, create the load balancer by running create-http-lb.

You must specify either a target or a configuration name, or both. If you do not specify a target, the configuration is created without a target and you add one later using create-http-lb-ref. If you don’t specify a configuration name, a name is created based on the target name. If you specify both, the configuration is created with the specified name, referencing the specified target.

Note: This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

Options

--help
-?
Displays the help text for the subcommand.

--responsetimeout
The time in seconds within which a server instance must return a response. If no response is received within the time period, the server is considered unhealthy. If set to a positive number, and the request is idempotent, the request is retried. If the request is not idempotent, an error page is returned. If set to 0 no timeout is used. The default is 60.

--httpsrouting
If set to true, HTTPS requests to the load balancer result in HTTPS requests to the server instance. If set to false, HTTPS requests to the load balancer result in HTTP requests to the server instance. The default is false.

--reloadinterval
The interval between checks for changes to the load balancer configuration file loadbalancer.xml. When the check detects changes, the configuration file is reloaded. A value of 0 disables reloading.
--monitor
Specifies whether monitoring is enabled. The default is false.

--routecookie
This option is deprecated. The value is always true.

--property
Optional attribute name/value pairs for configuring the load balancer.

--target
Specifies the target to which the load balancer configuration applies. If you don't specify a target, the load balancer configuration is created without a target. You can specify targets later using the subcommand create-http-lb-ref.

Valid values are:

- `cluster_name` - Specifies that requests for this cluster will be handled by the load balancer.
- `stand-alone_instance_name` - Specifies that requests for this standalone instance will be handled by the load balancer.

**Operands**

*config_name*

The name of the new load balancer configuration. This name must not conflict with any other load balancer groups, agents, configurations, clusters, or sever instances in the domain. If you don’t specify a name, the load balancer configuration name is based on the target name, `target_name-http-lb-config`.

**Examples**

**Example 1 Creating a Load Balancer Configuration**

This example creates a load balancer configuration on a target named `mycluster` and load balancer configuration named `mylbconfigname`.

```
asadmin> create-http-lb-config --target mycluster mylbconfigname
```

Command create-http-lb-config executed successfully.

**Exit Status**

0
subcommand executed successfully

1
error in executing the subcommand

**See Also**

*delete-http-lb-config(1), list-http-lb-configs(1), create-http-lb(1), asadmin(1M)*
create-http-lb-ref

adds an existing cluster or server instance to an existing load balancer configuration or load balancer

Synopsis

create-http-lb-ref [--help] --config config_name | --lbname load_balancer_name

[-lbpolicy round-robin] [-lbpolicymodule lb_policy_module]
[-healthcheckerurl url] [-healthcheckereinterval 10]
[-healthcheckertimeout 10] [-lbenableallinstances=true]
[-lbenableallapplications=true] [-lbweight instance=weight[:instance=weight]*]
target

Description

Use the create-http-lb-ref subcommand to:

- Add an existing cluster or server instance to an existing load balancer configuration or load balancer. The load balancer forwards the requests to the clustered and standalone instances it references.
- Set the load balancing policy to round-robin, weighted round-robin, or to a user-defined policy.
- Configure a health checker for the load balancer. Any health checker settings defined here apply only to the target. If you do not create a health checker with this subcommand, use create-http-health-checker.
- Enable all instances in the target cluster for load balancing, or use enable-http-lb-server to enable them individually.
- Enable all applications deployed to the target for load balancing, or use enable-http-lb-application to enable them individually.

Note: This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

Options

--help
-?
Displays the help text for the subcommand.

--config
Specifies which load balancer configuration to which to add clusters and server instances. Specify either a load balancer configuration or a load balancer. Specifying both results in an error.

--lbname
Specifies the load balancer to which to add clusters and server instances. Specify either a load balancer configuration or a load balancer. Specifying both results in an error.
create-http-lb-ref

--lbpolicy
The policy the load balancer follows. Valid values are round-robin, weighted-round-robin, and user-defined. If you choose user-defined, specify a load balancer policy module with the lbpolicymodule option. If you choose weighted-round-robin assign weights to the server instances using the configure-lb-weight subcommand. The default is round-robin.

--lbpolicymodule
If your load balancer policy is user-defined, use this option to specify the full path and name of the shared library of your load balancing policy module. The shared library needs to be in a location accessible by the web server.

--healthcheckerurl
The URL to ping to determine whether the instance is healthy.

--healthcheckerinterval
The interval in seconds the health checker waits between checks of an unhealthy instance to see whether it has become healthy. The default value is 30 seconds. A value of 0 disables the health checker.

--healthcheckertimeout
The interval in seconds the health checker waits to receive a response from an instance. If the health checker has not received a response in this interval, the instance is considered unhealthy. The default is 10.

--lbenableallinstances
Enables all instances in the target cluster for load balancing. If the target is a server instance, enables that instance for load balancing. The default value is true.

--lbenableallapplications
Enables all applications deployed to the target cluster or instance for load balancing. The default value is true.

--lbweight
The name of the instance and the weight you are assigning it. The weight must be an integer. The pairs of instances and weights are separated by colons. For example instance1=1:instance2=4 means that for every five requests, one goes to instance1 and four go to instance2. A weight of 1 is the default.

Operands

target
Specifies which cluster or instance to add to the load balancer. Valid values are:

- cluster_name- Specifies that requests for this cluster will be handled by the load balancer.
- stand-alone_instance_name- Specifies that requests for this standalone instance will be handled by the load balancer.

Examples

Example 1 Adding a Cluster Reference to a Load Balancer Configuration
This example adds a reference to a cluster named cluster2 to a load balancer configuration named mylbconfig.

asadmin> create-http-lb-ref --config mylbconfig cluster2
Command create-http-lb-ref executed successfully.

**Example 2  Adding a Cluster Reference to a Load Balancer**
This example adds a reference to a cluster named `cluster2` to a load balancer named `mylb`.

```
asadmin> create-http-lb-ref --lbname mylb cluster2
```

Command create-http-lb-ref executed successfully.

**Example 3  Configuring a Health Checker and Load Balancer Policy**
This example configures a health checker and load balancing policy, and enables the load balancer for instances and applications.

```
asadmin> create-http-lb-ref --config mylbconfig --lbpolicy weighted-round-robin --healthcheckinterval 40 --healthchecktimeout 20 --lbenableallinstances=true --lbenableallapplications=true cluster2
```

Command create-http-lb-ref executed successfully.

**Example 4  Setting a User-Defined Load Balancing Policy**
This example sets a user-defined load balancing policy.

```
asadmin> create-http-lb-ref --lbpolicy user-defined --lbpolicymodule /user/modules/module.so --config mylbconfig cluster2
```

Command create-http-lb-ref executed successfully.

**Exit Status**

- **0**
  subcommand executed successfully

- **1**
  error in executing the subcommand

**See Also**

- delete-http-lb-ref(1)
- create-http-health-checker(1)
- enable-http-lb-server(1)
- enable-http-lb-application(1)
- list-http-lb-configs(1)
- list-http-lbs(1)
- configure-lb-weight(1)
- asadmin(1M)
create-http-listener

adds a new HTTP network listener socket

Synopsis

create-http-listener [--help] --listeneraddress address
--listenerport listener-port
{--default-virtual-server | --defaultvs} virtual-server
[--servername server-name]
[--acceptor-threads acceptor-threads]
[--xpowered={true|false}]
[--redirectport redirect-port]
[--securityenabled={false|true}]
[--enabled={true|false}]
[--target target]
listener-id

Description

The create-http-listener subcommand creates an HTTP network listener. This subcommand is supported in remote mode only.

**Note:** If you edit the special HTTP network listener named admin-listener, you must restart the server for the changes to take effect. The Administration Console does not tell you that a restart is required in this case.

**Note:** This subcommand is provided for backward compatibility and as a shortcut for creating network listeners that use the HTTP protocol. Behind the scenes, this subcommand creates a network listener and its associated protocol, transport, and HTTP configuration.

Options

**--help**
-?
Displays the help text for the subcommand.

**--listeneraddress**
The IP address or the hostname (resolvable by DNS).

**--listenerport**
The port number to create the listen socket on. Legal values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges. Configuring an SSL listen socket to listen on port 443 is recommended.
create-http-listener

--default-virtual-server
--defaultvs
The ID attribute of the default virtual server for this listener. The --defaultvs option is deprecated.

--servername
Tells the server what to put in the host name section of any URLs it sends to the client. This affects URLs the server automatically generates; it doesn’t affect the URLs for directories and files stored in the server. This name should be the alias name if your server uses an alias. If a colon and port number are appended, that port will be used in URLs that the server sends to the client.

--acceptorthreads
The number of acceptor threads for the listener socket. The recommended value is the number of processors in the machine. The default value is 1.

--xpowered
If set to true, adds the X-Powered-By: Servlet/3.0 and X-Powered-By: JSP/2.0 headers to the appropriate responses. The Servlet 3.0 specification defines the X-Powered-By: Servlet/3.0 header, which containers may add to servlet-generated responses. Similarly, the JSP 2.0 specification defines the X-Powered-By: JSP/2.0 header, which containers may add to responses that use JSP technology. The goal of these headers is to aid in gathering statistical data about the use of Servlet and JSP technology. The default value is true.

--redirectport
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

--securityenabled
If set to true, the HTTP listener runs SSL. You can turn SSL2 or SSL3 ON or OFF and set ciphers using an SSL element. The security setting globally enables or disables SSL by making certificates available to the server instance. The default value is false.

--enabled
If set to true, the listener is enabled at runtime. The default value is true.

--target
Creates the HTTP listener only on the specified target. Valid values are as follows:

server
Creates the HTTP listener on the default server instance. This is the default value.

configuration-name
Creates the HTTP listener in the specified configuration.

cluster-name
Creates the HTTP listener on all server instances in the specified cluster.

standalone-instance-name
Creates the HTTP listener on the specified standalone server instance.
create-http-listener

Operands

listener-id
The listener ID of the HTTP network listener.

Examples

Example 1 Creating an HTTP Network Listener
The following command creates an HTTP network listener named sampleListener that uses a nondefault number of acceptor threads and is not enabled at runtime:

asadmin> create-http-listener --listeneraddress 0.0.0.0 --listenerport 7272 --defaultvs server --servername host1.sun.com --acceptorthreads 100 --securityenabled=false --enabled=false sampleListener
Command create-http-listener executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-http-listener(1), list-http-listeners(1), create-virtual-server(1), create-ssl(1), create-network-listener(1)

asadmin(1M)
create-http-redirect

adds a new HTTP redirect

Synopsis

create-http-redirect [--help]
[--redirect-port redirect-port]
[--secure-redirect={false|true}]
[--target target]
protocol-name

Description

The create-http-redirect subcommand creates an HTTP redirect. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--redirect-port
Port number for redirects. If the HTTP listener is supporting non-SSL requests, and a request is received for which a matching security-constraint requires SSL transport, GlassFish Server automatically redirects the request to this port number.

--secure-redirect
If set to true, the HTTP redirect runs SSL. The default value is false.

--target
Creates the HTTP redirect only on the specified target. Valid values are as follows:

server
Creates the HTTP redirect on the default server instance. This is the default value.

configuration-name
Creates the HTTP redirect in the specified configuration.

cluster-name
Creates the HTTP redirect on all server instances in the specified cluster.

standalone-instance-name
Creates the HTTP redirect on the specified standalone server instance.

Operands

protocol-name
The name of the protocol to which to apply the redirect.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-http-redirect(1)
asadmin(1M)
create-iiop-listener

adds an IIOP listener

Synopsis

create-iiop-listener [--help] --listeneraddress address
[--iiopport iiop-port-number] [--securityenabled={false|true}]
[--enabled={true|false}]
[--property {name=value} [:name=value]*]
[--target target] listener_id

Description

The create-iiop-listener subcommand creates an IIOP listener. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--listeneraddress
Either the IP address or the hostname (resolvable by DNS).

--iiopport
The IIOP port number. The default value is 1072.

--securityenabled
If set to true, the IIOP listener runs SSL. You can turn SSL2 or SSL3 ON or OFF and set ciphers using an SSL element. The security setting globally enables or disables SSL by making certificates available to the server instance. The default value is false.

--enabled
If set to true, the IIOP listener is enabled at runtime. The default value is true.

--property
Optional attribute name/value pairs for configuring the IIOP listener.

--target
Specifies the target for which you are creating the IIOP listener. Valid values are

server
Creates the listener for the default server instance server and is the default value.

configuration_name
Creates the listener for the named configuration.

cluster_name
Creates the listener for every server instance in the cluster.

stand-alone_instance_name
Creates the listener for a particular standalone server instance.
Operands

`listener_id`
A unique identifier for the IIOP listener to be created.

Examples

**Example 1 Creating an IIOP Listener**
The following command creates an IIOP listener named `sample_iiop_listener`:

```bash
asadmin> create-iiop-listener --listeneraddress 192.168.1.100
--iiopport 1400 sample_iiop_listener
Command create-iiop-listener executed successfully.
```

**Example 2 Creating an IIOP Listener with a Target Cluster**
The following command creates an IIOP listener named `iiop_listener_2` for the cluster `mycluster`. It uses the target option.

```bash
asadmin> create-iiop-listener --listenertaddress 0.0.0.0 --iioport 1401
--target mycluster iiop_listener_2
Command create-iiop-listener executed successfully.
```

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delte-iiop-listener(1), list-iiop-listeners(1), create-ssl(1)
asadmin(1M)
create-instance

creates a GlassFish Server instance

Synopsis

create-instance [--help] --node node-name
[--config config-name | --cluster cluster-name]
[--lbenabled={true|false}]
[--portbase=port-number] [--checkports={true|false}]
[--systemproperties (name=value)[|name=value]*]

instance-name

description

The create-instance subcommand creates a GlassFish Server instance. This subcommand requires the Distributed Component Object Model (DCOM) remote protocol or secure shell (SSH) to be configured on the host where the domain administration server (DAS) is running and on the host that is represented by the node where the instance is to reside.

Note: DCOM or SSH is not required if the instance is to reside on a node of type CONFIG that represents the local host. A node of type CONFIG is not enabled for remote communication over DCOM or SSH.

You may run this command from any host that can contact the DAS.

A GlassFish Server instance is a single Virtual Machine for the Java platform (Java Virtual Machine or JVM machine) on a single node in which GlassFish Server is running. A node defines the host where the GlassFish Server instance resides. The JVM machine must be compatible with the Java Platform, Enterprise Edition (Java EE).

A GlassFish Server instance requires a reference to the following items:

- The node that defines the host where the instance resides. The node must be specified in the command to create the instance.
- The named configuration that defines the configuration of the instance. The configuration can be specified in the command to create the instance, but is not required. If no configuration is specified for an instance that is not joining a cluster, the subcommand creates a configuration for the instance. An instance that is joining a cluster receives its configuration from its parent cluster.

Each GlassFish Server instance is one of the following types of instance:

Standalone instance

A standalone instance does not share its configuration with any other instances or clusters. A standalone instance is created if either of the following conditions is met:

- No configuration or cluster is specified in the command to create the instance.
- A configuration that is not referenced by any other instances or clusters is specified in the command to create the instance.
When no configuration or cluster is specified, a copy of the `default-config` configuration is created for the instance. The name of this configuration is `instance-name-config`, where `instance-name` represents the name of an unclustered server instance.

**Shared instance**
A shared instance shares its configuration with other instances or clusters. A shared instance is created if a configuration that is referenced by other instances or clusters is specified in the command to create the instance.

**Clustered instance**
A clustered instance inherits its configuration from the cluster to which the instance belongs and shares its configuration with other instances in the cluster. A clustered instance is created if a cluster is specified in the command to create the instance.

Any instance that is not part of a cluster is considered an unclustered server instance. Therefore, standalone instances and shared instances are unclustered server instances.

By default, this subcommand attempts to resolve possible port conflicts for the instance that is being created. The subcommand also assigns ports that are currently not in use and not already assigned to other instances on the same node. The subcommand assigns these ports on the basis of an algorithm that is internal to the subcommand. Use the `--systemproperties` option to resolve port conflicts for additional instances on the same node. System properties of an instance can be manipulated by using the `create-system-properties(1)` subcommand and the `delete-system-property(1)` subcommand.

This subcommand is supported in remote mode only.

**Options**

`--help`  
Displays the help text for the subcommand.

`--node`
The name of the node that defines the host where the instance is to be created. The node must already exist. If the instance is to be created on the host where the domain administration server (DAS) is running, use the predefined node `localhost-domain`.

`--config`
Specifies the named configuration that the instance references. The configuration must exist and must not be named `default-config` or `server-config`. Specifying the `--config` option creates a shared instance.

The `--config` option and the `--cluster` option are mutually exclusive. If both options are omitted, a standalone instance is created.

`--cluster`
Specifies the cluster from which the instance inherits its configuration. Specifying the `--cluster` option creates a clustered instance.

The `--config` option and the `--cluster` option are mutually exclusive. If both options are omitted, a standalone instance is created.

`--lbenabled`
Specifies whether the instance is enabled for load balancing. Possible values are as follows:
true
The instance is enabled for load balancing (default).
When an instance is enabled for load balancing, a load balancer sends requests to
the instance.

false
The instance is disabled for load balancing.
When an instance is disabled for load balancing, a load balancer does not send
requests to the instance.

--portbase
Determines the number with which the port assignment should start. An instance uses
a certain number of ports that are statically assigned. The portbase value determines
where the assignment should start. The values for the ports are calculated as follows:

- Administration port: portbase + 48
- HTTP listener port: portbase + 80
- HTTPS listener port: portbase + 81
- JMS port: portbase + 76
- IIOP listener port: portbase + 37
- Secure IIOP listener port: portbase + 38
- Secure IIOP with mutual authentication port: portbase + 39
- JMX port: portbase + 86
- JPA debugger port: portbase + 9
- Felix shell service port for OSGi module management: portbase + 66

When the --portbase option is specified, the output of this subcommand includes a
complete list of used ports.

--checkports
Specifies whether to check for the availability of the administration, HTTP, JMS, JMX,
and IIOP ports. The default value is true.

--systemproperties
Defines system properties for the instance. These properties override property
definitions for port settings in the instance’s configuration. Predefined port settings
must be overridden if, for example, two clustered instances reside on the same host. In
this situation, port settings for one instance must be overridden because both instances
share the same configuration.

The following properties are available:

**ASADMIN_LISTENER_PORT**
This property specifies the port number of the HTTP port or HTTPS port through
which the DAS connects to the instance to manage the instance. Valid values are
1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser
privileges.

**HTTP_LISTENER_PORT**
This property specifies the port number of the port that is used to listen for HTTP
requests. Valid values are 1-65535. On UNIX, creating sockets that listen on ports
1-1024 requires superuser privileges.
HTTP_SSL_LISTENER_PORT
This property specifies the port number of the port that is used to listen for HTTPS requests. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

IIOP_LISTENER_PORT
This property specifies the port number of the port that is used for IIOP connections. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

IIOP_SSL_LISTENER_PORT
This property specifies the port number of the port that is used for secure IIOP connections. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

IIOP_SSL_MUTUALAUTH_PORT
This property specifies the port number of the port that is used for secure IIOP connections with client authentication. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

JAVA_DEBUGGER_PORT
This property specifies the port number of the port that is used for connections to the Java Platform Debugger Architecture (JPDA) ([http://java.sun.com/javase/technologies/core/toolsapis/jpda/](http://java.sun.com/javase/technologies/core/toolsapis/jpda/)) debugger. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

JMS_PROVIDER_PORT
This property specifies the port number for the Java Message Service provider. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

JMX_SYSTEM_CONNECTOR_PORT
This property specifies the port number on which the JMX connector listens. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

OSGI_SHELL_TELNET_PORT
This property specifies the port number of the port that is used for connections to the Apache Felix Remote Shell ([http://felix.apache.org/site/apache-felix-remote-shell.html](http://felix.apache.org/site/apache-felix-remote-shell.html)). This shell uses the Felix shell service to interact with the OSGi module management subsystem. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

Operands

`instance-name`
The name of the instance that is being created.

The name must meet the following requirements:

- The name may contain only ASCII characters.
- The name must start with a letter, a number, or an underscore.
- The name may contain only the following characters:
  - Lowercase letters
- Uppercase letters
- Numbers
- Hyphen
- Period
- Underscore

- The name must be unique in the domain and must not be the name of another GlassFish Server instance, a cluster, a named configuration, or a node.
- The name must not be domain, server, or any other keyword that is reserved by GlassFish Server.

Examples

Example 1 Creating a Standalone GlassFish Server Instance
This example creates the standalone GlassFish Server instance pmdsainst in the domain domain1 on the local host.

```
asadmin> create-instance --node localhost-domain1 pmdsainst
Port Assignments for server instance pmdsainst:
JMX_SYSTEM_CONNECTOR_PORT=28688
JMS_PROVIDER_PORT=27678
ASADMIN_LISTENER_PORT=24850
HTTP_LISTENER_PORT=28082
IIOP_LISTENER_PORT=23702
IIOP_SSL_LISTENER_PORT=23822
HTTP_SSL_LISTENER_PORT=28183
IIOP_SSL_MUTUALAUTH_PORT=23922
Command create-instance executed successfully.
```

Example 2 Creating a Standalone GlassFish Server Instance With Custom Port Assignments
This example creates the standalone GlassFish Server instance pmdcpinst in the domain domain1 on the local host. Custom port numbers are assigned to the following ports:

- HTTP listener port
- HTTPS listener port
- IIOP connections port
- Secure IIOP connections port
- Secure IIOP connections port with mutual authentication
- JMX connector port

```
asadmin> create-instance --node localhost-domain1
--systemproperties HTTP_LISTENER_PORT=58294:
HTTP_SSL_LISTENER_PORT=58297:
IIOP_LISTENER_PORT=58300:
IIOP_SSL_LISTENER_PORT=58303:
IIOP_SSL_MUTUALAUTH_PORT=58306:
JMX_SYSTEM_CONNECTOR_PORT=58309 pmdcpinst
Port Assignments for server instance pmdcpinst:
JMS_PROVIDER_PORT=27679
```

-7 Oracle GlassFish Server 3.1.2 Reference Manual
Example 3  Creating a Shared GlassFish Server Instance
This example creates the shared GlassFish Server instance pmdsharedinst1 in the domain domain1 on the local host. The shared configuration of this instance is pmdsharedconfig.

```
asadmin create-instance --node localhost-domain1 --config pmdsharedconfig pmdsharedinst1
```

Port Assignments for server instance pmdsharedinst1:
- JMX_SYSTEM_CONNECTOR_PORT=28687
- JMS_PROVIDER_PORT=27677
- ASADMIN_LISTENER_PORT=24849
- HTTP_LISTENER_PORT=28081
- IIOP_LISTENER_PORT=23701
- IIOP_SSL_LISTENER_PORT=23821
- HTTP_SSL_LISTENER_PORT=28182
- IIOP_SSL_MUTUALAUTH_PORT=23921

Command create-instance executed successfully.

Example 4  Creating a Clustered GlassFish Server Instance
This example creates the clustered GlassFish Server instance pmdinst1 in the domain domain1 on the local host. The instance is a member of the cluster pmdclust1.

```
asadmin> create-instance --node localhost-domain1 --cluster pmdclust pmdinst1
```

Port Assignments for server instance pmdinst1:
- JMX_SYSTEM_CONNECTOR_PORT=28686
- JMS_PROVIDER_PORT=27676
- HTTP_LISTENER_PORT=28080
- ASADMIN_LISTENER_PORT=24848
- IIOP_LISTENER_PORT=23700
- IIOP_SSL_LISTENER_PORT=23820
- HTTP_SSL_LISTENER_PORT=28181
- IIOP_SSL_MUTUALAUTH_PORT=23920

Command create-instance executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-local-instance(1), create-node-config(1), create-node-dcom(1), create-node-ssh(1), create-system-properties(1), delete-instance(1), delete-system-property(1), list-instances(1), setup-ssh(1), start-instance(1), stop-instance(1)
asadmin(1M)
create-jacc-provider

enables administrators to create a JACC provider that can be used by third-party authorization modules for applications running in GlassFish Server

Synopsis

create-jacc-provider [--help]
[--policyproviderclass pol-provider-class]
[--policyconfigfactoryclass pc-factory-class]
[--property name=value][:name=value]*
[--target target] jacc-provider-name

Description

The create-jacc-provider subcommand creates a JSR-115—compliant Java Authorization Contract for Containers (JACC) provider that can be used for authorization of applications running in GlassFish Server. The JACC provider is created as a jacc-provider element within the security-service element in the domain's domain.xml file.

The default GlassFish Server installation includes two JACC providers, named default and simple. Any JACC providers created with the create-jacc-provider subcommand are in addition to these two default providers. The default GlassFish Server JACC providers implement a simple, file-based authorization engine that complies with the JACC specification. The create-jacc-provider subcommand makes it possible to specify additional third-party JACC providers.

You can create any number of JACC providers within the security-service element, but the GlassFish Server runtime uses only one of them at any given time. The jacc-provider element in the security-service element points to the name of the provider that is currently in use by GlassFish Server. If you change this element to point to a different JACC provider, restart GlassFish Server.

This command is supported in remote mode only.

Options

If an option has a short option name, then the short option precedes the long option name. Short options have one dash whereas long options have two dashes.

--help
-
? Displays the help text for the subcommand.

--policyproviderclass
Specifies the fully qualified class name for the java.security.jacc.policy.provider that implements the java.security.Policy.

--policyconfigfactoryclass
Specifies the fully qualified class name for the java.security.jacc.PolicyConfigurationFactory.provider that implements the provider-specific java.security.jacc.PolicyConfigurationFactory.
--property
Optional attribute name/value pairs for configuring the JACC provider. The following properties are available:

repository
The directory containing the JACC policy file. For the default GlassFish Server JACC provider, the default directory is ${com.sun.aas.instanceRoot}/generated/policy. This property is not defined by default for the simple GlassFish Server JACC provider.

--target
Specifies the target for which you are creating the JACC provider. The following values are valid:

server
Creates the JACC provider on the default server instance. This is the default value.

configuration_name
Creates the JACC provider in the specified configuration.

cluster_name
Creates the JACC provider on all server instances in the specified cluster.

instance_name
Creates the JACC provider on a specified server instance.

Operands

jacc-provider-name
The name of the provider used to reference the jacc-provider element in domain.xml.

Examples

Example 1 Creating a JACC Provider
The following example shows how to create a JACC provider named testJACC on the default server target.

asadmin> create-jacc-provider
--policyproviderclass com.sun.enterprise.security.provider.PolicyWrapper
--policyconfigfactoryclass com.sun.enterprise.security.provider.PolicyConfigurationFactoryImpl
testJACC

Command create-jacc-provider executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-jacc-provider(1), list-jacc-providers(1)
asadmin(IM)
create-javamail-resource

creates a JavaMail session resource

Synopsis

create-javamail-resource [--help] [--target target] --mailhost hostname
--mailuser username --fromaddress address [--storeprotocol protocol] 
[<storeprotocolclass protocolclass>] [--transprotocol protocol] 
[<transprotocolclass protocolclass>] [--debug={false|true}] 
[<enabled={true|false}>] 
[<description resource-description>] [--property (name=value)[::name=value]*] 
jndi-name

Description

The create-javamail-resource subcommand creates a JavaMail session resource. 
This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target for which you are creating the JavaMail session
resource. Valid values are:

server
Creates the resource for the default server instance. This is the default value.

domain
Creates the resource for the domain.

cluster_name
Creates the resource for every server instance in the cluster.

instance_name
Creates the resource for a particular server instance.

--mailhost
The DNS name of the default mail server. The connect methods of the Store and
Transport objects use this value if a protocol-specific host property is not supplied. The
name must be resolvable to an actual host name.

--mailuser
The name of the mail account user provided when connecting to a mail server. The
connect methods of the Store and Transport objects use this value if a protocol-specific
username property is not supplied.

--fromaddress
The email address of the default user, in the form username@host.domain.
create-javamail-resource

--storeprotocol
The mail server store protocol. The default is imap. Change this value only if you have reconfigured the GlassFish Server's mail provider to use a non-default store protocol.

--storeprotocolclass
The mail server store protocol class name. The default is com.sun.mail.imap.IMAPStore. Change this value only if you have reconfigured the GlassFish Server's mail provider to use a nondefault store protocol.

--transprotocol
The mail server transport protocol. The default is smtp. Change this value only if you have reconfigured the GlassFish Server's mail provider to use a nondefault transport protocol.

--transprotocolclass
The mail server transport protocol class name. The default is com.sun.mail.smtp.SMTPTransport. Change this value only if you have reconfigured the GlassFish Server's mail provider to use a nondefault transport protocol.

--debug
If set to true, the server starts up in debug mode for this resource. If the JavaMail log level is set to FINE or FINER, the debugging output will be generated and will be included in the server log file. The default value is false.

--enabled
If set to true, the resource is enabled at runtime. The default value is true.

--description
Text providing some details of the JavaMail resource.

--property
Optional attribute name/value pairs for configuring the JavaMail resource. The GlassFish Server-specific mail- prefix is converted to the standard mail prefix. The JavaMail API documentation lists the properties you might want to set.

Operands

jndi-name
The JNDI name of the JavaMail resource to be created. It is a recommended practice to use the naming subcontext prefix mail/ for JavaMail resources.

Examples

Example 1 Creating a JavaMail Resource
This example creates a JavaMail resource named mail/MyMailSession. The JNDI name for a JavaMail session resource customarily includes the mail/ naming subcontext.

asadmin> create-javamail-resource --mailhost localhost
--mailuser sample --fromaddress sample@sun.com mail/MyMailSession
Command create-javamail-resource executed successfully.

Exit Status

0
subcommand executed successfully
error in executing the subcommand

See Also

delete-javamail-resource(1), list-javamail-resources(1)

asadmin(1M)
create-jdbc-connection-pool

registers a JDBC connection pool

Synopsis

create-jdbc-connection-pool [--help]
[--datasourceclassname=datasourceclassname]
[--restype=resourcetype]
[--steadypoolsize=poolsize]
[--maxpoolsize=maxpoolsize]
[--maxwait=maxwaittime]
[--poolresizelimit=poolresizelimit]
[--idletimeout=idletimeout]
[--initsql=initsqlstring]
[--isolationlevel=isoationlevel]
[--isolationguaranteed={true|false}]
[--isconnectvalidatereq={false|true}]
[--validationmethod=validationmethod]
[--validationtable=validationtable]
[--failconnection={false|true}]
[--allownoncomponentcallers={false|true}]
[--nontransactionalconnections={false|true}]
[--validateatmostonceperiod=validationinterval]
[--leaktimeout=leaktimeout]
[--leakreclaim={false|true}]
[--statementleaktimeout=statementleaktimeout]
[--statementleakreclaim={false|true}]
[--creationretryattempts=creationretryattempts]
[--creationretryinterval=creationretryinterval]
[--sqltracelisteners=sqltracelisteners[,sqltracelisteners]]
[--statmenttimeout=statmenttimeout]
[--laryconnectionenlistment={false|true}]
[--laryconnectionassociation={false|true}]
[--associatewiththread={false|true}]
[--driverclassname=jdbcdriverclassname]
[--matchconnections={false|true}]
[--maxconnectionusagecount=maxconnectionusagecount]
[--ping={false|true}]
[--pooling={false|true}]
[--statementcachesize=statementcachesize]
[--validationclassname=validationclassname]
[--wrapjdbcobjects={false|true}]
[--description description]
[--property name=value[[:name=value]*]
[--target=target]
connectionpoolid

Description

The create-jdbc-connection-pool subcommand registers a new Java Database Connectivity ("JDBC") software connection pool with the specified JDBC connection pool name.
A JDBC connection pool with authentication can be created either by using a
--property option to specify user, password, or other connection information, or by
specifying the connection information in the XML descriptor file.

This subcommand is supported in remote mode only.

**Options**

**--help**
-?
Displays the help text for the subcommand.

**--datasourceclassname**
The name of the vendor-supplied JDBC datasource resource manager. An XA or global
transactions capable datasource class will implement the javax.sql.XADataSource
interface. Non-XA or exclusively local transaction datasources will implement the
javax.sql.DataSource interface.

**--restype**
Required when a datasource class implements two or more interfaces
javax.sql.DataSource, javax.sql.XADataSource, or
javax.sql.ConnectionPoolDataSource), or when a driver classname must be provided.
- If --restype = java.sql.Driver, then the --driverclassname option is required.
- If --restype = javax.sql.DataSource, javax.sql.XADataSource, or
javax.sql.ConnectionPoolDataSource, then the --datasourceclassname option
is required.
- If --restype is not specified, then either the --driverclassname or
--datasourceclassname option must be specified, but not both.

**--steadypoolsize**
The minimum and initial number of connections maintained in the pool. The default
value is 8.

**--maxpoolsize**
The maximum number of connections that can be created. The default value is 32.

**--maxwait**
The amount of time, in milliseconds, that a caller will wait before a connection timeout
is sent. The default is 60000 (60 seconds). A value of 0 forces the caller to wait
indefinitely.

**--poolresize**
Number of connections to be removed when idle-timeout-in-seconds timer expires.
This is the quantity by which the pool will scale up or scale down the number of
connections. Scale up: When the pool has no free connections, pool will scale up by
this quantity. Scale down: All the invalid and idle connections are removed, sometimes
resulting in removing connections of quantity greater than this value. Connections that
have been idle for longer than the timeout are candidates for removal. Steadypoolsize
will be ensured. Possible values are from 0 to MAX_INTEGER. The default value is 2.

**--idletimeout**
The maximum time, in seconds, that a connection can remain idle in the pool. After
this time, the implementation can close this connection. This timeout value must be
kept shorter than the database server side timeout value to prevent the accumulation
of unusable connections in the application. The default value is 300.
--initsql
An SQL string that is executed whenever a connection is created from the pool. If an
existing connection is reused, this string is not executed. Connections that have idled
for longer than the timeout are candidates for removal. This option has no default
value.

--isolationlevel
The transaction-isolation-level on the pooled database connections. This option does
not have a default value. If not specified, the pool operates with the default isolation
level that the JDBC driver provides. You can set a desired isolation level using one of
the standard transaction isolation levels: read-uncommitted, read-committed,
repeatable-read, serializable. Applications that change the isolation level on a
pooled connection programatically risk polluting the pool. This could lead to
program errors.

--isolationguaranteed
This is applicable only when a particular isolation level is specified for
transaction-isolation-level. The default value is true.

This option assures that every time a connection is obtained from the pool, isolation
level is set to the desired value. This could have some performance impact on some
JDBC drivers. Administrators can set this to false when the application does not
change --isolationlevel before returning the connection.

--isconnectvalidatereq
If set to true, connections are validated or checked to see if they are usable before
giving out to the application. The default value is false.

--validationmethod
Type of validation to be performed when is-connection-validation-required is
true. Valid settings are: auto-commit, meta-data, table, or custom-validation. The
default value is table.

--validationtable
The name of the validation table used to perform a query to validate a connection. If
is-connection-validation-required is set to true and connection-validation-type set
to table, this option is mandatory.

--failconnection
If set to true, all connections in the pool must be closed when a single validation check
fails. The default value is false. One attempt is made to reestablish failed connections.

--allownoncomponentcallers
A pool with this property set to true can be used by non-Java EE components, that is,
components other than EJBs or Servlets. The returned connection is enlisted
automatically with the transaction context obtained from the transaction manager.
Connections obtained by non-component callers are not automatically cleaned by the
container at the end of a transaction. These connections need to be explicitly closed by
the caller.

--nontransactionalconnections
A pool with this property set to true returns non-transactional connections. This
connection does not get automatically enlisted with the transaction manager.

--validateatmostonceperiod
Specifies the time interval in seconds between successive requests to validate a
connection at most once. Setting this attribute to an appropriate value minimizes the
number of validation requests by a connection. Default value is 0, which means that the attribute is not enabled.

--leaktimeout
Specifies the amount of time, in seconds, for which connection leaks in a connection pool are to be traced. When a connection is not returned to the pool by the application within the specified period, it is assumed to be a potential leak, and stack trace of the caller will be logged. This option only detects if there is a connection leak. The connection can be reclaimed only if connection-leak-reclaim is set to true.

If connection leak tracing is enabled, you can use the Administration Console to enable monitoring of the JDBC connection pool to get statistics on the number of connection leaks. The default value is 0, which disables connection leak tracing.

--leakreclaim
Specifies whether leaked connections are restored to the connection pool after leak connection tracing is complete. Possible values are as follows:

false
Leaked connections are not restored to the connection pool (default).

ture
Leaked connections are restored to the connection pool.

--statementleaktimeout
Specifies the amount of time, in seconds, after which any statements that have not been closed by an application are to be detected. Applications can run out of cursors if statement objects are not properly closed. This option only detects if there is a statement leak. The statement can be reclaimed only if statement-leak-reclaim is set to true. The leaked statement is closed when it is reclaimed.

The stack trace of the caller that creates the statement will be logged when a statement leak is detected. If statement leak tracing is enabled, you can use the Administration Console to enable monitoring of the JDBC connection pool to get statistics on the number of statement leaks. The default value is 0, which disables statement leak tracing.

The following limitations apply to the statement leak timeout value:

- The value must be less than the value set for the connection leak-timeout.
- The value must be greater than the value set for statement-timeout.

--statementleakreclaim
Specifies whether leaked statements are reclaimed after the statements leak. Possible values are as follows:

false
Leaked statements are not reclaimed (default).

ture
Leaked statements are reclaimed.

--creationretryattempts
Specifies the maximum number of times that GlassFish Server retries to create a connection if the initial attempt fails. The default value is 0, which specifies that GlassFish Server does not retry to create the connection.
--creationretryinterval
Specifies the interval, in seconds, between successive attempts to create a connection. If --creationretryattempts is 0, the --creationretryinterval option is ignored. The default value is 10.

--sqltracelisteners
A list of one or more custom modules that provide custom logging of database activities. Each module must implement the org.glassfish.api.jdbc.SQLTraceListener public interface. When set to an appropriate value, SQL statements executed by applications are traced. This option has no default value.

--statementtimeout
Specifies the length of time in seconds after which a query that is not completed is terminated.
A query that remains incomplete for a long period of time might cause the application that submitted the query to hang. To prevent this occurrence, use this option set a timeout for all statements that will be created from the connection pool that you are creating. When creating a statement, GlassFish Server sets the QueryTimeout property on the statement to the length of time that is specified. The default value is -1, which specifies that incomplete queries are never terminated.

--lazyconnectionenlistment
Specifies whether a resource to a transaction is enlisted only when a method actually uses the resource. Possible values are as follows:

    false
    Resources to a transaction are always enlisted and not only when a method actually uses the resource (default).

    true
    Resources to a transaction are enlisted only when a method actually uses the resource.

--lazyconnectionassociation
Specifies whether a physical connection should be associated with the logical connection only when the physical connection is used, and disassociated when the transaction is completed. Such association and dissociation enable the reuse of physical connections. Possible values are as follows:

    false
    A physical connection is associated with the logical connection even before the physical connection is used, and is not disassociated when the transaction is completed (default).

    true
    A physical connection is associated with the logical connection only when the physical connection is used, and disassociated when the transaction is completed. The --lazyconnectionenlistment option must also be set to true.

--associatewiththread
Specifies whether a connection is associated with the thread to enable the thread to reuse the connection. If a connection is not associated with the thread, the thread must obtain a connection from the pool each time that the thread requires a connection. Possible values are as follows:
false
A connection is not associated with the thread (default).

true
A connection is associated with the thread.

--driverclassname
The name of the vendor-supplied JDBC driver class. This driver should implement the java.sql.Driver interface.

--matchconnections
Specifies whether a connection that is selected from the pool should be matched by the resource adaptor. If all the connections in the pool are homogenous, a connection picked from the pool need not be matched by the resource adapter, which means that this option can be set to false. Possible values are as follows:

false
A connection should not be matched by the resource adaptor (default).

true
A connection should be matched by the resource adaptor.

--maxconnectionusagecount
Specifies the maximum number of times that a connection can be reused. When this limit is reached, the connection is closed. By limiting the maximum number of times that a connection can be reused, you can avoid statement leaks.

The default value is 0, which specifies no limit on the number of times that a connection can be reused.

--ping
Specifies if the pool is pinged during pool creation or reconfiguration to identify and warn of any erroneous values for its attributes. Default value is false.

--pooling
Specifies if connection pooling is enabled for the pool. The default value is true.

--statementcachesize
The number of SQL statements to be cached using the default caching mechanism (Least Recently Used). The default value is 0, which indicates that statement caching is not enabled.

--validationclassname
The name of the class that provides custom validation when the value of validationmethod is custom-validation. This class must implement the org.glassfish.api.jdbc.ConnectionValidation interface, and it must be accessible to GlassFish Server. This option is mandatory if the connection validation type is set to custom validation.

--wrapjdbcobjects
Specifies whether the pooling infrastructure provides wrapped JDBC objects to applications. By providing wrapped JDBC objects, the pooling infrastructure prevents connection leaks by ensuring that applications use logical connections from the connection pool, not physical connections. The use of logical connections ensures that the connections are returned to the connection pool when they are closed. However, the provision of wrapped JDBC objects can impair the performance of applications. The default value is true.
The pooling infrastructure provides wrapped objects for implementations of the following interfaces in the JDBC API:

- `java.sql.CallableStatement`
- `java.sql.DatabaseMetaData`
- `java.sql.PreparedStatement`
- `java.sql.ResultSet`
- `java.sql.Statement`

Possible values of `--wrapjdbcobjects` are as follows:

- **false**
  The pooling infrastructure does not provide wrapped JDBC objects to applications. (default).

- **true**
  The pooling infrastructure provides wrapped JDBC objects to applications.

`--description`
Text providing details about the specified JDBC connection pool.

`--property`
Optional attribute name/value pairs for configuring the pool. The following properties are available:

- **user**
  Specifies the user name for connecting to the database.

- **password**
  Specifies the password for connecting to the database.

- **databaseName**
  Specifies the database for this connection pool.

- **serverName**
  Specifies the database server for this connection pool.

- **port**
  Specifies the port on which the database server listens for requests.

- **networkProtocol**
  Specifies the communication protocol.

- **roleName**
  Specifies the initial SQL role name.

- **datasourceName**
  Specifies an underlying `XADataSource`, or a `ConnectionPoolDataSource` if connection pooling is done.

- **description**
  Specifies a text description.

- **url**
  Specifies the URL for this connection pool. Although this is not a standard property, it is commonly used.
**dynamic-reconfiguration-wait-timeout-in-seconds**

Used to enable dynamic reconfiguration of the connection pool transparently to the applications that are using the pool, so that applications need not be re-enabled for the attribute or property changes to the pool to take effect. Any in-flight transaction's connection requests will be allowed to complete with the old pool configuration as long as the connection requests are within the timeout period, so as to complete the transaction. New connection requests will wait for the pool reconfiguration to complete and connections will be acquired using the modified pool configuration.

**LazyConnectionEnlistment**

Deprecated. Use the equivalent attribute. The default value is false.

**LazyConnectionAssociation**

Deprecated. Use the equivalent attribute. The default value is false.

**AssociateWithThread**

Deprecated. Use the equivalent attribute. The default value is false.

**MatchConnections**

Deprecated. Use the equivalent attribute. The default value is true.

**Prefer-Validate-Over-Recreate**

Specifies whether pool resizer should validate idle connections before destroying and recreating them. The default value is true.

**time-to-keep-queries-in-minutes**

Specifies the number of minutes that will be cached for use in calculating frequently used queries. Takes effect when SQL tracing and monitoring are enabled for the JDBC connection pool. The default value is 5 minutes.

**number-of-top-queries-to-report**

Specifies the number of queries to list when reporting the top and most frequently used queries. Takes effect when SQL tracing and monitoring are enabled for the JDBC connection pool. The default value is 10 queries.

---

**Note:** If an attribute name or attribute value contains a colon, the backslash (\) must be used to escape the colon in the name or value. Other characters might also require an escape character. For more information about escape characters in command options, see the `asadmin(1M)` man page.

---

**--target**

Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

**Operands**

**connectionpoolid**

The name of the JDBC connection pool to be created.
Examples

Example 1 Creating a JDBC Connection Pool
This example creates a JDBC connection pool named sample_derby_pool.

```bash
asadmin> create-jdbc-connection-pool
--datasourceclassname org.apache.derby.jdbc.ClientDataSource
--restype javax.sql.XADataSource
--property portNumber=1527:password=APP:user=APP:serverName=localhost:databaseName=sun-appserv-samples:connectionAttributes=
create\=true sample_derby_pool
```
Command create-jdbc-connection-pool executed successfully

The escape character backslash (\) is used in the --property option to distinguish the semicolon (;). Two backslashes (\\) are used to distinguish the equal sign (=).

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-jdbc-connection-pool(1), list-jdbc-connection-pools(1)

asadmin(1M)
**create-jdbc-resource**

creates a JDBC resource with the specified JNDI name

**Synopsis**

create-jdbc-resource [--help]
--connectionpoolid connectionpoolid
[--enabled={false|true}]
[--description description]
[--property (property=value)[;name=value]...]
[--target target]
jndi_name

**Description**

The create-jdbc-resource subcommand creates a new JDBC resource.

This subcommand is supported in remote mode only.

**Options**

--help
Displays the help text for the subcommand.

--connectionpoolid
The name of the JDBC connection pool. If two or more JDBC resource elements point to the same connection pool element, they use the same pool connection at runtime.

--enabled
Determines whether the JDBC resource is enabled at runtime. The default value is true.

--description
Text providing descriptive details about the JDBC resource.

--property
Optional attribute name/value pairs for configuring the resource.

--target
This option helps specify the target to which you are deploying. Valid values are:

  server
  Deploys the component to the default server instance. This is the default value.

  domain
  Deploys the component to the domain.

  cluster_name
  Deploys the component to every server instance in the cluster.

  instance_name
  Deploys the component to a particular sever instance.
create-jdbc-resource

Operands

$jndi\_name$
The JNDI name of this JDBC resource.

Examples

Example 1  Creating a JDBC Resource
This example creates a JDBC resource named jdbc/DerbyPool.

```
asadmin> create-jdbc-resource
--connectionpoolid sample_derby_pool jdbc/DerbyPool
Command create-jdbc-resource executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-jdbc-resource(1), list-jdbc-resources(1), create-resource-ref(1)
asadmin(1M)

Note: The resource is always created for the domain as a whole, but the resource-ref for the resource is only created for the specified --target. This means that although the resource is defined at the domain level, it is only available at the specified target level. Use the create-resource-ref subcommand to refer to the resource in multiple targets if needed.
create-jmsdest

creates a JMS physical destination

Synopsis

create-jmsdest [--help]
--desttype dest_type
[--property \{name=value\}[:name=value]*]
[--target target]
dest_name

Description

The create-jmsdest subcommand creates a Java Message Service (JMS) physical destination. Typically, you use the create-jms-resource subcommand to create a JMS destination resource that has a Name property that specifies the physical destination. The physical destination is created automatically when you run an application that uses the destination resource. Use the create-jmsdest subcommand if you want to create a physical destination with non-default property settings.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

--help
-?
Displays the help text for the subcommand.

--desttype
The type of the JMS destination. Valid values are topic and queue.

--property
Optional attribute name/value pairs for configuring the physical destination. You can specify the following properties for a physical destination.

maxNumMsgs
The maximum number of unconsumed messages permitted for the destination. A value of -1 denotes an unlimited number of messages. The default value is -1. For the dead message queue, the default value is 1000.

If the limitBehavior property is set to FLOW_CONTROL, it is possible for the specified message limit to be exceeded because the broker cannot react quickly enough to stop the flow of incoming messages. In such cases, the value specified for maxNumMsgs serves as merely a hint for the broker rather than a strictly enforced limit.

maxBytesPerMsg
The maximum size, in bytes, of any single message. Rejection of a persistent message is reported to the producing client with an exception; no notification is sent for non-persistent messages.
The value may be expressed in bytes, kilobytes, or megabytes, using the following suffixes:

- **b**
  - Bytes

- **k**
  - Kilobytes (1024 bytes)

- **m**
  - Megabytes (1024 x 1024 = 1,048,576 bytes)

A value with no suffix is expressed in bytes; a value of -1 denotes an unlimited message size. The default value is -1.

**maxTotalMsgBytes**
The maximum total memory, in bytes, for unconsumed messages. The default value is -1. The syntax is the same as for `maxBytesPerMsg`. For the dead message queue, the default value is 10m.

**limitBehavior**
The behavior of the message queue broker when the memory-limit threshold is reached. Valid values are as follows.

- **REJECT_NEWEST**
  - Reject newest messages and notify the producing client with an exception only if the message is persistent. This is the default value.

- **FLOW_CONTROL**
  - Slow the rate at which message producers send messages.

- **REMOVE_OLEST**
  - Throw out the oldest messages.

- **REMOVE_LOW_PRIORITY**
  - Throw out the lowest-priority messages according to age, with no notification to the producing client.

If the value is **REMOVE_OLEST** or **REMOVE_LOW_PRIORITY** and the `useDMQ` property is set to `true`, excess messages are moved to the dead message queue. For the dead message queue itself, the default limit behavior is **REMOVE_OLEST**, and the value cannot be set to **FLOW_CONTROL**.

**maxNumProducers**
The maximum number of message producers for the destination. When this limit is reached, no new producers can be created. A value of -1 denotes an unlimited number of producers. The default value is 100. This property does not apply to the dead message queue.

**consumerFlowLimit**
The maximum number of messages that can be delivered to a consumer in a single batch. A value of -1 denotes an unlimited number of messages. The default value is 1000. The client runtime can override this limit by specifying a lower value on the connection factory object.

In load-balanced queue delivery, this is the initial number of queued messages routed to active consumers before load balancing begins.
useDMQ
If set to true, dead messages go to the dead message queue. If set to false, dead messages are discarded. The default value is true.

validateXMLSchemaEnabled
If set to true, XML schema validation is enabled for the destination. The default value is false.

When XML validation is enabled, the Message Queue client runtime will attempt to validate an XML message against the specified XSDs (or against the DTD, if no XSD is specified) before sending it to the broker. If the specified schema cannot be located or the message cannot be validated, the message is not sent, and an exception is thrown.

This property should be set when a destination is inactive: that is, when it has no consumers or producers and when there are no messages in the destination. Otherwise the producer must reconnect.

XMLSchemaURIList
A space-separated list of XML schema document (XSD) URI strings. The URIs point to the location of one or more XSDs to use for XML schema validation, if validateXMLSchemaEnabled is set to true. The default value is null.

Use double quotes around this value if multiple URIs are specified, as in the following example:


If this property is not set or null and XML validation is enabled, XML validation is performed using a DTD specified in the XML document. If an XSD is changed as a result of changing application requirements, all client applications that produce XML messages based on the changed XSD must reconnect to the broker.

To modify the value of these properties, you can use the as-install/mq/bin/imqcmd command. See “Physical Destination Property Reference” in Oracle GlassFish Server Message Queue Administration Guide for more information.

--target
Creates the physical destination only for the specified target. Although the create-jmsdest subcommand is related to resources, a physical destination is created using the JMS Service (JMS Broker), which is part of the configuration. A JMS Broker is configured in the config section of domain.xml. Valid values are as follows:

server
Creates the physical destination for the default server instance. This is the default value.

configuration-name
Creates the physical destination in the specified configuration.

cluster-name
Creates the physical destination for every server instance in the specified cluster.

instance-name
Creates the physical destination for the specified server instance.
Operands

dest_name
A unique identifier for the JMS destination to be created.

Examples

Example 1 Creating a JMS physical destination
The following subcommand creates a JMS physical queue named PhysicalQueue with
non-default property values.

asadmin> create-jmsdest --desttype queue
   --property maxNumMsgs=1000:maxBytesPerMsg=5k PhysicalQueue
Command create-jmsdest executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jms-resource(1), delete-jmsdest(1), list-jmsdest(1), flush-jmsdest(1)
asadmin(1M)
**create-jms-host**

creates a JMS host

**Synopsis**

create-jms-host [--help]
--mqhost mq-host --mqport mq-port
--mquser mq-user --mqpassword mq-password
[--target target] jms_host_name

**Description**

Creates a Java Message Service (JMS) host within the JMS service.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

**Options**

--help
-?
Displays the help text for the subcommand.

--mqhost
The host name for the JMS service.

--mqport
The port number used by the JMS service.

--mquser
The user name for the JMS service.

--mqpassword
The password for the JMS service.

--target
Creates the JMS host only for the specified target. Valid values are as follows:

server
Creates the JMS host for the default server instance. This is the default value.

configuration-name
Creates the JMS host in the specified configuration.

cluster-name
Creates the JMS host for every server instance in the specified cluster.

instance-name
Creates the JMS host for the specified server instance.
create-jms-host

Operands

`jms_host_name`
A unique identifier for the JMS host to be created.

Examples

Example 1  Creating a JMS host using a non-default port
The following command creates a JMS host named `MyNewHost` on the system `pigeon`.

```bash
asadmin> create-jms-host --mqhost pigeon.example.com --mqport 7677
--mquser admin --mqpassword admin MyNewHost
Jms Host MyNewHost created.
Command create-jms-host executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

`list-jms-hosts(1), delete-jms-host(1), jms-ping(1)`
`asadmin(1M)`
create-jms-resource

creates a JMS resource

Synopsis

create-jms-resource [--help]
--restype type
[--target target]
[--enabled={true|false}]  
[--description text]
[--property (name=value)[;name=value]*]

jndi_name

Description

The create-jms-resource subcommand creates a Java Message Service (JMS) connection factory resource or a JMS destination resource.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

--help
-?
Displays the help text for the subcommand.

--restype
The JMS resource type, which can be javax.jms.Topic, javax.jms.Queue, javax.jms.ConnectionFactory, javax.jms.TopicConnectionFactory, or javax.jms.QueueConnectionFactory.

--target
Creates the JMS resource only for the specified target. Valid values are as follows:

server
Creates the JMS resource for the default server instance. This is the default value.

domain
Creates the JMS resource for the domain.

cluster-name
Creates the JMS resource for every server instance in the specified cluster.

Note: The resource is always created for the domain as a whole, but the <resource-ref> element for the resource is only created for the specified --target. This means that although the resource is defined at the domain level, it is only active at the specified --target.
instance-name
Creates the JMS resource for the specified server instance.

--enabled
If set to true (the default), the resource is enabled at runtime.

--description
Text providing details about the JMS resource.

--property
Optional attribute name/value pairs for configuring the JMS resource.

You can specify the following properties for a connection factory resource:

ClientId
A client ID for a connection factory that will be used by a durable subscriber.

AddressList
A comma-separated list of message queue addresses that specify the host names
(and, optionally, port numbers) of a message broker instance or instances with
which your application will communicate. For example, the value could be earth
or earth:7677. Specify the port number if the message broker is running on a port
other than the default (7676), composed from the JMS hosts defined in the server’s
JMS service configuration. The default value is localhost and the default port
number is 7676. The client will attempt a connection to a broker on port 7676 of the
local host.

UserName
The user name for the connection factory. The default value is guest.

Password
The password for the connection factory. The default value is guest.

ReconnectEnabled
A value of true indicates that the client runtime attempts to reconnect to a
message server (or the list of addresses in the AddressList) when a connection is
lost. The default value is false.

ReconnectAttempts
The number of attempts to connect (or reconnect) for each address in the
AddressList before the client runtime tries the next address in the list. A value of
-1 indicates that the number of reconnect attempts is unlimited (the client runtime
attempts to connect to the first address until it succeeds). The default value is 6.

ReconnectInterval
The interval in milliseconds between reconnect attempts. This applies to attempts
on each address in the AddressList and for successive addresses in the list. If the
interval is too short, the broker does not have time to recover. If it is too long, the
reconnect might represent an unacceptable delay. The default value is 30,000
milliseconds.

AddressListBehavior
Specifies whether connection attempts are in the order of addresses in the
AddressList (PRIORITY) or in a random order (RANDOM). PRIORITY means that the
reconnect will always try to connect to the first server address in the AddressList
and will use another one only if the first broker is not available. If you have many
clients attempting a connection using the same connection factory, specify RANDOM
to prevent them from all being connected to the same address. The default value is the AddressListBehavior value of the server's JMS service configuration.

**AddressListIterations**
The number of times the client runtime iterates through the AddressList in an effort to establish (or re-establish) a connection. A value of -1 indicates that the number of attempts is unlimited. The default value is -1.

You can specify the following properties for a destination resource:

**Name**
The name of the physical destination to which the resource will refer. The physical destination is created automatically when you run an application that uses the destination resource. You can also create a physical destination with the create-jmsdest subcommand. If you do not specify this property, the JMS service creates a physical destination with the same name as the destination resource (replacing any forward slash in the JNDI name with an underscore).

**Description**
A description of the physical destination.

**Operands**

```
jndi_name
```
The JNDI name of the JMS resource to be created.

**Examples**

**Example 1  Creating a JMS connection factory resource for durable subscriptions**
The following subcommand creates a connection factory resource of type javax.jms.ConnectionFactory whose JNDI name is jms/DurableConnectionFactory. The ClientId property sets a client ID on the connection factory so that it can be used for durable subscriptions. The JNDI name for a JMS resource customarily includes the jms/naming subcontext.

```
asadmin> create-jms-resource --restype javax.jms.ConnectionFactory
--description "connection factory for durable subscriptions"
--property ClientId=MyID jms/DurableConnectionFactory
Connector resource jms/DurableConnectionFactory created.
Command create-jms-resource executed successfully.
```

**Example 2  Creating a JMS destination resource**
The following subcommand creates a destination resource whose JNDI name is jms/MyQueue. The Name property specifies the physical destination to which the resource refers.

```
asadmin> create-jms-resource --restype javax.jms.Queue
--property Name=PhysicalQueue jms/MyQueue
Administered object jms/MyQueue created.
Command create-jms-resource executed successfully.
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-jms-resource(1), list-jms-resources(1)
asadmin(1M)
create-jndi-resource

registers a JNDI resource

Synopsis

create-jndi-resource [--help] [--target target] 
--restype restype --factoryclass factoryclass 
--jndilookupname jndilookupname [--enabled={true|false}] 
[--description description] 
[--property {name=value}[;name=value]*] 
jndi-name

Description

The create-jndi-resource subcommand registers a JNDI resource. 
This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target for which you are registering a JNDI resource. Valid 
values for target are described below.

Note: The resource is always created for the domain as a whole, 
but the resource-ref for the resource is only created for the 
specified --target. This means that although the resource is 
declared at the domain level, it is only available at the specified 
target level. Use the create-resource-ref subcommand to refer to 
the resource in multiple targets if needed.

server
Creates the resource for the default server instance. This value is the default.

domain
Creates the resource for the domain

cluster-name
Creates the resource for every server instance in the cluster

instance-name
Creates the resource for a particular server instance

--restype
The JNDI resource type. Valid values are topic or queue.
--factoryclass
The class that creates the JNDI resource.

--jndilookupname
The lookup name that the external container uses.

--enabled
Determines whether the resource is enabled at runtime. Default is true.

--description
The text that provides details about the JNDI resource.

--property
Optional properties for configuring the resource. Each property is specified as a name-value pair.

The available properties are specific to the implementation that is specified by the --factoryclass option and are used by that implementation. GlassFish Server itself does not define any properties for configuring a JNDI resource.

Operands

jndi-name
The unique name of the JNDI resource to be created.

Examples

Example 1 Creating a JNDI Resource
This example creates the JNDI resource my-jndi-resource for the default server instance.

asadmin> create-jndi-resource
--restype com.example.jndi.MyResourceType
--factoryclass com.example.jndi.MyInitialContextFactoryClass
--jndilookupname remote-jndi-name
--description "sample JNDI resource" my-jndi-resource
JNDI resource my-jndi-resource created.
Command create-jndi-resource executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-jndi-resource(1), list-jndi-resources(1), create-resource-ref(1)
asadmin(1M)
**create-jvm-options**

creates options for the Java application launcher

**Synopsis**

```bash
create-jvm-options [-help] [--target target] [--profiler={true|false}]
(jvm-option-name=jvm-option-value) [:jvm-option-name=jvm-option-value*]
```

**Description**

The `create-jvm-options` subcommand creates command-line options that are passed to the Java application launcher when GlassFish Server is started. The options that this subcommand creates are in addition to the options that are preset with GlassFish Server. Java application launcher options are stored in the Java configuration `java-config` element or the profiler `profiler` element of the `domain.xml` file. The options are sent to the command line in the order they appear in the `java-config` element or the profiler `profiler` element in the `domain.xml` file.

Profiler options are used to record the settings that are required to start a particular profiler. The profiler must already exist. If necessary, use the `create-profiler(1)` subcommand to create the profiler.

This subcommand can be used to create the following types of options:

- **Java system properties.** These options are set through the `-D` option of the Java application launcher. For example:
  
  ```bash
  -Djava.security.manager
  -Denvironment=Production
  ```

- **Startup parameters for the Java application launcher.** These options are preceded by the dash character (`-`). For example:
  ```bash
  --XX:PermSize=size
  -Xmx1024m
  -d64
  ```

If the subcommand specifies an option that already exists, the command does not re-create the option.

---

**Note:** Ensure that any option that you create is valid. The subcommand might allow you to create an invalid option, but such an invalid option can cause startup to fail.

---

An option can be verified by examining the server log after GlassFish Server starts. Options for the Java application launcher are written to the `server.log` file before any other information when GlassFish Server starts.

The addition of some options requires a server restart for changes to become effective. Other options are set immediately in the environment of the domain administration server (DAS) and do not require a restart. Whether a restart is required depends on the type of option.
- Restart is not required for Java system properties whose names do not start with -Djava. or -Djavax. (including the trailing period). For example, restart is not required for the following Java system property:
  -Denvironment=Production
- Restart is required for the following options:
  - Java system properties whose names start with -Djava. or -Djavax. (including the trailing period). For example:
    -Djava.security.manager
  - Startup parameters for the Java application launcher. For example:
    -client
    -Xmx1024m
    -d64
To restart the DAS, use the restart-domain(1) command.
This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target on which you are creating Java application launcher options.
Valid values are as follows:
  server
  Specifies the DAS (default).
  instance-name
  Specifies a GlassFish Server instance.
  cluster-name
  Specifies a cluster.
  configuration-name
  Specifies a named configuration.

--profiler
Indicates whether the Java application launcher options are for the profiler. The profiler must exist for this option to be true. Default is false.

Operands

jvm-option-name
One or more options delimited by a colon (:). The format of an option depends on the following:
- If the option has a name and a value, the format is option-name=value.
- If the option has only a name, the format is option-name. For example, -Xmx2048m.
If the first option name could be misinterpreted as one or more asadmin short options, the format is -- option-name. For example, -server in the following command could be misinterpreted as -se, the asadmin short forms for --secure and --echo:

create-jvm-options -server

To create the JVM option -server, instead use the command:

create-jvm-options -- -server

---

**Note:** If an option name or option value contains a colon, the backslash (\) must be used to escape the colon in the name or value. Other characters might also require an escape character. For more information about escape characters in subcommand options, see the asadmin(1M) man page.

---

**Examples**

**Example 1 Setting Java System Properties**
This example sets multiple Java system properties.

```
asadmin> create-jvm-options -Dunixlocation=/root/example: -Dvariable=$HOME: -Dwindowslocation=d:\sun\appserver: -Doption1=value1
created 4 option(s)
Command create-jvm-options executed successfully.
```

**Example 2 Setting a Startup Parameter for the Java Application Launcher**
This example sets the maximum available heap size to 1024.

```
asadmin> create-jvm-options -Xmx1024m
created 1 option(s)
Command create-jvm-options executed successfully.
```

**Example 3 Setting Multiple Startup Parameters for the Java Application Launcher**
This example sets the maximum available heap size to 1024 and requests details about garbage collection.

```
asadmin> create-jvm-options "-Xmx1024m: -XX:+PrintGCDetails"
created 1 option(s)
Command create-jvm-options executed successfully.
```

In this case, one of the two parameters already exists, so the subcommand reports that only one option was set.

**Example 4 Setting a JVM Startup Parameter for the Profiler**
This example sets a JVM startup parameter for the profiler.

```
asadmin> create-jvm-options --profiler=true -XX:MaxPermSize=192m
created 1 option(s)
Command create-jvm-options executed successfully.
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-jvm-options(1), list-jvm-options(1), create-profile(1),
restart-domain(1)

asadmin(1M)

For more information about the Java application launcher, see the reference page for the operating system that you are using:

- Oracle Solaris and Linux: java - the Java application launcher
  (http://java.sun.com/javase/6/docs/technotes/tools/solaris/java.html)
- Windows: java - the Java application launcher
  (http://java.sun.com/javase/6/docs/technotes/tools/windows/java.html)
create-lifecycle-module

creates a lifecycle module

Synopsis

create-lifecycle-module [--help] --classname classname
[--enabled={true|false}] [--target target]
[--classpath classpath] [--loadorder loadorder]
[--failurefatal={false|true}] [--description description]
[--property (name=value)|:name=value]*

module_name

Description

The create-lifecycle-module subcommand creates a lifecycle module. A lifecycle module provides a means of running a short or long duration Java-based task at a specific stage in the server life cycle. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--classname
This is the fully qualified name of the startup class.

--target
Indicates the location where the lifecycle module is to be created. Valid values are

- server- Specifies the default server instance as the target for creating the lifecycle module. server is the name of the default server instance and is the default value for this option.
- cluster_name- Specifies a particular cluster as the target for creating the lifecycle module.
- instance_name- Specifies a particular stand-alone server instance as the target for creating the lifecycle module.

--classpath
This option indicates where the lifecycle module is located. It is a classpath with the standard format: either colon-separated (Unix) or semicolon-separated (Windows) JAR files and directories. The referenced JAR files and directories are not uploaded to the server instance.

--loadorder
This option represents an integer value that can be used to force the order in which deployed lifecycle modules are loaded at server startup. Smaller numbered modules are loaded sooner. Order is unspecified if two or more lifecycle modules have the same load-order value. The default is Integer.MAX_VALUE, which means the lifecycle module is loaded last.
--failurefatal
This option tells the system what to do if the lifecycle module does not load correctly. When this option is set to true, the system aborts the server startup if this module does not load properly. The default value is false.

--enabled
This option determines whether the lifecycle module is enabled at runtime. The default value is true.

--description
This is the text description of the lifecycle module.

--property
This is an optional attribute containing name/value pairs used to configure the lifecycle module.

Operands

module_name
This operand is a unique identifier for the deployed server lifecycle event listener module.

Examples

Example 1 Creating a Lifecycle Module
The following example creates a lifecycle module named customSetup.

asadmin> create-lifecycle-module --classname "com.acme.CustomSetup"
--classpath "/export/customSetup" --loadorder 1 --failurefatal=true
--description "this is a sample customSetup"
--property rmi="Server\=acme1\:7070":timeout=30 customSetup
Command create-lifecycle-module executed successfully

The escape character \ is used in the property option to specify that the equal sign (=) and colon (:) are part of the rmi property value.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-lifecycle-module(1), list-lifecycle-modules(1)
asadmin(1M)

"Developing Lifecycle Listeners" in Oracle GlassFish Server Application Development Guide
create-local-instance

creates a GlassFish Server instance on the host where the subcommand is run

Synopsis

create-local-instance [--help]
[<--node name-of-node] [--node-dir node-dir]
[<--config config-name] [--cluster cluster-name]
[<--lbenabled={true|false}]
[<--portbase port-number] [--checkports={true|false}]
[<--savemasterpassword={false|true}]
[<--usemasterpassword={false|true}]
[<--systemproperties {name=value} [:name=value]* ]
instance-name

Description

The create-local-instance subcommand creates a GlassFish Server instance on the node that represents the host where the subcommand is run. This subcommand does not require the Distributed Component Object Model (DCOM) remote protocol or secure shell (SSH) to be configured.

You must run this subcommand from the host that is represented by the node where the instance is to reside. To contact the domain administration server (DAS), this subcommand requires the name of the host where the DAS is running. If a nondefault port is used for administration, this subcommand also requires the port number. If you are adding the first instance to a node, you must provide this information through the --host option and the --port option of the asadmin(1M) utility. For the second and later instances, this information is obtained from the DAS properties of the node.

A GlassFish Server instance is a single Virtual Machine for the Java platform (Java Virtual Machine or JVM machine) on a single node in which GlassFish Server is running. A node defines the host where the GlassFish Server instance resides. The JVM machine must be compatible with the Java Platform, Enterprise Edition (Java EE).

A GlassFish Server instance requires a reference to the following items:

- The node that defines the host where the instance resides. The node can be specified in the command to create the instance, but is required only if more than one node exists in the directory where files for nodes are stored. If no node is specified, the behavior of the subcommand depends on the number of existing nodes in the directory where nodes are stored:
  - If no nodes exist, the subcommand creates a node for the instance. The name of the node is the name of the host on which the subcommand is run.
  - If only one node exists, the subcommand creates a reference to the existing node for the instance.
  - If two or more nodes exist, an error occurs.

- The named configuration that defines the configuration of the instance. The configuration can be specified in the command to create the instance, but is not required. If no configuration is specified for an instance that is not joining a cluster,
the subcommand creates a configuration for the instance. An instance that is joining a cluster receives its configuration from its parent cluster.

Each GlassFish Server instance is one of the following types of instance:

**Standalone instance**
A standalone instance does not share its configuration with any other instances or clusters. A standalone instance is created if either of the following conditions is met:

- No configuration or cluster is specified in the command to create the instance.
- A configuration that is not referenced by any other instances or clusters is specified in the command to create the instance.

When no configuration or cluster is specified, a copy of the `default-config` configuration is created for the instance. The name of this configuration is `instance-name-config`, where `instance-name` represents the name of an unclustered server instance.

**Shared instance**
A shared instance shares its configuration with other instances or clusters. A shared instance is created if a configuration that is referenced by other instances or clusters is specified in the command to create the instance.

**Clustered instance**
A clustered instance inherits its configuration from the cluster to which the instance belongs and shares its configuration with other instances in the cluster. A clustered instance is created if a cluster is specified in the command to create the instance.

Any instance that is not part of a cluster is considered an unclustered server instance. Therefore, standalone instances and shared instances are unclustered server instances.

By default, this subcommand attempts to resolve possible port conflicts for the instance that is being created. The subcommand also assigns ports that are currently not in use and not already assigned to other instances on the same node. The subcommand assigns these ports on the basis of an algorithm that is internal to the subcommand. Use the `--systemproperties` option to resolve port conflicts for additional instances on the same node. System properties of an instance can be manipulated by using the `create-system-properties(1)` subcommand and the `delete-system-property(1)` subcommand.

When creating an instance, the subcommand retrieves the files that are required for secure synchronization with the domain administration server (DAS). The instance is synchronized with the DAS when the instance is started.

**Options**

```
--help
-?
```
Displays the help text for the subcommand.

```
--node
```
The name of the node that defines the host where the instance is to be created. The node must be specified only if more than one node exists in the directory where nodes are stored. Otherwise, the node may be omitted. If a node is specified, the node must exist.

If no node is specified, the behavior of the subcommand depends on the number of existing nodes in the directory where nodes are stored:
If no nodes exist, the subcommand creates a node for the instance. The name of the node is the name of the host on which the subcommand is run.

If only one node exists, the subcommand creates a reference to the existing node for the instance.

If two or more nodes exist, an error occurs.

`--nodedir`
The path to the directory in which the files for instance’s node is to be stored. The default is `as-install/nodes`.

`--config`
Specifies the named configuration that the instance references. The configuration must exist and must not be named `default-config` or `server-config`. Specifying the `--config` option creates a shared instance.

The `--config` option and the `--cluster` option are mutually exclusive. If both options are omitted, a standalone instance is created.

`--cluster`
Specifies the cluster from which the instance inherits its configuration. Specifying the `--cluster` option creates a clustered instance.

The `--config` option and the `--cluster` option are mutually exclusive. If both options are omitted, a standalone instance is created.

`--lbenabled`
Specifies whether the instance is enabled for load balancing. Possible values are as follows:

- **true**
  The instance is enabled for load balancing (default).
  When an instance is enabled for load balancing, a load balancer sends requests to the instance.

- **false**
  The instance is disabled for load balancing.
  When an instance is disabled for load balancing, a load balancer does not send requests to the instance.

`--portbase`
Determines the number with which the port assignment should start. An instance uses a certain number of ports that are statically assigned. The `portbase` value determines where the assignment should start. The values for the ports are calculated as follows:

- Administration port: `portbase + 48`
- HTTP listener port: `portbase + 80`
- HTTPS listener port: `portbase + 81`
- JMS port: `portbase + 76`
- IIOP listener port: `portbase + 37`
- Secure IIOP listener port: `portbase + 38`
- Secure IIOP with mutual authentication port: `portbase + 39`
- JMX port: `portbase + 86`
- JPA debugger port: `portbase + 9`
- Felix shell service port for OSGi module management: `portbase + 66`

When the `--portbase` option is specified, the output of this subcommand includes a complete list of used ports.

`--checkports`
Specifies whether to check for the availability of the administration, HTTP, JMS, JMX, and IIOP ports. The default value is `true`.

`--savemasterpassword`
Setting this option to `true` allows the master password to be written to the file system. If the master password is written to the file system, the instance can be started without the need to prompt for the password. If this option is `true`, the `--usemasterpassword` option is also true, regardless of the value that is specified on the command line. Because writing the master password to the file system is an insecure practice, the default is `false`.

The master-password file for an instance is saved in the node directory, not the domain directory. Therefore, this option is required only for the first instance that is created for each node in a domain.

`--usemasterpassword`
Specifies whether the key store is encrypted with a master password that is built into the system or a user-defined master password.

If `false` (default), the keystore is encrypted with a well-known password that is built into the system. Encrypting the keystore with a password that is built into the system provides no additional security.

If `true`, the subcommand obtains the master password from the `AS_ADMIN_MASTERPASSWORD` entry in the password file or prompts for the master password. The password file is specified in the `--passwordfile` option of the `asadmin(1M)` utility.

If the `--savemasterpassword` option is `true`, this option is also true, regardless of the value that is specified on the command line.

The master password must be the same for all instances in a domain.

`--systemproperties`
Defines system properties for the instance. These properties override property definitions for port settings in the instance's configuration. Predefined port settings must be overridden if, for example, two clustered instances reside on the same host. In this situation, port settings for one instance must be overridden because both instances share the same configuration.

The following properties are available:

**ASADMIN_LISTENER_PORT**
This property specifies the port number of the HTTP port or HTTPS port through which the DAS connects to the instance to manage the instance. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**HTTP_LISTENER_PORT**
This property specifies the port number of the port that is used to listen for HTTP requests. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.
HTTP_SSL_LISTENER_PORT
This property specifies the port number of the port that is used to listen for HTTPS requests. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

IIOP_LISTENER_PORT
This property specifies the port number of the port that is used for IIOP connections. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

IIOP_SSL_LISTENER_PORT
This property specifies the port number of the port that is used for secure IIOP connections. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

IIOP_SSL_MUTUALAUTH_PORT
This property specifies the port number of the port that is used for secure IIOP connections with client authentication. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

JAVA_DEBUGGER_PORT
This property specifies the port number of the port that is used for connections to the Java Platform Debugger Architecture (JPDA) (http://java.sun.com/javase/technologies/core/tools/apis/jpda/) debugger. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

JMS_PROVIDER_PORT
This property specifies the port number for the Java Message Service provider. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

JMX_SYSTEM_CONNECTOR_PORT
This property specifies the port number on which the JMX connector listens. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

OSGI_SHELL_TELNET_PORT
This property specifies the port number of the port that is used for connections to the Apache Felix Remote Shell (http://felix.apache.org/site/apache-felix-remote-shell.html). This shell uses the Felix shell service to interact with the OSGi module management subsystem. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.

**Operands**

*instance-name*
The name of the instance that is being created.

The name must meet the following requirements:

- The name may contain only ASCII characters.
- The name must start with a letter, a number, or an underscore.
- The name may contain only the following characters:
  - Lowercase letters
- Uppercase letters
- Numbers
- Hyphen
- Period
- Underscore

- The name must be unique in the domain and must not be the name of another GlassFish Server instance, a cluster, a named configuration, or a node.
- The name must not be domain, server, or any other keyword that is reserved by GlassFish Server.

Examples

Example 1 Creating a Standalone GlassFish Server Instance
This example creates the standalone instance il3 on the host where the command is run. The DAS is running on the same host. The instance references the only existing node.

```
asadmin> create-local-instance il3
Rendezvoused with DAS on localhost:4848.
Port Assignments for server instance il3:
  JMX_SYSTEM_CONNECTOR_PORT=28686
  JMS_PROVIDER_PORT=27676
  HTTP_LISTENER_PORT=28080
  ASADMIN_LISTENER_PORT=24848
  JAVA_DEBUGGER_PORT=29009
  IIOP_SSL_LISTENER_PORT=23820
  IIOP_LISTENER_PORT=23700
  OSGI_SHELL_TELNET_PORT=26666
  HTTP_SSL_LISTENER_PORT=28181
  IIOP_SSL_MUTUALAUTH_PORT=23920
Command create-local-instance executed successfully.
```

Example 2 Creating a Clustered GlassFish Server Instance on a Specific Node
This example creates the clustered instance ymli2 on node sj02. The instance is a member of the cluster ymlclust.

The command is run on the host sj02, which is the host that the node sj02 represents. The DAS is running on the host sr04 and uses the default HTTP port for administration. Because no instances exist on the node, the host on which the DAS is running is provided through the --host option of the asadmin utility.

```
sj02# asadmin --host sr04 create-local-instance --cluster ymlclust --node sj02 ymli2
Rendezvoused with DAS on sr04:4848.
Port Assignments for server instance ymli2:
  JMX_SYSTEM_CONNECTOR_PORT=28686
  JMS_PROVIDER_PORT=27676
  HTTP_LISTENER_PORT=28080
  ASADMIN_LISTENER_PORT=24848
  JAVA_DEBUGGER_PORT=29009
  IIOP_SSL_LISTENER_PORT=23820
  IIOP_LISTENER_PORT=23700
  OSGI_SHELL_TELNET_PORT=26666
  HTTP_SSL_LISTENER_PORT=28181
```
IIOP_SSL_MUTUALAUTH_PORT=23920
Command create-local-instance executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-instance(1), create-node-config(1), create-node-dcom(1),
create-node-ssh(1), create-system-properties(1), delete-local-instance(1),
delete-system-property(1), list-instances(1), start-local-instance(1),
stop-local-instance(1)

asadmin(1M)
create-message-security-provider

enables administrators to create a message security provider, which specifies how
SOAP messages will be secured.

Synopsis

create-message-security-provider [--help]

[--target target]

--classname provider_class

[--layer message_layer] [--providertype provider_type]

[--requestauthsource request_auth_source]

[--requestauthrecipient request_auth_recipient]

[--responseauthsource response_auth_source]

[--responseauthrecipient response_auth_recipient]

[--isdefaultprovider]

[--property name=value[:name=value]*]

provider_name

Description

The create-message-security-provider subcommand enables the administrator to
create a message security provider for the security service which specifies how SOAP
messages will be secured.

This command is supported in remote mode only.

Options

If an option has a short option name, then the short option precedes the long option
name. Short options have one dash whereas long options have two dashes.

--help

-?

Displays the help text for the subcommand.

--target

Specifies the target for which you are creating the message security provider. The
following values are valid:

server

Creates the provider for the default server instance server and is the default
value.

domain

Creates the provider for the domain.

cluster_name

Creates the provider for every server instance in the cluster.

instance_name

Creates the provider for a particular sever instance.

--classname

Defines the Java implementation class of the provider. Client authentication providers
must implement the com.sun.enterprise.security.jauth.ClientAuthModule
interface. Server-side providers must implement the com.sun.enterprise.security.jauth.ServerAuthModule interface. A provider may implement both interfaces, but it must implement the interface corresponding to its provider type.

--layer
The message-layer entity used to define the value of the auth-layer attribute of message-security-config elements. The default is HttpServlet. Another option is SOAP.

--providertype
Establishes whether the provider is to be used as client authentication provider, server authentication provider, or both. Valid options for this property include client, server, or client-server.

--requestauthsource
The auth-source attribute defines a requirement for message-layer sender authentication (e.g. username password) or content authentication (e.g. digital signature) to be applied to request messages. Possible values are sender or content. When this argument is not specified, source authentication of the request is not required.

--requestauthrecipient
The auth-recipient attribute defines a requirement for message-layer authentication of the receiver of a message to its sender (e.g. by XML encryption). Possible values are before-content or after-content. The default value is after-content.

--responseauthsource
The auth-source attribute defines a requirement for message-layer sender authentication (e.g. username password) or content authentication (e.g. digital signature) to be applied to response messages. Possible values are sender or content. When this option is not specified, source authentication of the response is not required.

--responseauthrecipient
The auth-recipient attribute defines a requirement for message-layer authentication of the receiver of the response message to its sender (e.g. by XML encryption). Possible values are before-content or after-content. The default value is after-content.

--isdefaultprovider
The default-provider attribute is used to designate the provider as the default provider (at the layer) of the type or types identified by the providertype argument. There is no default associated with this option.

--property
Use this property to pass provider-specific property values to the provider when it is initialized. Properties passed in this way might include key aliases to be used by the provider to get keys from keystores, signing, canonicalization, encryption algorithms, etc.

The following properties may be set:

security.config
Specifies the location of the message security configuration file. To point to a configuration file in the domain-dir/config directory, use the system property ${com.sun.aas.instanceRoot}/config/, for example:
${com.sun.aas.instanceRoot}/config/wss-server-config-1.0.xml. The default is domain-dir/config/ wss-serverconfig-1.0.xml.
debug
If true, enables dumping of server provider debug messages to the server log. The default is false.

dynamic.username.password
If true, signals the provider runtime to collect the user name and password from the CallbackHandler for each request. If false, the user name and password for wsse:UsernameToken(s) is collected once, during module initialization. This property is only applicable for a ClientAuthModule. The default is false.

encryption.key.alias
Specifies the encryption key used by the provider. The key is identified by its keystore alias. The default value is s1as.

signature.key.alias
Specifies the signature key used by the provider. The key is identified by its keystore alias. The default value is s1as.

Operands

provider_name
The name of the provider used to reference the provider-config element.

Examples

Example 1 Creating a Message Security Provider
The following example shows how to create a message security provider for a client.

asadmin> create-message-security-provider
--classname com.sun.enterprise.security.jauth.ClientAuthModule
--providertype client mySecurityProvider

Exit Status

0
command executed successfully

1
ter error in executing the command

See Also

delete-message-security-provider(1), list-message-security-providers(1)

asadmin(1M)
create-network-listener

adds a new network listener socket

Synopsis

create-network-listener [--help]
[--address address]
--listenerport listener-port
[--threadpool thread-pool]
--protocol protocol
[--transport transport]
[--enabled={true|false}]
[--jkenabled={false|true}]
[--target target]
listener-name

Description

The create-network-listener subcommand creates a network listener. This subcommand is supported in remote mode only.

Note: If you edit the special network listener named admin-listener, you must restart the server for the changes to take effect. The Administration Console does not tell you that a restart is required in this case.

Note: You can use the create-http-listener subcommand to create a network listener that uses the HTTP protocol without having to first create a protocol, transport, or HTTP configuration. This subcommand is a convenient shortcut, but it gives access to only a limited number of options.

Options

--help
-?
Displays the help text for the subcommand.

--address
The IP address or the hostname (resolvable by DNS).

--listenerport
The port number to create the listen socket on. Legal values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges. Configuring an SSL listen socket to listen on port 443 is standard.

--threadpool
The name of the thread pool for this listener. Specifying a thread pool is optional. The default is http-thread-pool.
--protocol
The name of the protocol for this listener.

--transport
The name of the transport for this listener. Specifying a transport is optional. The default is tcp.

--enabled
If set to true, the default, the listener is enabled at runtime.

--jkenabled
If set to true, mod_jk is enabled for this listener. The default is false.

--target
Creates the network listener only on the specified target. Valid values are as follows:

  server
  Creates the network listener on the default server instance. This is the default value.

  configuration-name
  Creates the network listener in the specified configuration.

  cluster-name
  Creates the network listener on all server instances in the specified cluster.

  standalone-instance-name
  Creates the network listener on the specified standalone server instance.

Operands

listener-name
The name of the network listener.

Examples

Example 1 Creating a Network Listener
The following command creates a network listener named sampleListener that is not enabled at runtime:

  asadmin> create-network-listener --listenerport 7272 protocol http-1
  --enabled=false sampleListener
Command create-network-listener executed successfully.

Exit Status

  0
  command executed successfully

  1
  error in executing the command

See Also

delete-network-listener(1), list-network-listeners(1), create-transport(1),
create-protocol(1), create-threadpool(1), create-http-listener(1)
asadmin(1M)
create-node-config

creates a node that is not enabled for remote communication

Synopsis

create-node-config [--help]
[-nodehost node-host]
[-installdir as-install-parent] [-nodedir node-dir] node-name

Description

The create-node-config subcommand creates a node that is not enabled for remote communication. The create-node-config subcommand does not require the Distributed Component Object Model (DCOM) remote protocol or secure shell (SSH) to be configured to create the node.

A node represents a host on which the GlassFish Server software is installed. A node must exist for every host on which GlassFish Server instances reside.

Note: To represent the host where the DAS is running, GlassFish Server provides the predefined node localhost-domain. The predefined node localhost-domain is not enabled for remote communication.

All administration of instances on a node that is not enabled for remote communication must be performed on the host that the node represents. The domain administration server (DAS) on a remote host cannot contact the node. To administer instances on a node that represents a host that is remote from the DAS, you must use the following subcommands:

- create-local-instance(1)
- delete-local-instance(1)
- start-local-instance(1)

However, you may use stop-local-instance(1) or stop-instance(1) to stop the instances.

This subcommand is supported in remote mode only.

Options

--help
?-
Displays the help text for the subcommand.

--nodehost
The name of the host that the node represents. If this option is omitted, no host is specified for the node.
--installdir
The full path to the parent of the base installation directory of the GlassFish Server software on the host, for example, /export/glassfish3/. If this option is omitted, no parent of the base installation directory of the GlassFish Server software is specified for the node.

--nodedir
The path to the directory that is to contain GlassFish Server instances that are created on the node. If a relative path is specified, the path is relative to the as-install directory. If this option is omitted, no directory for instances is specified for the node.

Operands

node-name
The name of the node.

The name must meet the following requirements:

■ The name may contain only ASCII characters.
■ The name must start with a letter, a number, or an underscore.
■ The name may contain only the following characters:
  – Lowercase letters
  – Uppercase letters
  – Numbers
  – Hyphen
  – Period
  – Underscore
■ The name must be unique in the domain and must not be the name of another node, a cluster, a named configuration, or a GlassFish Server instance.
■ The name must not be domain, server, or any other keyword that is reserved by GlassFish Server.

Examples

Example 1 Creating a Node That Is Not Enabled for Remote Communication
This example creates the node sj03 for host sj03.example.com. The node is not enabled for remote communication.

asadmin> create-node-config --nodehost sj03.example.com sj03

Command create-node-config executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command
See Also

create-local-instance(1), create-node-dcom(1), create-node-ssh(1),
delete-local-instance(1), delete-node-config(1), install-node(1),
install-node-dcom(1), install-node-ssh(1), list-nodes(1),
start-local-instance(1), stop-instance(1), stop-local-instance(1),
uninstall-node(1), uninstall-node-dcom(1), uninstall-node-ssh(1),
update-node-config(1), update-node-dcom(1), update-node-ssh(1)
asadmin(1M)
create-node-dcom

creates a node that is enabled for communication over DCOM

Synopsis

create-node-dcom [--help]
--nodehost node-host
[--installdir as-install-parent] [--nodedir node-dir]
[--windowsuser windows-user] [--windowsdomain windows-domain]
[--force={false|true}]
[--install={false|true}] [--archive archive]
node-name

Description

The create-node-dcom subcommand creates a node that is enabled for communication over the Distributed Component Object Model (DCOM) remote protocol. The DCOM protocol is available only on Windows systems.

A node represents a host on which the GlassFish Server software is installed. A node must exist for every host on which GlassFish Server instances reside.

In a DCOM-enabled network, the domain administration server (DAS) contacts a node's host through the DCOM connector to manage GlassFish Server instances that reside on the node. DCOM is used only for communications between hosts. DCOM is never used for communications within a single host.

By default, the subcommand fails and the node is not created if the DAS cannot contact the node's host through DCOM. To force the node to be created in the DAS configuration even if the host cannot be contacted through DCOM, set the --force option to true.

The create-node-dcom subcommand cannot create a node that represents the local host, that is, the host on which the subcommand is run. The subcommand fails regardless of how the local host is specified, for example, by using the keyword localhost, the Internet Protocol (IP) address of the host, or the host name.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--nodehost
The name of the host that the node represents. The name of the host must be specified. Otherwise, an error occurs.

--installdir
The full path to the parent of the base installation directory of the GlassFish Server software on the host, for example, C:\glassfish3. The default is the parent of the default base installation directory of the GlassFish Server software for the DAS. This default is useful only if GlassFish Server is installed in the same location on all hosts.
create-node-dcom

--nodedir
The path to the directory that is to contain GlassFish Server instances that are created on the node. The default is as-install\nodes, where as-install is the base installation directory of the GlassFish Server software on the host. If a relative path is specified, the path is relative to the as-install directory.

--windowsuser
-w
The user on this node’s host that is to run the process for connecting to the host through DCOM. The default is the user that is running the DAS process. If the --nodehost option is set to localhost-domain, the --windowsuser option is ignored.

--windowsdomain
-d
The name of the Windows domain that contains the user that the --windowsuser option specifies. The default is the name of the host on which the subcommand is run.

--force
Specifies whether the node is created in the DAS configuration even if validation of the node’s parameters fails. To validate a node’s parameters, the DAS must be able to contact the node’s host through DCOM. Possible values are as follows:

false
The node is not created if validation of the node’s parameters fails (default).

true
The node is created even if validation of the node’s parameters fails.

--install
Specifies whether the subcommand shall install the GlassFish Server software on the host that the node represents.

Possible values are as follows:

false
The subcommand shall not install the GlassFish Server software on the host (default).

true
The subcommand shall install the GlassFish Server software on the host.

--archive
The absolute path to the archive file of the GlassFish Server software that is to be installed. If this option is omitted and the --install is true, the subcommand creates a ZIP archive of the GlassFish Server software from the installation where this subcommand is run. The archive does not contain the domains directory or the nodes directory.

Operands

node-name
The name of the node.

The name must meet the following requirements:

- The name may contain only ASCII characters.
- The name must start with a letter, a number, or an underscore.
The name may contain only the following characters:
- Lowercase letters
- Uppercase letters
- Numbers
- Hyphen
- Period
- Underscore

The name must be unique in the domain and must not be the name of another node, a cluster, a named configuration, or a GlassFish Server instance.

The name must not be domain, server, or any other keyword that is reserved by GlassFish Server.

Examples

Example 1 Creating a Node
This example creates the node wpmdl1 for the host wpmdl1.example.com. By default, the parent of the base installation directory of the GlassFish Server software is C:\glassfish3.

asadmin> create-node-dcom
--nodehost wpmdl1.example.com wpmdl1
Command create-node-dcom executed successfully.

Example 2 Creating a Node With a Specific Installation Directory
This example creates the node wyml1 for the host wyml1.example.com. The parent of the base installation directory of the GlassFish Server software on this host is C:\middleware\glassfish3.

asadmin> create-node-dcom --installdir C:\middleware\glassfish3
--nodehost wyml1.example.com wyml1
Command create-node-dcom executed successfully.

Example 3 Forcing the Creation of a Node
This example forces the creation of node xkyd for the host xkyd.example.com. The node is created despite the failure of the DAS to contact the host xkyd.example.com to validate the node's parameters.

asadmin> create-node-dcom --force --nodehost xkyd.example.com xkyd
Warning: some parameters appear to be invalid.
com.sun.enterprise.universal.process.WindowsException: Connection in error
Continuing with node creation due to use of --force.
Command create-node-dcom executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command
See Also

create-node-config(1), create-node-ssh(1), delete-node-dcom(1),
install-node-dcom(1), list-nodes(1), uninstall-node-dcom(1), update-node-dcom(1)
asadmin(1M)
**create-node-ssh**

creates a node that is enabled for communication over SSH

**Synopsis**

create-node-ssh [--help]
--nodehost node-host
[--installdir as-install-parent] [--nodedir node-dir]
[--sshport ssh-port] [--sshuser ssh-user]
[--sshkeyfile ssh-keyfile]
[--force={false|true}]
[--install={false|true}] [--archive archive]
node-name

**Description**

The create-node-ssh subcommand creates a node that is enabled for communication over secure shell (SSH).

A node represents a host on which the GlassFish Server software is installed. A node must exist for every host on which GlassFish Server instances reside.

The domain administration server (DAS) contacts an SSH node's host through the SSH connector to manage GlassFish Server instances that reside on the node. However, the DAS does not use the SSH connector to contact the host where the DAS is running because the DAS can run all asadmin subcommands locally.

By default, the subcommand fails and the node is not created if the DAS cannot contact the node's host through SSH. To force the node to be created in the DAS configuration even if the host cannot be contacted through SSH, set the --force option to true.

This subcommand is supported in remote mode only.

**Options**

--help
-?
Displays the help text for the subcommand.

--nodehost
The name of the host that the node represents. The name of the host must be specified. Otherwise, an error occurs.

--installdir
The full path to the parent of the base installation directory of the GlassFish Server software on the host, for example, /export/glassfish3/. The default is the parent of the default base installation directory of the GlassFish Server software for the DAS. This default is useful only if GlassFish Server is installed in the same location on all hosts.

--nodedir
The path to the directory that is to contain GlassFish Server instances that are created on the node. The default is as-install/nodes, where as-install is the base installation
directory of the GlassFish Server software on the host. If a relative path is specified, the path is relative to the as-install directory.

`--sshport`
The port to use for SSH connections to this node’s host. The default is 22. If the `--nodehost` option is set to localhost-domain, the `--sshport` option is ignored.

`--sshuser`
The user on this node’s host that is to run the process for connecting to the host through SSH. The default is the user that is running the DAS process. To ensure that the DAS can read this user’s SSH private key file, specify the user that is running the DAS process. If the `--nodehost` option is set to localhost-domain, the `--sshuser` option is ignored.

`--sshkeyfile`
The absolute path to the SSH private key file for user that the `--sshuser` option specifies. This file is used for authentication to the sshd daemon on the node’s host.

---

**Note:** GlassFish Server also supports password authentication through the AS_ADMIN_SSHPASSWORD entry in the password file. The password file is specified in the `--passwordfile` option of the asadmin(1M) utility.

---

If the SSH private key file is protected by a passphrase, the password file must contain the AS_ADMIN_SSHPASSPASSPHRASE entry.

The path to the key file must be reachable by the DAS and the key file must be readable by the DAS.

The default is the a key file in the user’s .ssh directory. If multiple key files are found, the subcommand uses the following order of preference:

1. id_rsa
2. id_dsa
3. identity

`--force`
Specifies whether the node is created in the DAS configuration even if validation of the node’s parameters fails. To validate a node’s parameters, the DAS must be able to contact the node’s host through SSH. Possible values are as follows:

false
The node is not created if validation of the node’s parameters fails (default).

ture
The node is created even if validation of the node’s parameters fails.

`--install`
Specifies whether the subcommand shall install the GlassFish Server software on the host that the node represents.

Possible values are as follows:

false
The subcommand shall not install the GlassFish Server software on the host (default).
true
The subcommand shall install the GlassFish Server software on the host.

--archive
The absolute path to the archive file of the GlassFish Server software that is to be installed. If this option is omitted and the --install is true, the subcommand creates a ZIP archive of the GlassFish Server software from the installation where this subcommand is run. The archive does not contain the domains directory or the nodes directory.

Operands

node-name
The name of the node.

The name must meet the following requirements:

- The name may contain only ASCII characters.
- The name must start with a letter, a number, or an underscore.
- The name may contain only the following characters:
  - Lowercase letters
  - Uppercase letters
  - Numbers
  - Hyphen
  - Period
  - Underscore
- The name must be unique in the domain and must not be the name of another node, a cluster, a named configuration, or a GlassFish Server instance.
- The name must not be domain, server, or any other keyword that is reserved by GlassFish Server.

Examples

Example 1 Creating a Node
This example creates the node adc for the host adc.example.com. By default, the parent of the base installation directory of the GlassFish Server software is /export/glassfish3.

```
asadmin> create-node-ssh
--nodehost adc.example.com
--installdir /export/glassfish3 adc
```
Command create-node-ssh executed successfully.

Example 2 Forcing the Creation of a Node
This example forces the creation of node egl for the host eghost.example.com. The node is created despite the failure of the DAS to contact the host eghost.example.com to validate the node's parameters.

```
asadmin> create-node-ssh --force --nodehost eghost.example.com egl
Warning: some parameters appear to be invalid.
```
Could not connect to host eghost.example.com using SSH.
There was a problem while connecting to eghost.example.com:22
eghost.example.com
Continuing with node creation due to use of --force.

Command create-node-ssh executed successfully.

**Exit Status**

0
command executed successfully

1
error in executing the command

**See Also**

create-node-config(1), create-node-dcom(1), delete-node-ssh(1), install-node(1),
install-node-ssh(1), list-nodes(1), ping-node-ssh(1), setup-ssh(1),
uninstall-node(1), uninstall-node-ssh(1), update-node-ssh(1)

asadmin(1M)
create-password-alias

creates a password alias

Synopsis

create-password-alias [--help]
aliasname

Description

The create-password-alias subcommand creates an alias for a password. An alias is a token of the form ${ALIAS=aliasname}. The password that corresponds to the alias name is stored in an encrypted form.

The create-password-alias subcommand can be run interactively or noninteractively.

- When run interactively, the subcommand prompts the user for the alias password and to confirm the alias password.
- When run noninteractively, the subcommand reads the alias password from a file that is passed through the --passwordfile option of the asadmin(1M) utility. The file must contain an entry of the form AS_ADMIN_ALIASPASSWORD=alias-password, where alias-password is the alias password. The noninteractive form of this command is suitable for use in scripts.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

aliasname
Your choice of name for the password alias.

Examples

Example 1 Creating a Password Alias Interactively
This example creates the password alias jmspassword-alias interactively.

asadmin> create-password-alias jmspassword-alias
Enter the alias password>
Enter the alias password again>
Command create-password-alias executed successfully.
Example 2 Creating a Password Alias Noninteractively
This example uses the `--passwordfile` option of the `asadmin` utility to create the password alias `winuser` noninteractively.

```
$ asadmin --passwordfile aspwfile.txt create-password-alias winuser
Command create-password-alias executed successfully.
```

The file `aspwfile.txt` contains the following entry to specify the alias password:
```
AS_ADMIN_ALIASPASSWORD=sp@rky
```

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-password-alias(1), list-password-aliases(1), update-password-alias(1)
asadmin(1M)
create-profiler

creates the profiler element

Synopsis

create-profiler [--help] [--target target_name] [--classpath classpath] [--nativelibpath native_library_path] [--enabled=true] [--property(name=value)[(name=value)]] profiler_name

Description

The create-profiler subcommand creates the profiler element. A server instance is tied to the profiler by the profiler element in the Java configuration. Only one profiler exists at a time. If you attempt to create a profiler while one already exists, an error message is displayed.

For changes to take effect, the server must restarted.
This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target on which you are creating a profiler. Valid values are

server
Creates the profiler for the default server instance. This is the default value.

configuration_name
Creates the profiler for the named configuration

cluster_name
Creates the profiler for every server instance in the cluster

instance_name
Creates the profiler for a particular server instance

--classpath
Java classpath string that specifies the classes needed by the profiler.

--nativelibpath
This path is automatically constructed to be a concatenation of the GlassFish Server installation relative path for its native shared libraries, standard JRE native library path, the shell environment setting (LD_LIBRARY_PATH on UNIX) and any path that may be specified in the profile element.

--enabled
Profiler is enabled by default.
--property
Name/value pairs of provider-specific attributes.

Operands

profiler_name
Name of the profiler.

Examples

Example 1  Creating a Profiler
This example creates a profiler named sample_profiler.

asadmin> create-profiler --classpath /home/appserver/
   --nativelibpath /u/home/lib --enabled=false
   --property defaultuser=admin:password=adminadmin sample_profiler
Created Profiler with id = sample_profiler

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-profiler(1)
asadmin(1M)
create-protocol

adds a new protocol

Synopsis

create-protocol [--help]
[--securityenabled={false|true}]
 [--target target]
 protocol-name

Description

The create-protocol subcommand creates a protocol. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--securityenabled
If set to true, the protocol runs SSL. You can turn SSL2 or SSL3 ON or OFF and set ciphers using an ssl element. The security setting globally enables or disables SSL by making certificates available to the server instance. The default value is false.

--target
Creates the protocol only on the specified target. Valid values are as follows:

server
Creates the protocol on the default server instance. This is the default value.

configuration-name
Creates the protocol in the specified configuration.

cluster-name
Creates the protocol on all server instances in the specified cluster.

standalone-instance-name
Creates the protocol on the specified standalone server instance.

Operands

protocol-name
The name of the protocol.

Examples

Example 1 Creating a Protocol
The following command creates a protocol named http-1 with security enabled:

asadmin> create-protocol --securityenabled=true http-1
Command create-protocol executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-protocol(1), list-protocols(1), create-network-listener(1)
asadmin(1M)
create-protocol-filter

adds a new protocol filter

Synopsis

create-protocol-filter [--help]
--protocol protocol-name
--classname class-name
[--target server]
protocol-filter-name

Description

The create-protocol-filter subcommand creates a protocol filter for a protocol. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--protocol
The name of the associated protocol.

--classname
The fully qualified name of the Java class that implements the protocol filter.

--target
Creates the protocol filter only on the specified target. Valid values are as follows:

server
Creates the protocol filter on the default server instance. This is the default value.

configuration-name
Creates the protocol filter in the specified configuration.

cluster-name
Creates the protocol filter on all server instances in the specified cluster.

standalone-instance-name
Creates the protocol filter on the specified standalone server instance.

Operands

protocol-filter-name
The name of the protocol filter.
Examples

Example 1 Creating a Protocol Filter
The following command creates a protocol filter named http1-filter:

```bash
asadmin> create-protocol-filter --protocol http1
Command create-protocol-filter executed successfully.
```

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-protocol-filter(1), list-protocol-filters(1), create-protocol(1)
asadmin(1M)
create-protocol-finder

adds a new protocol finder

Synopsis

create-protocol-finder [--help]
--protocol protocol-name
--targetprotocol target-protocol-name
--classname class-name
[--target server]
protocol-finder-name

Description

The create-protocol-finder subcommand creates a protocol finder for a protocol.
This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--protocol
The name of the associated protocol.

--targetprotocol
The name of the target protocol.

--classname
The fully qualified name of the Java class that implements the protocol finder.

--target
Creates the protocol finder only on the specified target. Valid values are as follows:

server
Creates the protocol finder on the default server instance. This is the default value.

configuration-name
Creates the protocol finder in the specified configuration.

cluster-name
Creates the protocol finder on all server instances in the specified cluster.

standalone-instance-name
Creates the protocol finder on the specified standalone server instance.

Operands

protocol-finder-name
The name of the protocol finder.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-protocol-finder(1), list-protocol-finders(1)
asadmin(1M)
create-resource-adapter-config

creates the configuration information for the connector module

Synopsis

create-resource-adapter-config [--help] [--threadpoolid threadpool]
[--objecttype object-type] [--property {property-name=value}|:{name=value}]*
raname

Description

The create-resource-adapter-config subcommand creates configuration information for the connector module. This subcommand can be run before deploying a resource adapter, so that the configuration information is available at the time of deployment. The resource adapter configuration can also be created after the resource adapter is deployed. In this case, the resource adapter is restarted with the new configuration. You must first create a thread pool, using the create-threadpool subcommand, and then identify that thread pool value as the ID in the --threadpoolid option.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option has been deprecated.

--threadpoolid
The thread pool ID from which the work manager gets the thread. This option takes only one thread pool ID.

--objecttype
The default is user.

--property
Keyword-value pairs that specify additional configuration properties of the resource adapter Java bean. The keyword-value pairs are separated by a colon (:). The properties are the names of setter methods of the class that is referenced by the resourceadapter-class element in the ra.xml file.

Operands

raname
Indicates the connector module name. It is the value of the resource-adapter-name in the domain.xml file.
Examples

**Example 1 Creating a Resource Adapter Configuration**
This example creates a resource adapter configuration for ra1.

```
asadmin> create-resource-adapter-config --property foo=bar --threadpoolid mycustomerthreadpool ra1
Command create-resource-adapter-config executed successfully
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-threadpool(1), delete-resource-adapter-config(1),
list-resource-adapter-configs(1)
asadmin(1M)
create-resource-ref

creates a reference to a resource

Synopsis

create-resource-ref [--help] [--target target] [--enabled={false|true}] reference_name

Description

The create-resource-ref subcommand creates a reference from a cluster or an unclustered server instance to a previously created resource, for example, a JDBC resource created by using the create-jdbc-resource subcommand. This effectively results in the resource being made available in the JNDI tree of the instance or cluster.

The target instance or instances making up the cluster need not be running or available for this subcommand to succeed. If one or more instances are not available, they will receive the new resource the next time they start.

Note: A resource-ref can only be created for bindable resources, such as a jdbc-resource, connector-resource, admin-object-resource, mail-resource, custom-resource, or jndi-resource.

A jdbc-connection-pool or a connector-connection-pool are not referred to directly by applications. Instead, they are referred to through a jdbc-resource or connector-resource, respectively.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target for which you are creating the resource reference. Valid targets are as follows:

server
Creates the resource reference for the default server instance. This is the default target.

cluster_name
Creates the resource reference for every server instance in the cluster.

instance_name
Creates the resource reference for the named unclustered server instance.
--enabled
Indicates whether the resource should be enabled. This value will take effect only if the resource is enabled at the global level. The default is true.

Operands

reference_name
The name or JNDI name of the resource.

Examples

Example 1 Creating a Reference to a JMS Destination Resource
This example creates a reference to the JMS destination resource jms/Topic on the cluster Cluster1.

asadmin> create-resource-ref --target Cluster1 jms/Topic
resource-ref jms/Topic created successfully.
Command create-resource-ref executed successfully.

Exit Status

0
subcommand executed successfully
1
error in executing the subcommand

See Also

delete-resource-ref(1), list-resource-refs(1)
asadmin(1M)
create-schedule

creates a new schedule

Synopsis

create-schedule [--help]
[--hour hour] [--minute minute] [--second second]
[--dayofmonth day-of-month]
[--dayofweek day-of-week]
[--month month]
[--year year]
schedule-name

Description

The create-schedule subcommand creates a schedule representing a specific date and time or a recurring interval.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--hour
--minute
--second
Together these options specify a time using a 24-hour clock. These options accept the following types of values:

- A number. For hours, the numbers 0 through 23 are valid; for minutes and seconds, the numbers 0 through 59 are valid.
- A dash-separated number range; for example: 0-11.
- A comma-separated list of numbers, number ranges or mixture of the two; for example: 0-3,8-11,23.
- A repeating increment in the form start/interval. For example, the increment 0/5 indicates values starting at 0 (zero) and continuing every five thereafter (5, 10, 15 and so on) up to a maximum of 23 for hours or 59 for minutes or seconds.
- An asterisk (*) to indicate all hours, minutes or seconds.

The default value is 0 (zero) for all of these options, indicating the time 00:00:00 (midnight).

--dayofmonth
Specifies the day or days based on days in a month. This option accepts the following types of values:

- A positive number 1 through 31, representing the days of the month.
A negative number -7 through -1, representing days back from the end of the month. -1 represents the day before the last day of the month, and -7 represents the seventh day before the last day of the month.

The value Last, representing the last day of the month.

A dash-separated number range (including Last as a number); for example: 8-14 or "-6-Last".

An ordinal day in the form ordinal day, where ordinal is one of 1st, 2nd, 3rd, 4th, 5th or Last and day is one of Sun, Mon, Tue, Wed, Thu, Fri or Sat. Note that a space is required between the ordinal and the day. Consequently, the value must be enclosed in quotes ("); for example: "2nd Mon".

A comma-separated list of numbers, the value last, number ranges, ordinal days, or mixture of the four; for example: 1,8,15,22,Last or "1st Mon,3rd Mon".

An asterisk (*) to indicate all days.

A value that contains spaces or begins with a negative number must be enclosed in quotes (").

The default value is * (asterisk).

---

**Note:** The dayofmonth and dayofweek options are combined to specify scheduled days as follows:

- **Both options set to * (asterisk)** — Neither option restricts days, so every day is a scheduled day.
- **One option set to * (asterisk)** — The option set to * does not restrict days. Scheduled days are specified by the other option.
- **Neither option set to * (asterisk)** — Both options restrict days. Scheduled days are those that match either option.

---

**--dayofweek**

Specifies the day or days based on days in a week. This option accepts the following types of values:

- A number 0 through 7, representing the days of the week beginning with Sunday. The number zero and seven both represent Sunday.
- A day abbreviation; one of: Sun, Mon, Tue, Wed, Thu, Fri or Sat.
- A dash-separated number range or day-abbreviation range; for example: 1-5 or Mon-Fri.
- A comma-separated list of numbers, day abbreviations, ranges, or mixture of the three; for example: Sun, Thu-Fri.
- An asterisk (*) to indicate all days.

The default value is * (asterisk).
Note: The dayofmonth and dayofweek options are combined to specify scheduled days as follows:

- Both options set to * (asterisk) — Neither option restricts days, so every day is a scheduled day.
- One option set to * (asterisk) — The option set to * does not restrict days. Scheduled days are specified by the other option.
- Neither option set to * (asterisk) — Both options restrict days. Scheduled days are those that match either option.

--month
Specifies the month or months. This option accepts the following types of values:

- A number 1 through 12, representing the months of the year beginning with January.
- A month abbreviation; one of: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov or Dec.
- A dash-separated number range or month-abbreviation range; for example: 1-3 or Jan-Mar.
- A comma-separated list of numbers, month abbreviations, ranges, or mixture of the three; for example: 1,4,7,10 or Jan,Apr,Jul,Oct.
- An asterisk (*) to indicate all months.

The default value is * (asterisk).

--year
Specifies the year or years. This option accepts the following types of values:

- A four-digit number, representing a single year.
- A dash-separated range of four-digit numbers, representing a range of years; for example: 2011-2014.
- A comma-separated list of four-digit numbers, ranges, or mixture of the two; for example: 2011,2013,2015.
- An asterisk (*) to indicate all years.

The default value is * (asterisk).

Operands

schedule-name
Specifies the name of the schedule to create.

Examples

Example 1 Creating a Quarterly Schedule
This example creates the schedule quarterly, which specifies the first day of every quarter-year at midnight:

```
asadmin> create-schedule --month Jan, Apr, Jul, Oct --dayofmonth 1 quarterly
Command create-schedule executed successfully.
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-schedule(1), list-schedules(1)

asadmin(1M)
create-service

configures the starting of a DAS or a GlassFish Server instance on an unattended boot

Synopsis

create-service [--help] [--name service-name]
[--serviceproperties service-properties]
[--dry-run={false|true}] [--force={false|true}]
[--serviceuser service-user]
[--domaindir domain-dir]
[--nodedir node-dir] [--node node]
[domain-or-instance-name]

Description

The create-service subcommand configures the starting of a domain administration server (DAS) or a GlassFish Server instance on an unattended boot on Windows, Linux, and Oracle Solaris systems.

If no operand is specified and the domains directory contains only one domain, the subcommand configures the starting of the DAS for the default domain. If no operand is specified and the domains directory contains multiple domains, an error occurs.

If the operand specifies an instance, the create-service subcommand does not contact the domain administration server (DAS) to determine the node on which the instance resides. To determine the node on which the instance resides, the subcommand searches the directory that contains the node directories. If multiple node directories exist, the node must be specified as an option of the subcommand.

The subcommand contains internal logic to determine whether the supplied operand is a DAS or an instance.

This subcommand is supported in local mode only.

Behavior of create-service on Windows Systems

On Windows systems, the create-service subcommand creates a Windows service to represent the DAS or instance. The service is created in the disabled state. After this subcommand creates the service, you must use the Windows Services Manager or the Windows Services Wrapper to start, stop, uninstall, or install the service.

On Windows systems, this subcommand must be run as the OS-level administrator user.

The subcommand creates the following Windows Services Wrapper files for the service in the domain-dir\bin directory or the instance-dir\bin directory:

- Configuration file: service-nameService.xml
- Executable file: service-nameService.exe

On Windows systems, this subcommand requires the Microsoft .NET Framework (http://www.microsoft.com/net/). Otherwise, the subcommand fails.
Behavior of `create-service` on Linux Systems

On Linux systems, the `create-service` subcommand creates a System-V-style initialization script `/etc/init.d/GlassFish_domain-or-instance-name` and installs a link to this script in the `/etc/rc?.d` directories. After this subcommand creates the script, you must use this script to start, stop, or restart the domain or instance.

On Linux systems, this subcommand must be run as the OS-level root user.

Behavior of `create-service` on Oracle Solaris Systems

On Oracle Solaris systems, the `create-service` subcommand creates a Service Management Facility (SMF) service to represent the DAS or instance. The service is created in the disabled state. After this subcommand creates the service, you must use SMF commands to start, enable, disable, delete, or stop the service. For more information about SMF, see the following documentation for the Oracle Solaris operating system:

- "Managing Services (Overview)" in *System Administration Guide: Basic Administration*
- "Managing Services (Tasks)" in *System Administration Guide: Basic Administration*

On Oracle Solaris systems, this subcommand must be run as the OS-level user with superuser privileges. The configuration file for the DAS or instance must be stored in a directory to which the superuser has access and cannot be stored on a network file system. The service that is created is controlled by the OS-level user who owns the directory where the configuration of the DAS or instance resides.

On Oracle Solaris systems, the manifest file is created in the following directory by default:

```
/var/svc/manifest/application/GlassFish/domain-or-instance-name_domain-or-instance-root-dir
```

To run this subcommand, you must have `solaris.smf.*` authorization. For information about how to grant authorizations to users, see the `useradd(1M)` and `usermod(1M)` man pages.

To run these commands as non-root user, the system administrator must be contacted so that the relevant authorizations are granted. You must also ensure that the following conditions are met:

- Oracle Solaris 10 administration commands such as `svccfg(1M)`, `svcs(1)`, and `auths(1)` are available through the `PATH` statement, so that these commands can be executed. A simple test to do so is to run the command `which svccfg` in the shell.

- You must have write permission for the path `/var/svc/manifest/application/GlassFish`. Usually, the superuser has write permission to this path.

If you delete a service that you created by using the `create-service` subcommand, you must delete the directory that contains the manifest file and the entire contents of the directory. Otherwise, an attempt to re-create the service by using the `create-service` subcommand fails. The Oracle Solaris command `svccfg` does not delete this directory.
create-service

Options

--help
-?
Displays the help text for the subcommand.

--name
(Windows and Oracle Solaris systems only) The name of the service that you will use when administering the service through Oracle Solaris SMF commands or the service management features of the Windows operating system. The default is the name of the domain or instance that is specified as the operand of this subcommand.

--serviceproperties
Specifies a colon(:)-separated list of various properties that are specific to the service.
To customize the display name of the service in the Windows Service list, set the DISPLAY_NAME property to the required name.
For Oracle Solaris 10 systems, if you specify net_privaddr, the service's processes will be able to bind to the privileged ports (<1024) on the platform. You can bind to ports< 1024 only if the owner of the service is superuser, otherwise, this is not allowed.

--dry-run
-n
Previews your attempt to create a service. Indicates issues and the outcome that will occur if you run the command without using the --dry-run option. Nothing is actually configured. Default is false.

--force
Specifies whether the service is created even if validation of the service fails.
Possible values are as follows:

  true
  The service is created even if validation of the service fails.

  false
  The service is not created (default).

--serviceuser
(Linux systems only) The user that is to run the GlassFish Server software when the service is started. The default is the user that is running the subcommand. Specify this option if the GlassFish Server software is to be run by a user other than the root user.

--domaindir
The absolute path of the directory on the disk that contains the configuration of the domain. If this option is specified, the operand must specify a domain.

--nodedir
Specifies the directory that contains the instance’s node directory. The instance’s files are stored in the instance’s node directory. The default is as-install/nodes. If this option is specified, the operand must specify an instance.

--node
 Specifies the node on which the instance resides. This option may be omitted only if the directory that the --nodedir option specifies contains only one node directory. Otherwise, this option is required. If this option is specified, the operand must specify an instance.
Operands

*domain-or-instance-name*

The name of the domain or instance to configure. If no operand is specified, the default domain is used.

Examples

**Example 1  Creating a Service on a Windows System**

This example creates a service for the default domain on a system that is running Windows.

```plaintext
asadmin> create-service
Found the Windows Service and successfully uninstalled it.
The Windows Service was created successfully. It is ready to be started. Here are the details:
  ID of the service: domain1
  Display Name of the service: domain1 GlassFish Server
  Domain Directory: C:\glassfish3\glassfish\domains\domain1
  Configuration file for Windows Services Wrapper: C:\glassfish3\glassfish\domains\domain1\bin\domain1Service.xml
The service can be controlled using the Windows Services Manager or you can use the Windows Services Wrapper instead:
  Start Command:  C:\glassfish3\glassfish\domains\domain1\bin\domain1Service.exe start
  Stop Command:   C:\glassfish3\glassfish\domains\domain1\bin\domain1Service.exe stop
  Uninstall Command:  C:\glassfish3\glassfish\domains\domain1\bin\domain1Service.exe uninstall
  Install Command:  C:\glassfish3\glassfish\domains\domain1\bin\domain1Service.exe install
This message is also available in a file named PlatformServices.log in the domain's root directory
Command create-service executed successfully.
```

**Example 2  Creating a Service on a Linux System**

This example creates a service for the default domain on a system that is running Linux.

```plaintext
asadmin> create-service
Found the Linux Service and successfully uninstalled it.
The Service was created successfully. Here are the details:
  Name of the service: domain1
  Type of the service: Domain
  Configuration location of the service:/etc/init.d/GlassFish_domain1
  User account that will run the service: root
You have created the service but you need to start it yourself. Here are the most typical Linux commands of interest:
* /etc/init.d/GlassFish_domain1 start
* /etc/init.d/GlassFish_domain1 stop
* /etc/init.d/GlassFish_domain1 restart
For your convenience this message has also been saved to this file:
```
Example 3 Creating a Service on an Oracle Solaris System
This example creates a service for the default domain on a system that is running Oracle Solaris.

asadmin> create-service
The Service was created successfully. Here are the details:
Name of the service:application/GlassFish/domain1
Type of the service:Domain
Configuration location of the service:/home/gfuser/glassfish-installations
/glassfish3/glassfish/domains
Manifest file location on the system:/var/svc/manifest/application
/GlassFish/domain1_home_gfuser_glassfish-installations_glassfish3
_glassfish_domains/Domain-service-smf.xml.
You have created the service but you need to start it yourself.
Here are the most typical Solaris commands of interest:
* /usr/bin/svcs -a | grep domain1 // status
* /usr/sbin/svcadm enable domain1 // start
* /usr/sbin/svcadm disable domain1 // stop
* /usr/sbin/svccfg delete domain1 // uninstall
Command create-service executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

asadmin(1M)
auths(1), svcs(1)
svccfg(1M), useradd(1M), usermod(1M)
"Managing Services (Overview)" in System Administration Guide: Basic Administration,
"Managing Services (Tasks)" in System Administration Guide: Basic Administration
Microsoft .NET Framework (http://www.microsoft.com/net/)
create-ssl

creates and configures the SSL element in the selected HTTP listener, IIOP listener, or IIOP service

Synopsis

create-ssl [--help]
[--target target]
--type listener_or_service_type
--certname cert_name
[--ssl2enabled={false|true}] [--ssl2ciphers ss12ciphers]
[--ssl3enabled={true|false}] [--tlsenabled={true|false}]
[--ssl3tlsciphers ss13tlsciphers]
[--tlsrollbackenabled={true|false}]
[--clientauthenabled={false|true}]
[listener_id]

Description

The create-ssl subcommand creates and configures the SSL element in the selected HTTP listener, IIOP listener, or IIOP service to enable secure communication on that listener/service.

This subcommand is supported in remote mode only.

Options

If an option has a short option name, then the short option precedes the long option name. Short options have one dash whereas long options have two dashes.

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target on which you are configuring the ssl element. The following values are valid:

server
Specifies the server in which the iiop-service or HTTP/IIOP listener is to be configured for SSL.

config
Specifies the configuration that contains the HTTP/IIOP listener or iiop-service for which SSL is to be configured.

cluster
Specifies the cluster in which the HTTP/IIOP listener or iiop-service is to be configured for SSL. All the server instances in the cluster will get the SSL configuration for the respective listener or iiop-service.
instance
Specifies the instance in which the HTTP/IIOP listener or iiop-service is to be configured for SSL.

--type
The type of service or listener for which the SSL is created. The type can be:

- network-listener
- http-listener
- iiop-listener
- iiop-service
- jmx-connector

When the type is iiop-service, the ssl-client-config along with the embedded ssl element is created in domain.xml.

--certname
The nickname of the server certificate in the certificate database or the PKCS#11 token. The format of the name in the certificate is tokenname:nickname. For this property, the tokenname: is optional.

--ssl2enabled
Set this property to true to enable SSL2. The default value is false. If both SSL2 and SSL3 are enabled for a virtual server, the server tries SSL3 encryption first. In the event SSL3 encryption fails, the server then tries SSL2 encryption.

--ssl2ciphers
A comma-separated list of the SSL2 ciphers to be used. Ciphers not explicitly listed will be disabled for the target, even if those ciphers are available in the particular cipher suite you are using. If this option is not used, all supported ciphers are assumed to be enabled. Allowed values are:

- rc4
- rc4export
- rc2
- rc2export
- idea
- des
- desede3

--ssl3enabled
Set this property to false to disable SSL3. The default value is true. If both SSL2 and SSL3 are enabled for a virtual server, the server tries SSL3 encryption first. In the event SSL3 encryption fails, the server then tries SSL2 encryption.

--tlsenabled
Set this property to false to disable TLS. The default value is true. It is good practice to enable TLS, which is a more secure version of SSL.

--ssl3tlsciphers
A comma-separated list of the SSL3 and/or TLS ciphers to be used. Ciphers not explicitly listed will be disabled for the target, even if those ciphers are available in the
particular cipher suite you are using. If this option is not used, all supported ciphers are assumed to be enabled. Allowed values are:

- `SSL_RSA_WITH_RC4_128_MD5`
- `SSL_RSA_WITH_3DES_EDE_CBC_SHA`
- `SSL_RSA_WITH_DES_CBC_SHA`
- `SSL_RSA_EXPORT_WITH_RC4_40_MD5`
- `SSL_RSA_WITH_NULL_MD5`
- `SSL_RSA_WITH_RC4_128_SHA`
- `SSL_RSA_WITH_NULL_SHA`

**--tlsrollbackenabled**
Set to true (default) to enable TLS rollback. TLS rollback should be enabled for Microsoft Internet Explorer 5.0 and 5.5. This option is only valid when `-tlsenabled=true`.

**--clientauthenabled**
Set to true if you want SSL3 client authentication performed on every request independent of ACL-based access control. Default value is false.

**Operands**

*listener_id*
The ID of the HTTP or IIOP listener for which the SSL element is to be created. The *listener_id* is not required if the --type is iiop-service.

**Examples**

**Example 1  Creating an SSL element for an HTTP listener**
The following example shows how to create an SSL element for an HTTP listener named http-listener-1.

```
asadmin> create-ssl
--type http-listener
--certname sampleCert http-listener-1
Command create-ssl executed successfully.
```

**Exit Status**

0  
subcommand executed successfully

1  
error in executing the subcommand

**See Also**

*delete-ssl*(1)
*asadmin*(1M)
create-system-properties

adds one or more system property elements that can be referenced elsewhere in the configuration.

Synopsis

create-system-properties [--help]
[--target target]
[name=value] [:name=value]*

Description

The create-system-properties subcommand adds or updates system properties that can be referenced elsewhere on the server.

GlassFish Server provides hooks where tokens (system properties) can be specified. Because GlassFish Server does not have multiple server elements, you can specify a particular token at any level. When a domain supports multiple servers, the override potential can be exploited. When a domain is started or restarted, all <system-property> elements are resolved and available to the Java Virtual Machine by using the System.setProperty() call on each of them (with its name and value derived from the corresponding attributes of the element). This is analogous to sending the elements as -D parameters on the Java command line.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
The target on which you are creating the system properties.

Operands

target
The valid targets for this subcommand are instance, cluster, configuration, domain, and server. Server is the default option. Valid values are:

server
Creates the properties on the default server instance. This is the default value.

domain
Creates the properties for all server instances in the default domain.

configuration_name
Creates the properties in the specified configuration.

cluster_name
Creates the properties on all server instances in the specified cluster.


instance_name
Creates the properties on a specified server instance.

name=value
The name value pairs of the system properties to add to the specified target. Multiple system properties must be separated by a : (colon). If a : (colon) appears in the name or value of a system property, it must be escaped with a \ (blackslash). If any system properties were previously defined, they are updated with the new values.

Examples

Example 1 Creating System Properties
This example creates a system property associated with an HTTP listener on a server instance named myserver.

asadmin> create-system-properties --target myserver http-listener-port=1088
Command create-system-properties executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-system-property(1), list-system-properties(1)

asadmin(1M)
create-threadpool

adds a thread pool

Synopsis

```
```

Description

The `create-threadpool` subcommand creates a thread pool with the specified name. You can specify maximum and minimum number of threads in the pool, the quantity of messages, and the idle timeout of a thread. The created thread pool can be used for servicing IIOP requests and for resource adapters to service work management requests. A thread pool can be used in multiple resource adapters.

This subcommand is supported in remote mode only.

Options

--help

-?

Displays the help text for the subcommand.

--target

This option specifies the target on which you are creating the thread pool.

Valid values are as follows:

- **server**
  Creates the thread pool for the default GlassFish Server instance `server` and is the default value

- **configuration-name**
  Creates the thread pool for the named configuration.

- **cluster-name**
  Creates the thread pool for every instance in the cluster.

- **instance-name**
  Creates the thread pool for a particular instance.

--maxthreadpoolsize

Specifies the maximum number of threads the pool can contain. Default is 5.

--minthreadpoolsize

Specifies the minimum number of threads in the pool. These are created when the thread pool is instantiated. Default is 2.
create-threadpool

--idletimeout
Specifies the amount of time in seconds after which idle threads are removed from the pool. Default is 900.

--maxqueuesize
Specifies the maximum number of messages that can be queued until threads are available to process them for a network listener or IIOP listener. A value of -1 specifies no limit. Default is 4096.

--workqueues
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

Operands

threadpool-id
An ID for the work queue, for example, threadpool-1.

Examples

Example 1 Creating a Thread Pool
This command creates a new thread pool called threadpool-1.

asadmin> create-threadpool --maxthreadpoolsize 100
   --minthreadpoolsize 20 --idletimeout 2 threadpool-1
Command create-threadpool executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-threadpool(1), list-threadpools(1)
asadmin(1M)
create-transport

adds a new transport

Synopsis

create-transport [--help]
[--acceptorthreads acceptor-threads]
[--buffersizebytes buffer-size]
[--bytebuffertype byte-buffer-type]
[--classname class-name]
[--displayconfiguration={false|true}]
[--enablesnoop={false|true}]
[--idlekeytimeoutseconds idle-key-timeout]
[--maxconnectionscount max-connections]
[--readtimeoutmilliseconds read-timeout]
[--writetimeoutmilliseconds write-timeout]
[--selectorkeyhandler selection-key-handler]
[--selectorpolltimeoutmilliseconds selector-poll-timeout]
[--tcpenabledelay={false|true}]
[--target target]
transport-name

Description

The create-transport subcommand creates a transport for a network listener. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--acceptorthreads
The number of acceptor threads for the transport. The recommended value is the number of processors in the machine. The default value is 1.

--buffersizebytes
The size, in bytes, of the buffer to be provided for input streams created by the network listener that references this transport. The default value is 8192.

--bytebuffertype
The type of the buffer to be provided for input streams created by a network-listener. Allowed values are HEAP and DIRECT. The default value is HEAP.

--classname
The fully qualified name of the Java class that implements the transport. The default is com.sun.grizzly.TCPSelectorHandler.

--displayconfiguration
If true, flushes the internal network configuration to the server log. Useful for debugging, but reduces performance. The default is false.
--enablesnoop
If true, writes request/response information to the server log. Useful for debugging, but reduces performance. The default is false.

--idlekeytimeoutseconds
The idle key timeout. The default is 30 seconds.

--maxconnectionscount
The maximum number of connections for the network listener that references this transport. A value of -1 specifies no limit. The default value is 4096.

--readtimeoutmillis
The amount of time the server waits during the header and body parsing phase. The default is 30000 milliseconds, or 30 seconds.

--writetimeoutmillis
The amount of time the server waits before considering the remote client disconnected when writing the response. The default is 30000 milliseconds, or 30 seconds.

--selectionkeyhandler
The name of the selection key handler associated with this transport. There is no default.

--selectorpolltimeoutmillis
The number of milliseconds a NIO Selector blocks waiting for events (user requests). The default value is 1000 milliseconds.

--tcpnodelay
If true, the default, enables TCP_NODELAY (also called Nagle's algorithm). The default is false.

--target
Creates the transport only on the specified target. Valid values are as follows:

  server
  Creates the transport on the default server instance. This is the default value.

  configuration-name
  Creates the transport in the specified configuration.

  cluster-name
  Creates the transport on all server instances in the specified cluster.

  standalone-instance-name
  Creates the transport on the specified standalone server instance.

Operands

transport-name
The name of the transport.

Examples

Example 1 Creating a Transport
The following command creates a transport named http1-trans that uses a non-default number of acceptor threads:
asadmin> create-transport --acceptorthreads 100 http1-trans
Command create-transport executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-transport(1), list-transports(1), create-network-listener(1)
asadmin(1M)
create-virtual-server

creates the named virtual server

Synopsis

create-virtual-server [--help]
--hosts hosts
[--http-listeners http-listeners]
[--network-listeners network-listeners]
[--default-web-module default-web-module]
[--state={on|off}]
[--logfile log-file]
[--property (name=value)[:name=value]*]
[--target target]
virtual-server-id

Description

The create-virtual-server subcommand creates the named virtual server. Virtualization in the GlassFish Server allows multiple URL domains to be served by a single HTTP server process that is listening on multiple host addresses. If the application is available at two virtual servers, they still share the same physical resource pools.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--hosts
A comma-separated (,) list of values allowed in the host request header to select the current virtual server. Each virtual server that is configured to the same connection group must have a unique host for that group.

--http-listeners
A comma-separated (,) list of HTTP listener IDs. Required only for a virtual server that is not the default virtual server. HTTP listeners are converted to network listeners. This option is deprecated but maintained for backward compatibility. Use --network-listeners instead. If --network-listeners is used, this option is ignored.

--network-listeners
A comma-separated (,) list of network listener IDs. Required only for a virtual server that is not the default virtual server.

--default-web-module
The standalone web module associated with this virtual server by default.

--state
Determines whether a virtual server is active (on) or inactive (off or disabled). Default is on. When inactive, the virtual server does not service requests.
--logfile
Name of the file where log entries for this virtual server are to be written. By default, this is the server log. The file and directory in which the access log is kept must be writable by the user account under which the server runs.

--property
Optional property name/value pairs for configuring the virtual server. The following properties are available:

**sso-max-inactive-seconds**
Specifies the number of seconds after which a user’s single sign-on record becomes eligible for purging if no client activity is received. Since single sign-on applies across several applications on the same virtual server, access to any of the applications keeps the single sign-on record active. The default value is 300 seconds (5 minutes). Higher values provide longer single sign-on persistence for users, but at the expense of more memory use on the server.

**sso-reap-interval-seconds**
Specifies the number of seconds between purges of expired single sign-on records. The default value is 60.

**setCacheControl**

**allowLinking**
If the value of this property is true, resources that are symbolic links will be served for all web applications deployed on this virtual server. Individual web applications may override this setting by using the property allowLinking under the sun-web-app element in the sun-web.xml file:

```
<sun-web-app>
  <property name="allowLinking" value="[true|false]"/>
</sun-web-app>
```

The default value is true.

**accessLogWriteInterval**
Indicates the number of seconds before the log will be written to the disk. The access log is written when the buffer is full or when the interval expires. If the value is 0 (zero), then the buffer is always written even if it is not full. This means that each time the server is accessed, the log message is stored directly to the file.

**accessLogBufferSize**
Specifies the size, in bytes, of the buffer where access log calls are stored.

**allowRemoteAddress**
This is a comma-separated list of regular expression patterns to which the remote client’s IP address is compared. If this property is specified, the remote address must match for this request to be accepted. If this property is not specified, all requests will be accepted unless the remote address matches a denyRemoteAddress pattern. The default value for this property is null.

**denyRemoteAddress**
This is a comma-separated list of regular expression patterns to which the remote client’s IP address is compared. If this property is specified, the remote address must not match for this request to be accepted. If this property is not specified,
request acceptance is governed solely by the `allowRemoteAddress` property. The default value for this property is null.

**allowRemoteHost**
This is a comma-separated list of regular expression patterns to which the remote client’s host name (as returned by `java.net.Socket.getInetAddress().getHostName()`) is compared. If this property is specified, the remote host name must match for this request to be accepted. If this property is not specified, all requests will be accepted unless the remote host name matches a `denyRemoteHost` pattern. The default value for this property is null.

**denyRemoteHost**
This is a comma-separated list of regular expression patterns to which the remote client’s host name (as returned by `java.net.Socket.getInetAddress().getHostName()`) is compared. If this property is specified, the remote host name must not match for this request to be accepted. If this property is not specified, request acceptance is governed solely by the `allowRemoteHost` property. The default value for this property is null.

**authRealm**
Specifies the name attribute of an `auth-realm`, which overrides the server instance’s default realm for standalone web applications deployed to this virtual server. A realm defined in a standalone web application’s `web.xml` file overrides the virtual server’s realm.

**securePagesWithPragma**
Set this property to `false` to ensure that for all web applications on this virtual server file downloads using SSL work properly in Internet Explorer.

You can set this property for a specific web application. For details, see "glassfish-web-app" in *Oracle GlassFish Server Application Deployment Guide.*

**contextXmlDefault**
Specifies the location, relative to `domain-dir`, of the `context.xml` file for this virtual server, if one is used. For more information about the `context.xml` file, see "Using a context.xml File" in *Oracle GlassFish Server Application Development Guide* and The Context Container ([http://tomcat.apache.org/tomcat-5.5-doc/config/context.html](http://tomcat.apache.org/tomcat-5.5-doc/config/context.html)). Context parameters, environment entries, and resource definitions in `context.xml` are supported in the GlassFish Server.

**alternatedocroot_<n>**
Specifies an alternate document root (docroot), where n is a positive integer that allows specification of more than one. Alternate docroots allow web applications to serve requests for certain resources from outside their own docroot, based on whether those requests match one (or more) of the URI patterns of the web application’s alternate docroots.

If a request matches an alternate docroot’s URI pattern, it is mapped to the alternate docroot by appending the request URI (minus the web application’s context root) to the alternate docroot’s physical location (directory). If a request matches multiple URI patterns, the alternate docroot is determined according to the following precedence order:

- Exact match
- Longest path match
- Extension match

For example, the following properties specify three alternate docroots. The URI pattern of the first alternate docroot uses an exact match, whereas the URI patterns of the second and third alternate docroots use extension and longest path prefix matches, respectively.

```xml
<property name="alternatedocroot_1"
  value="from=/my.jpg dir=/srv/images/jpg"/>
<property name="alternatedocroot_2"
  value="from=*.jpg dir=/srv/images/jpg"/>
<property name="alternatedocroot_3"
  value="from=/jpg/* dir=/src/images"/>
```

The value of each alternate docroot has two components: The first component, from, specifies the alternate docroot's URI pattern, and the second component, dir, specifies the alternate docroot's physical location (directory). Spaces are allowed in the dir component.

You can set this property for a specific web application. For details, see "glassfish-web-app" in Oracle GlassFish Server Application Deployment Guide.

### send-error_<n>

Specifies custom error page mappings for the virtual server, which are inherited by all web applications deployed on the virtual server. A web application can override these custom error page mappings in its web.xml deployment descriptor. The value of each send-error_<n> property has three components, which may be specified in any order:

- The first component, code, specifies the three-digit HTTP response status code for which the custom error page should be returned in the response.
- The second component, path, specifies the absolute or relative file system path of the custom error page. A relative file system path is interpreted as relative to the domain-dir/config directory.
- The third component, reason, is optional and specifies the text of the reason string (such as Unauthorized or Forbidden) to be returned.

For example:

```xml
<property name="send-error_1"
  value="code=401 path=/myhost/401.html reason=MY-401-REASON"/>
```

This example property definition causes the contents of /myhost/401.html to be returned with 401 responses, along with this response line:

```
HTTP/1.1 401 MY-401-REASON
```

### redirect_<n>

Specifies that a request for an old URL is treated as a request for a new URL. These properties are inherited by all web applications deployed on the virtual server. The value of each redirect_<n> property has two components, which may be specified in any order:

- The first component, from, specifies the prefix of the requested URI to match.
- The second component, url-prefix, specifies the new URL prefix to return to the client. The from prefix is simply replaced by this URL prefix.

For example:
<property name="redirect_1"
value="from=/dummy url-prefix=http://etude"/>

valve_n
Specifies a fully qualified class name of a custom valve, where n is a positive integer that allows specification of more than one. The valve class must implement the org.apache.catalina.Valve interface from Tomcat or previous GlassFish Server releases, or the org.glassfish.web.valve.GlassFishValve interface from the current GlassFish Server release. For example:

<property name="valve_1"
value="org.glassfish.extension.Valve"/>

You can set this property for a specific web application. For details, see "glassfish-web-app" in Oracle GlassFish Server Application Deployment Guide.

listener_n
Specifies a fully qualified class name of a custom Catalina listener, where n is a positive integer that allows specification of more than one. The listener class must implement the org.apache.catalina.ContainerListener or org.apache.catalina.LifecycleListener interface. For example:

<property name="listener_1"
value="org.glassfish.extension.MyLifecycleListener"/>

You can set this property for a specific web application. For details, see "glassfish-web-app" in Oracle GlassFish Server Application Deployment Guide.

docroot
Absolute path to root document directory for server. Deprecated. Replaced with a virtual-server attribute, docroot, that is accessible using the get, set, and list subcommands.

accesslog
Absolute path to server access logs. Deprecated. Replaced with a virtual-server attribute, access-log, that is accessible using the get, set, and list subcommands.

accessLoggingEnabled
If true, access logging is enabled for this virtual server. Deprecated. Replaced with a virtual-server attribute, access-logging-enabled, that is accessible using the get, set, and list subcommands.

sso-enabled
If true, single sign-on is enabled for web applications on this virtual server that are configured for the same realm. Deprecated. Replaced with a virtual-server attribute, sso-enabled, that is accessible using the get, set, and list subcommands.

ssoCookieSecure
Sets the Secure attribute of any JSESSIONIDSSO cookies associated with the web applications deployed to this virtual server. Deprecated. Replaced with a virtual-server attribute, sso-cookie-secure, that is accessible using the get, set, and list subcommands.
errorReportValve
Specifies a fully qualified class name of a custom valve that produces default error pages for applications on this virtual server. Specify an empty string to disable the default error page mechanism for this virtual server.

--target
Creates the virtual server only on the specified target. Valid values are as follows:

server
Creates the virtual server on the default server instance. This is the default value.

configuration-name
Creates the virtual server in the specified configuration.

cluster-name
Creates the virtual server on all server instances in the specified cluster.

standalone-instance-name
Creates the virtual server on the specified standalone server instance.

Operands

virtual-server-id
Identifies the unique ID for the virtual server to be created. This ID cannot begin with a number.

Examples

Example 1 Creating a Virtual Server
The following command creates a virtual server named sampleServer:

asadmin> create-virtual-server --hosts pigeon,localhost
--property authRealm=ldap sampleServer
Command create-virtual-server executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-virtual-server(1), list-virtual-servers(1), create-http-listener(1),
create-network-listener(1)
get(1), list(1), set(1)
asadmin(1M)
delete-admin-object

removes the administered object with the specified JNDI name.

Synopsis

delete-admin-object [-help] [--target target] jndi_name

Description

The delete-admin-object subcommand removes an administered object with the specified JNDI name.

This subcommand is supported in remote mote only.

Options

--help
-?
Displays the help text for the subcommand.

--target

This is the name of the targets for which the administered object is to be deleted. Valid values are:

**server**

Deletes the administered object for the default server instance `server` and is the default value.

**configuration_name**

Deletes the administered object for the specified configuration.

**cluster_name**

Deletes the administered object for the specified cluster.

**instance_name**

Deletes the administered object for a particular server instance.

Operands

**jndi_name**

JNDI name of the administered object to be deleted.

---

**Note:** Resources are always created for a domain as a whole but are only active for targets for which a `<resource-ref>` has been created using the `--target` option when the resource was created. This means that deleting a resource only deletes the `<resource-ref>` element for the specified `--target`, and does not delete the resource from the domain as a whole unless `domain` is specified as the `--target` for the deletion.
Examples

Example 1 Deleting an Administered Object
This example deletes the administered object named jms/samplequeue.

asadmin> delete-admin-object jms/samplequeue
Command delete-admin-object executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-admin-object(1), list-admin-objects(1)
asadmin(1M)
delete-application-ref

removes a reference to an application

Synopsis
delete-application-ref [--help] [--target target] [--cascade=false] reference_name

Description
The delete-application-ref subcommand removes a reference from a cluster or an unclustered server instance to an application. This effectively results in the application element being undeployed and no longer available on the targeted instance or cluster.

The target instance or instances making up the cluster need not be running or available for this subcommand to succeed. If one or more instances are not available, they will no longer load the application the next time they start.

Removal of the reference does not result in removal of the application from the domain. The bits are removed only by the undeploy subcommand.

This subcommand is supported in remote mode only.

Options
--help
-?
Displays the help text for the subcommand.

--target
Specifies the target from which you are removing the application reference. Valid values are

- server- Specifies the default server instance as the target. server is the name of the default server instance and is the default value.
- cluster_name- Specifies a certain cluster as the target.
- instance_name- Specifies a certain stand-alone server instance as the target.

--cascade
For a connector module, indicates whether the resources dependent on the module should also be recursively deleted. The default is false. The connector module can be either a stand-alone RAR file or a module within an EAR file.

Operands
reference_name
The name of the application or module, which can be a Java EE application module, Web module, EJB module, connector module, application client module, or lifecycle module.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter
or number. It can contain alphanumeric characters plus underscore (\_), dash (-), and period (.) characters. To delete references to multiple versions, you can use an asterisk (*\) as a wildcard character. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.

Examples

Example 1 Deleting an Application Reference
The following example removes a reference to the Web module MyWebApp from the unclustered server instance NewServer.

asadmin> delete-application-ref --target NewServer MyWebApp
Command delete-application-ref executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-application-ref(1), list-application-refs(1), undeploy(1)
asadmin(1M)

Oracle GlassFish Server Application Deployment Guide
delete-audit-module

removes the named audit-module

Synopsis

delete-audit-module [--help]
[--target target]
audit_module_name

Description

This subcommand removes the named audit module. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target on which you are deleting the audit module. Valid values are as follows:

server
Deletes the audit module for the default server instance server and is the default value.

collection_name
Deletes the audit module for the named configuration.

collection_name
Deletes the audit module for every server instance in the cluster.

instance_name
Deletes the audit module for a particular server instance.

Operands

audit_module_name
The name of the audit module to be deleted.

Examples

Example 1  Deleting an audit module

asadmin> delete-audit-module sampleAuditModule
Command delete-audit-module executed successfully
Exit Status

0  
command executed successfully

1  
error in executing the command

See Also

create-audit-module(1), list-audit-modules(1)
asadmin(1M)
**delete-auth-realm**

removes the named authentication realm

**Synopsis**

delete-auth-realm [--help]
[--target target]
auth_realm-name

**Description**
The *delete-auth-realm* subcommand removes the named authentication realm. This subcommand is supported in remote mode only.

**Options**

`--help`
-?
Displays the help text for the subcommand.

`--target`
Specifies the target on which you are deleting the authentication realm. Valid values are

- `server`
  Deletes the realm for the default server instance `server` and is the default value.

- `configuration_name`
  Deletes the realm for the named configuration.

- `cluster_name`
  Deletes the realm for every server instance in the cluster.

- `instance_name`
  Deletes the realm for a particular server instance.

**Operands**

*auth_realm_name*
Name of the realm to be deleted.

**Examples**

**Example 1 Deleting an Authentication Realm**
This example deletes the authentication realm `db`.

```
asadmin> delete-auth-realm db
Command delete-auth-realm executed successfully
```
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-auth-realm(1), list-auth-realms(1)
asadmin(1M)
delete-backup-config

deletes an existing domain backup configuration

Synopsis

delete-backup-config [--help] backup-config-name

Description

The delete-backup-config subcommand deletes an existing domain backup configuration.
This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

backup-config-name
Specifies the name of the domain backup configuration to delete.

Examples

Example 1 Deleting a Domain Backup Configuration
This example deletes the monthly-config domain backup configuration.

asadmin> delete-backup-config monthly-backup
Command delete-backup-config executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-backup-config(1), disable-backup-config(1), enable-backup-config(1),
list-backup-configs(1)
asadmin(1M)
delete-cluster

deletes a GlassFish Server cluster

Synopsis

delete-cluster [--help]
[--autohadboverride={true|false}] [--node-agent=node-agent--name]
cluster-name

Description

The delete-cluster subcommand deletes a GlassFish Server cluster. A cluster can be
deleted only if the cluster contains no GlassFish Server instances. If a cluster that you
are deleting contains any instances, stop and delete the instances before deleting the
cluster.

If the cluster's named configuration was created automatically for the cluster and no
other clusters or unclustered instances refer to the configuration, the configuration is
deleted when the cluster is deleted. A configuration that is created automatically for a
cluster is named cluster-name-config, where cluster-name is the name of the cluster.

This command is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--autohadboverride
Do not specify this option. This option is retained for compatibility with earlier
releases. If you specify this option, a syntax error does not occur. Instead, the
subcommand runs successfully and displays a warning message that the option is
ignored.

--nodeagent
Do not specify this option. This option is retained for compatibility with earlier
releases. If you specify this option, a syntax error does not occur. Instead, the
subcommand runs successfully and displays a warning message that the option is
ignored.

Operands

cluster-name
The name of the cluster to delete.

Examples

Example 1 Deleting a GlassFish Server Cluster

This example deletes the GlassFish Server cluster adccluster.

asadmin> delete-cluster adccluster
Command delete-cluster executed successfully.

**Exit Status**

0  
command executed successfully  

1  
error in executing the command

**See Also**

create-cluster(1), delete-instance(1), delete-local-instance(1),
list-clusters(1), start-cluster(1), stop-instance(1), stop-local-instance(1),
stop-cluster(1)

asadmin(1M)
delete-config

deletes an existing named configuration

Synopsis

delete-config [--help] configuration-name

Description

The `delete-config` subcommand deletes an existing named configuration from the configuration of the domain administration server (DAS). You can delete a configuration only if no GlassFish Server instances or clusters refer to the configuration. A standalone configuration is automatically deleted when the instance or cluster that refers to it is deleted. You cannot delete the `default-config` configuration that is copied to create standalone configurations.

This subcommand is supported in remote mode only.

Options

- `--help`
- `-?`
  Displays the help text for the subcommand.

Operands

- `configuration-name`
  The name of the configuration that you are deleting.

Examples

**Example 1 Deleting a Named Configuration**

This example deletes the named configuration `pmdconfig`.

```
asadmin> delete-config pmdconfig
Command delete-config executed successfully.
```

Exit Status

- `0`
  command executed successfully
- `1`
  error in executing the command

See Also

copy-config(1), list-configs(1)
asadmin(1M)
configuration(5ASC)
delete-connector-connection-pool

removes the specified connector connection pool

Synopsis

delete-connector-connection-pool [--help] [--target target] [--cascade={false|true}] poolname

Description

The delete-connector-connection-pool subcommand removes the specified
connector connection pool.

This subcommand is supported in remote mode only.

Options

--help

Displays the help text for the subcommand.

--target

Do not specify this option. This option is retained for compatibility with earlier
releases. If you specify this option, a syntax error does not occur. Instead, the
subcommand runs successfully and displays a warning message that the option is
ignored.

--cascade

When set to true, all connector resources associated with the pool, and the pool itself,
are deleted. When set to false, the deletion of pool fails if any resources are associated
with the pool. The resource must be deleted explicitly or the option must be set to true.
Default is false.

Operands

poolname

The name of the connection pool to be removed.

Examples

Example 1 Deleting a Connector Connection Pool

This example deletes the connector connection pool named jms/qConnPool.

asadmin> delete-connector-connection-pool
--cascade=false jms/qConnPool

Command delete-connector-connection-pool executed successfully
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-connector-connection-pool(1), list-connector-connection-pools(1),
ping-connection-pool(1)
asadmin(1M)
delete-connector-resource

removes the connector resource with the specified JNDI name

Synopsis

delete-connector-resource [--help] [--target target] jndi_name

Description

The delete-connector-resource subcommand removes the connector resource with
the specified JNDI name.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target from which you want to remove the connector
resource. Valid targets are:

server
Deletes the connector resource from the default server instance. This is the default
value.

domain
Deletes the connector resource from the domain.

cluster_name
Deletes the connector resource from every server instance in the cluster.

instance_name
Deletes the connector resource from a specified server instance.

Operands

jndi_name
The JNDI name of this connector resource.

Note: Resources are always created for a domain as a whole but
are only active for targets for which a <resource-ref> has been
created using the --target option when the resource was created.
This means that deleting a resource only deletes the
<resource-ref> element for the specified --target, and does not
delete the resource from the domain as a whole unless domain is
specified as the --target for the deletion.
Examples

Example 1 Deleting a Connector Resource
This example deletes a connector resource named jms/qConnFactory.

```
asadmin> delete-connector-resource jms/qConnFactory
Command delete-connector-resource executed successfully
```

Example 2 Using the delete-connector-resource subcommand
This example shows the usage of this subcommand.

```
asadmin> delete-connector-resource jms/qConnFactory
Command delete-connector-resource executed successfully

Where jms/qConnFactory is the connector resource that is removed.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-connector-resource(1), list-connector-resources(1)

asadmin(1M)
**delete-connector-security-map**

deletes a security map for the specified connector connection pool

**Synopsis**

delete-connector-security-map [--help]
--poolname connector_connection_pool_name [--target target] mapname

**Description**
The `delete-connector-security-map` subcommand deletes a security map for the specified connector connection pool.

For this subcommand to succeed, you must have first created a connector connection pool using the `create-connector-connection-pool` subcommand.

This subcommand is supported in remote mode only.

**Options**

--help
-?
Displays the help text for the subcommand.

--poolname
Specifies the name of the connector connection pool to which the security map that is to be deleted belongs.

--target
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

**Operands**

mapname
Name of the security map to be deleted.

**Examples**

**Example 1 Deleting a Connector Security Map**

This example deletes `securityMap1` for the existing connection pool named `connector-pool1`.

```bash
asadmin> delete-connector-security-map
--poolname connector-pool1 securityMap1
Command delete-connector-security-map executed successfully
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-connector-security-map(1), list-connector-security-maps(1),
update-connector-security-map(1)
asadmin(1M)
delete-connector-work-security-map

   deletes a work security map for the specified resource adapter

Synopsis

delete-connector-work-security-map [--help] --raname raname mapname

Description

The delete-connector-work-security-map subcommand deletes a security map associated with the specified resource adapter. For this subcommand to succeed, you must have first created and deployed the specified resource adapter.

The enterprise information system (EIS) is any system that holds the data of an organization. It can be a mainframe, a messaging system, a database system, or an application.

This subcommand is supported in remote mode only.

Options

   --help
   -?
   Displays the help text for the subcommand.

   --raname
   Indicates the connector module name with which the work security map is associated.

Operands

   mapname
   The name of the work security map to be deleted.

Examples

Example 1 Deleting a Connector Work Security Map

This example deletes the work security map named work_security_map_name for the resource adapter named ra_name.

   asadmin delete-connector-work-security-map
   --raname ra_name work_security_map_name

Command delete-connector-work-security-map executed successfully.

Exit Status

   0
   subcommand executed successfully

   1
   error in executing the subcommand
See Also

create-connector-work-security-map(1), list-connector-work-security-maps(1),
update-connector-work-security-map(1)

asadmin(1M)
delete-custom-resource

removes a custom resource

Synopsis

delete-custom-resource [--help] [--target target] jndi-name

Description

The delete-custom-resource subcommand removes a custom resource. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option helps specify the location of the custom resources that you are deleting. Valid targets are server, domain, cluster, and instance. The default is server.

Note: Resources are always created for a domain as a whole but are only active for targets for which a <resource-ref> has been created using the --target option when the resource was created. This means that deleting a resource only deletes the <resource-ref> element for the specified --target, and does not delete the resource from the domain as a whole unless domain is specified as the --target for the deletion.

server
Deletes the resource for the default server instance. This is the default value.

domain
Deletes the resource for the domain.

cluster_name
Deletes the resource for every server instance in the cluster.

instance_name
Deletes the resource for a particular server instance.

Operands

jndi-name
The JNDI name of this resource.
Examples

Example 1 Deleting a Custom Resource
This example deletes a custom resource named mycustomresource.

asadmin> delete-custom-resource mycustomresource
Command delete-custom-resource executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-custom-resource(1), list-custom-resources(1)
asadmin(1M)
delete-domain

deletes a domain

Synopsis
delete-domain [--help] [--domaindir domaindir] domain-name

Description
The delete-domain subcommand deletes the specified domain. The domain must already exist and must be stopped.
This subcommand is supported in local mode only.

Options
--help
-?
Displays the help text for the subcommand.

--domaindir
The directory where the domain to be deleted is located. If specified, the path must be accessible in the file system. If not specified, the domain under the domain root directory, which defaults to as-install/domains, is deleted.

Operands
domain-name
The unique name of the domain you want to delete.

Examples

Example 1 Deleting a Domain
This example deletes a domain named mydomain4 from the default domains directory.
asadmin> delete-domain mydomain4
Domain mydomain4 deleted.
Command delete-domain executed successfully.

Example 2 deleting a Domain From an Alternate Location
This example deletes a domain named sampleDomain from the /home/someuser/domains directory.
asadmin> delete-domain --domaindir /home/someuser/domains sampleDomain
Domain sampleDomain deleted
Command delete-domain executed successfully.
Exit Status

0
subcommand executed successfully

1
ter error in executing the subcommand

See Also

create-domain(1), start-domain(1), stop-domain(1), list-domains(1)
asadmin(1M)
delete-file-user

removes the named file user

Synopsis

delete-file-user [--help][--authrealmname auth_realm_name][--target target] username

Description

The delete-file-user subcommand deletes the entry in the keyfile for the specified username.

Options

--help
-?
Displays the help text for the subcommand.

--authrealmname
The name of the authentication realm with which the user was created.

--target
This is the name of the target on which the command operates. The valid targets are:

  server
  Deletes the file user on the default server instance. This is the default value

  domain
  Deletes the file user in the domain.

  cluster_name
  Deletes the file user from every server instance in the cluster.

  instance_name
  Deletes the file user from a particular server instance.

Operands

username
This is the name of file user to be deleted.

Examples

Example 1 Deleting a User From a File Realm
The following example shows how to delete user named sample_user from a file realm.

asadmin> delete-file-user
sample_user
Command delete-file-user executed successfully
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-file-user(1), list-file-users(1), update-file-user(1),
list-file-groups(1)

asadmin(1M)
delete-http

removes HTTP parameters from a protocol

Synopsis

delte-http [--help]
[--target target]
protocol-name

Description

The delete-http subcommand removes the specified HTTP parameter set from a protocol. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Deletes the HTTP parameter set only from the specified target. Valid values are as follows:

server
Deletes the HTTP parameter set from the default server instance. This is the default value.

configuration-name
Deletes the HTTP parameter set from the specified configuration.

cluster-name
Deletes the HTTP parameter set from all server instances in the specified cluster.

standalone-instance-name
Deletes the HTTP parameter set from the specified standalone server instance.

Operands

protocol-name
The name of the protocol from which to delete the HTTP parameter set.

Examples

Example 1 Deleting an HTTP Parameter Set
The following command deletes the HTTP parameter set from a protocol named http-1:

asadmin> delete-http http-1
Command delete-http executed successfully.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-http(1)
asadmin(1M)
delete-http-health-checker

deletes the health-checker for a specified load balancer configuration

Synopsis

delete-http-health-checker [--help] [--config config_name]

target

Description

The `delete-http-health-checker` subcommand deletes the health checker from a
load balancer configuration. A health checker is unique for the combination of target
and load balancer configuration.

```
Note: This subcommand is only applicable to Oracle GlassFish
Server. This subcommand is not applicable to GlassFish Server
```

Options

--help
-?
Displays the help text for the subcommand.

--config
The load balancer configuration from which you delete the health-checker.

Operands

target
Specifies the target from which you are deleting the health checker.
Valid values are:
- `cluster_name` - The name of a target cluster.
- `instance_name` - The name of a target server instance.

Examples

Example 1 Deleting a Health Checker from a Load Balancer Configuration
This example deletes the health checker for load balancer configuration named
mycluster-http-lb-config on a cluster named mycluster.

```
asadmin> delete-http-health-checker --user admin
--passwordfile password.txt --config mycluster-http-lb-config mycluster
```

Command delete-http-health-checker executed successfully.
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-http-health-checker(1)
asadmin(1M)
delete-http-lb

**Synopsis**

delete-http-lb [--help] load_balancer_name

**Description**

Use the `delete-http-lb` subcommand to delete a physical load balancer.

---

**Note:** This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

---

**Options**

--help

-?

Displays the help text for the subcommand.

**Operands**

*load_balancer_name*

The name of the load balancer to be deleted.

**Examples**

**Example 1 Deleting a Load Balancer Configuration**

This example deletes the load balancer configuration named *mylb*.

```plaintext
asadmin> delete-http-lb mylb

Command delete-http-lb executed successfully.
```

**Exit Status**

0

subcommand executed successfully

1

error in executing the subcommand

**See Also**

*create-http-lb(1), list-http-lbs(1)*

*asadmin(1M)*
delete-http-lb-config

deletes a load balancer configuration

Synopsis

delete-http-lb-config [--help] config_name

Description

Use the delete-http-lb-config subcommand to delete a load balancer configuration. The load balancer configuration must not reference any clusters or server instances enabled for load balancing. In addition, the load balancer configuration must not be referenced by any physical load balancers.

Note: This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

Options

--help
-?
Displays the help text for the subcommand.

Operands

config_name
The name of the load balancer configuration to delete. The configuration must not reference any clusters or server instances enabled for load balancing, or be used by any physical load balancers.

Examples

Example 1 Deleting a Load Balancer Configuration
This example deletes a load balancer configuration named mylbconfig

asadmin> delete-http-lb-config mylbconfig
Command delete-http-lb-config executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand
See Also

create-http-lb-config(1), list-http-lb-configs(1)
asadmin(1M)
delete-http-lb-ref

deletes the cluster or server instance from a load balancer

Synopsis

delete-http-lb-ref [--help] --config config_name | --lbname load_balancer_name
[--force=false] target

Description

Use the delete-http-lb-ref subcommand to remove a reference to a cluster or
standalone server instance from a load balancer configuration or load balancer. So that
you do not interrupt user requests, make sure the standalone server instance or all
server instances in the cluster are disabled before you remove them from the load
balancer configuration. If the force option is set to true, the references are deleted
even if server instances or clusters are enabled.

Note: This subcommand is only applicable to Oracle GlassFish
Server. This subcommand is not applicable to GlassFish Server

Options

--help
-?
Displays the help text for the subcommand.

--config
Specifies which load balancer configuration to delete cluster and server instance
references from.
Specify either a load balancer configuration or a load balancer. Specifying both results
in an error.

--lbname
Specifies the load balancer to delete cluster and server instance references from.
Specify either a load balancer configuration or a load balancer. Specifying both results
in an error.

--force
If force is set to true, then the references are deleted even if there are currently
enabled applications or instances. The default is false.

Operands

target
Specifies which cluster or instance to remove from the load balancer. Valid values are:

- cluster_name- The name of a target cluster.
- instance_name- The name of a target server instance.
Examples

Example 1 Deleting a Cluster Reference from a Load Balancer Configuration
This example deletes the reference to cluster named cluster2 from a load balancer configuration named mycluster-http-lb-config.

asadmin> delete-http-lb-ref --config mycluster-http-lb-config cluster2

Command delete-http-lb-ref executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-http-lb-ref(1), disable-http-lb-server(1)
asadmin(1M)
delete-http-listener

removes a network listener

Synopsis

delete-http-listener [--help]
[-target target]
listener-id

Description

The delete-http-listener subcommand removes the specified network listener. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Deletes the network listener only from the specified target. Valid values are as follows:

server
Deletes the network listener from the default server instance. This is the default value.

configuration-name
Deletes the network listener from the specified configuration.

cluster-name
Deletes the network listener from all server instances in the specified cluster.

standalone-instance-name
Deletes the network listener from the specified standalone server instance.

Operands

listener-id
The unique identifier for the network listener to be deleted.

Examples

Example 1 Using the delete-http-listener subcommand
The following command deletes the network listener named sampleListener:

asadmin> delete-http-listener sampleListener
Command delete-http-listener executed successfully.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-http-listener(1), list-http-listeners(1)
asadmin(1M)
delete-http-redirect

removes an HTTP redirect

Synopsis

delete-http-redirect [--help]
[---target target]
protocol-name

Description

The delete-http-redirect subcommand removes the specified HTTP redirect. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Deletes the HTTP redirect only from the specified target. Valid values are as follows:

server
Deletes the HTTP redirect from the default server instance. This is the default value.

configuration-name
Deletes the HTTP redirect from the specified configuration.

cluster-name
Deletes the HTTP redirect from all server instances in the specified cluster.

standalone-instance-name
Deletes the HTTP redirect from the specified standalone server instance.

Operands

protocol-name
The name of the associated protocol.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-http-redirect(1)
asadmin(1M)
delete-iiop-listener

removes an IIOP listener

Synopsis

delete-iiop-listener [--help] [--target target] listener_id

Description

The delete-iiop-listener subcommand removes the specified IIOP listener. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target from which you are deleting the IIOP listener. Valid values are

server
Deletes the listener from the default server instance server and is the default value.

configuration_name
Deletes the listener from the named configuration.

cluster_name
Deletes the listener from every server instance in the cluster.

instance_name
Deletes the listener from a particular server instance.

Operands

listener_id
The unique identifier for the IIOP listener to be deleted.

Examples

Example 1 Deleting an IIOP Listener
The following command deletes the IIOP listener named sample_iiop_listener:

asadmin> delete-iiop-listener sample_iiop_listener
Command delete-iiop-listener executed successfully.
Exit Status

0  command executed successfully
1  error in executing the command

See Also

create-iiop-listener(1), list-iiop-listeners(1)
asadmin(1M)
delete-instance

deletes a GlassFish Server instance

Synopsis

delete-instance [--help] instance-name

Description

The delete-instance subcommand deletes a GlassFish Server instance. This subcommand requires the Distributed Component Object Model (DCOM) remote protocol or secure shell (SSH) to be configured on the host where the domain administration server (DAS) is running and on the host that is represented by the node where the instance resides.

Note: DCOM or SSH is not required if the instance resides on a node of type CONFIG that represents the local host. A node of type CONFIG is not enabled for remote communication over DCOM or SSH.

You may run this subcommand from any host that can contact the DAS.

The subcommand can delete any GlassFish Server instance, regardless of how the instance was created. For example, this subcommand can delete an instance that was created by using the create-local-instance(1) subcommand.

The instance that is being deleted must not be running. Otherwise, an error occurs.

The subcommand deletes an instance by performing the following actions:

- Removing the instance from the configuration of the domain administration server (DAS)
- Deleting the instance's files from file system

If the instance that is being deleted is the only instance that is using the node directory, that directory is also removed.

If a standalone instance is deleted, the instance's standalone configuration is also deleted. A standalone instance refers to a configuration that is named instance-name-config to which no other clusters or unclustered instances refer.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.
Operands

instance-name
The name of the instance to delete.

Examples

Example 1 Deleting a GlassFish Server Instance
This example deletes the GlassFish Server instance pmdsainst.

asadmin> delete-instance pmdsainst
Command delete-instance executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-instance(1), create-local-instance(1), delete-local-instance(1),
setup-ssh(1), start-instance(1), start-local-instance(1), stop-instance(1),
stop-local-instance(1), validate-dcom(1)

asadmin(1M)
delete-jacc-provider

enables administrators to delete JACC providers defined for a domain

Synopsis

delete-jacc-provider [--help]
[--target target] jacc-provider-name

Description

The delete-jacc-provider subcommand enables administrators to delete JACC providers defined for a domain. JACC providers are defined as jacc-provider elements in the security-service element in the domain's domain.xml file. JACC providers can be created using the GlassFish Server Admin Console or the create-jacc-provider subcommand.

The default GlassFish Server installation includes two JACC providers, named default and simple. These default providers should not be deleted.

The JACC provider used byGlassFish Server for authorization is identified by the jacc-provider element of security-service in domain.xml. Therefore, if you delete the jacc-provider provider, make sure you change jacc-provider to the name of some other JACC provider that exists under security-service.

If you change the jacc-provider element to point to a different JACC provider, you must restart GlassFish Server.

This subcommand is supported in remote mode only.

Options

If an option has a short option name, then the short option precedes the long option name. Short options have one dash whereas long options have two dashes.

--help
-?
Displays the help text for the subcommand.

--target

Specifies the target from which you are deleting the JACC provider. The following values are valid:

server
Deletes the JACC provider on the default server instance. This is the default value.

capiplication_name
Deletes the JACC provider in the specified configuration.

cluster_name
Deletes the JACC provider on all server instances in the specified cluster.

instance_name
Deletes the JACC provider on a specified server instance.
Operands

`jacc-provider-name`

The name of the JACC provider to be deleted.

Examples

Example 1 Deleting a JACC provider

The following example shows how to delete a JACC provider named `testJACC` from the default domain.

```
asadmin> delete-jacc-provider testJACC
```

Command delete-jacc-provider executed successfully.

Exit Status

0

subcommand executed successfully

1

error in executing the subcommand

See Also

`create-jacc-provider(1), list-jacc-providers(1)`

`asadmin(1M)`
**delete-javamail-resource**

removes a JavaMail session resource

**Synopsis**

`delete-javamail-resource [--help] [--target target] jndi_name`

**Description**

The `delete-javamail-resource` subcommand removes the specified JavaMail session resource. Ensure that you remove all references to this resource before running this subcommand.

This subcommand is supported in remote mode only.

**Options**

`--help`

Displays the help text for the subcommand.

`--target`

This option specifies the target from which you are deleting the JavaMail session resource. Valid values are:

- **server**
  Deletes the resource from the default server instance. This is the default value.

- **domain**
  Deletes the resource from the domain.

- **cluster_name**
  Deletes the resource from every server instance in the cluster.

- **instance_name**
  Deletes the resource from a particular server instance.

**Operands**

`jndi_name`

The JNDI name of the JavaMail session resource to be deleted.

**Examples**

**Example 1 Deleting a JavaMail Resource**

This example deletes the JavaMail session resource named `mail/MyMailSession`.

```bash
asadmin> delete-javamail-resource mail/MyMailSession
Command delete-javamail-resource executed successfully.
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-javamail-resource(1), list-javamail-resources(1)
asadmin(1M)
delete-jdbc-connection-pool

removes the specified JDBC connection pool

Synopsis

deleate-jdbc-connection-pool [--help]
[-cascase={false|true}]
[--target target]
jdbc_connection_pool_id

Description

The delete-jdbc-connection-pool subcommand deletes a JDBC connection pool. Before running this subcommand, all associations to the JDBC connection pool must be removed.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--cascade
If the option is set to true, all the JDBC resources associated with the pool, apart from the pool itself, are deleted. When set to false, the deletion of pool fails if any resources are associated with the pool. Resources must be deleted explicitly or the option must be set to true. The default value is false.

--target
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

Operands

jdbc_connection_pool_id
The name of the JDBC resource to be removed.

Examples

Example 1 Deleting a JDBC Connection Pool
This example deletes the sample_derby_pool JDBC connection pool.
asadmin> delete-jdbc-connection-pool --cascade=false sample_derby_pool
Command delete-jdbc-connection-pool executed correctly.
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jdbc-connection-pool(1), list-jdbc-connection-pools(1)
asadmin(1M)
delete-jdbc-resource

removes a JDBC resource with the specified JNDI name

Synopsis

delete-jdbc-resource [--help] [--target target] jndi_name

Description

The delete-jdbc-resource subcommand removes a JDBC resource. Ensure that all associations to the JDBC resource are removed before running this subcommand. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option helps specify the target from which you are removing the JDBC resource. Valid targets are:

server
Removes the resource from the default server instance. This is the default value.

domain
Removes the resource from the domain.

cluster_name
Removes the resource from every server instance in the cluster.

instance_name
Removes the resource from a particular sever instance.

Note: Resources are always created for a domain as a whole but are only active for targets for which a <resource-ref> has been created using the --target option when the resource was created. This means that deleting a resource only deletes the <resource-ref> element for the specified --target, and does not delete the resource from the domain as a whole unless domain is specified as the --target for the deletion.

Operands

jndi_name
The JNDI name of this JDBC resource to be removed.
Examples

Example 1 Deleting a JDBC Resource
The following example deletes the JDBC resource named jdbc/DerbyPool.

    asadmin> delete-jdbc-resource jdbc/DerbyPool
    Command delete-jdbc-resource executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jdbc-resource(1), list-jdbc-resources(1)
asadmin(1M)
delete-jmsdest

removes a JMS physical destination

Synopsis

delete-jmsdest [--help]
--desttype type
[--target target]
dest_name

Description

The delete-jmsdest subcommand removes the specified Java Message Service (JMS) physical destination.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

--help
-?
Displays the help text for the subcommand.

--desttype
The type of the JMS destination. Valid values are topic and queue.

--target
Deletes the physical destination only from the specified target. Although the delete-jmsdest subcommand is related to resources, a physical destination is deleted using the JMS Service (JMS Broker), which is part of the configuration. A JMS Broker is configured in the config section of domain.xml. Valid values are as follows:

  server
  Deletes the physical destination from the default server instance. This is the default value.

  configuration-name
  Deletes the physical destination from the specified configuration.

  cluster-name
  Deletes the physical destination from every server instance in the specified cluster.

  instance-name
  Creates the physical destination from the specified server instance.

Operands

dest_name
The unique identifier of the JMS destination to be deleted.
Examples

**Example 1 Deleting a physical destination**
The following subcommand deletes the queue named PhysicalQueue.

```plaintext
asadmin> delete-jmsdest --desttype queue PhysicalQueue
Command delete-jmsdest executed successfully.
```

Exit Status

- **0**
  subcommand executed successfully

- **1**
  error in executing the subcommand

See Also

`create-jmsdest(1), list-jmsdest(1), flush-jmsdest(1)`

`asadmin(1M)`
delete-jms-host

removes a JMS host

Synopsis

delete-jms-host [--help]
|--target target
 jms_host_name

Description

The delete-jms-host subcommand removes the specified Java Message Service (JMS) host.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Deleting the default JMS host, named default_JMS_host, is not recommended.

Options

--help
-?
Displays the help text for the subcommand.

--target
Deletes the JMS host only from the specified target. Valid values are as follows:

server
Deletes the JMS host from the default server instance. This is the default value.

configuration-name
Deletes the JMS host from the specified configuration.

cluster-name
Deletes the JMS host from every server instance in the specified cluster.

instance-name
Deletes the JMS host from the specified server instance.

Operands

jms_host_name
The name of the host to be deleted.

Examples

Example 1 Deleting a JMS host
The following subcommand deletes the JMS host named MyNewHost.

asadmin> delete-jms-host MyNewHost
Command delete-jms-host executed successfully.
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jms-host(1), list-jms-hosts(1), jms-ping(1)
asadmin(1M)
delete-jms-resource

removes a JMS resource

Synopsis

delete-jms-resource [--help]
   [--target target]
   jndi_name

Description

The delete-jms-resource subcommand removes the specified Java Message Service (JMS) resource. Ensure that you remove all references to this resource before executing this subcommand.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

   --help
   -?
   Displays the help text for the subcommand.

   --target
   Deletes the JMS resource only from the specified target. Valid values are as follows:

   server
   Deletes the JMS resource from the default server instance. This is the default value.

   domain
   Deletes the JMS resource from the domain.

   cluster-name
   Deletes the JMS resource from every server instance in the specified cluster.

   instance-name
   Deletes the JMS resource from the specified server instance.

Operands

   jndi_name
   The JNDI name of the JMS resource to be deleted.
Examples

Example 1 Deleting a JMS destination resource
The following subcommand deletes the JMS destination resource named jms/MyQueue.

asadmin> delete-jms-resource jms/MyQueue
Administered object jms/MyQueue deleted.
Command delete-jms-resource executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jms-resource(1), list-jms-resources(1)
asadmin(1M)
delete-jndi-resource

removes a JNDI resource

Synopsis

delete-jndi-resource [--help] [--target target] jndi_name

Description

The delete-jndi-resource subcommand removes the specified JNDI resource. You must remove all associations to the JNDI resource before running this subcommand. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Valid targets are described below.

Note: Resources are always created for a domain as a whole but are only active for targets for which a <resource-ref> has been created using the --target option when the resource was created. This means that deleting a resource only deletes the <resource-ref> element for the specified --target, and does not delete the resource from the domain as a whole unless domain is specified as the --target for the deletion.

server
Deletes the resource from the default server instance. This is the default value.

domain
Deletes the resource from the domain.

cluster_name
Deletes the resource for every server instance in the cluster.

instance_name
Deletes the resource from the specified server instance.

Operands

jndi_name
The name of the JNDI resource to be removed.
Examples

Example 1 Deleting a JNDI Resource
This example removes an existing JNDI resource named sample_jndi_resource.

asadmin> delete-jndi-resource sample_jndi_resource
Command delete-jndi-resource executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jndi-resource(1), list-jndi-resources(1)
asadmin(1M)
delete-jvm-options

removes one or more options for the Java application launcher

Synopsis

delete-jvm-options [--help] [--target target] [--profiler={true|false}]
(jvm-option-name[=jvm-option-value]) [:jvm-option-name[=jvm-option-name]]*

Description

The delete-jvm-options subcommand removes one or more command-line options
for the Java application launcher. These options are removed from the Java
configuration java-config element or the profiler profiler element of the domain.xml
file. To see the Java application launcher options that can be deleted, use the
list-jvm-options(1) subcommand.

The deletion of some options requires a server restart for changes to become effective.
Other options are set immediately in the environment of the domain administration
server (DAS) and do not require a restart.

Whether a restart is required depends on the type of option.

■ Restart is not required for Java system properties whose names do not start with
  -Djava. or -Djavax. (including the trailing period). For example, restart is not
  required for the following Java system property:
  -Denvirontment=Production

■ Restart is required for the following options:
  - Java system properties whose names start with -Djava. or -Djavax.
    (including the trailing period). For example:
    -Djava.security.manager
  - Startup parameters for the Java application launcher. For example:
    -client
    -Xmx1024m
    -d64

To restart the DAS, use the restart-domain(1) command.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target from which you are removing Java application launcher options.

Valid values are as follows:
server
Specifies the DAS (default).

instance-name
Specifies a GlassFish Server instance.

cluster-name
Specifies a cluster.

configuration-name
Specifies a named configuration.

--profiler
Indicates whether the Java application launcher options are for the profiler. The option must have been set for a profiler for this option to be true.

Operands

jvm-option-name
One or more options delimited by a colon (:). The format of the operand depends on the following:

- If the option has a name and a value, the format is option-name=value.
- If the option has only a name, the format is option-name. For example, -Xmx2048m.

Note: If an option name or option value contains a colon, the backslash (\) must be used to escape the colon in the name or value. Other characters might also require an escape character. For more information about escape characters in subcommand options, see the asadmin(1M) man page.

Examples

Example 1 Deleting Java Application Launcher Options
This example removes multiple Java application launcher options.

asadmin> delete-jvm-options -Doption1=value1
"-Doption1=value1:-Doption2=value2"
Command delete-jvm-options executed successfully

Example 2 Deleting a Java Application Launcher Option From the Profiler
This example removes a Java application launcher startup parameter for the profiler.

asadmin> delete-jvm-options --profiler=true -XX:MaxPermSize=192m
Command delete-jvm-options executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand
See Also

create-jvm-options(1), list-jvm-options(1), restart-domain(1)

asadmin(1M)

For more information about the Java application launcher, see the reference page for the operating system that you are using:

- Oracle Solaris and Linux: java - the Java application launcher
  (http://java.sun.com/javase/6/docs/technotes/tools/solaris/java.html)

- Windows: java - the Java application launcher
  (http://java.sun.com/javase/6/docs/technotes/tools/windows/java.html)
delete-lifecycle-module

removes the lifecycle module

Synopsis

delete-lifecycle-module [--help] [--target target] module_name

Description

The delete-lifecycle-module subcommand removes a lifecycle module. A lifecycle module provides a means of running a short or long duration Java-based task at a specific stage in the server life cycle. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Indicates the location where the lifecycle module is to be deleted. Valid values are

- server - Specifies the default server instance as the target for deleting the lifecycle module. server is the name of the default server instance and is the default value for this option.
- cluster_name - Specifies a particular cluster as the target for deleting the lifecycle module.
- instance_name - Specifies a particular server instance as the target for deleting the lifecycle module.

Operands

module_name
This operand is a unique identifier for the deployed server lifecycle event listener module.

Examples

Example 1 Deleting a Lifecycle Module
The following example deletes a lifecycle module named customSetup.

asadmin> delete-lifecycle-module customSetup
Command delete-lifecycle-module executed successfully

Exit Status

0
command executed successfully
1
error in executing the command

See Also

create-lifecycle-module(1), list-lifecycle-modules(1)
asadmin(1M)
delete-local-instance

deletes a GlassFish Server instance on the machine where the subcommand is run

Synopsis

delete-local-instance [--help]
[--nodir node-dir] [--node node-name] [instance-name]

Description

The delete-local-instance subcommand deletes a GlassFish Server instance on the
machine where the subcommand is run. This subcommand does not require the
Distributed Component Object Model (DCOM) remote protocol or secure shell (SSH)
to be configured. You must run this command from the machine where the instance
resides.

The subcommand can delete any GlassFish Server instance, regardless of how the
instance was created. For example, this subcommand can delete an instance that was
created by using the create-instance(1) subcommand.

The instance that is being deleted must not be running. Otherwise, an error occurs.

The subcommand deletes an instance by performing the following actions:

- Removing the instance from the configuration of the domain administration server
  (DAS)
- Deleting the instance's files from file system

If the instance that is being deleted is the only instance that is using the node directory,
that directory is also removed.

If a standalone instance is deleted, the instance's standalone configuration is also
deleted. A standalone instance refers to a configuration that is named
instance-name-config to which no other clusters or unclustered instances refer.

The delete-local-instance subcommand does not contact the DAS to determine the
node on which the instance resides. To determine the node on which the instance
resides, the subcommand searches the directory that contains the node directories. If
multiple node directories exist, the node must be specified as an option of the
subcommand.

If no operand is specified and only one instance resides on the specified node, the
subcommand deletes the instance. If no operand is specified and multiple instances
reside on the node, an error occurs.

This subcommand is supported in remote mode only.

Options

- --help
  -?  Displays the help text for the subcommand.
--nodedir
Specifies the directory that contains the instance's node directory. The instance's files are stored in the instance's node directory. The default is as-install/nodes.

--node
Specifies the node on which the instance resides. This option may be omitted only if the directory that the --nodedir option specifies contains only one node directory. Otherwise, this option is required.

Operands

instance-name
The name of the instance to delete. This operand may be omitted if only one instance resides on the specified node. Otherwise, this operand is required.

Examples

Example 1 Deleting an Instance
This example deletes the instance pmdsainst.

asadmin> delete-local-instance pmdsainst

Command delete-local-instance executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-instance(1), create-local-instance(1), delete-instance(1),
start-instance(1), start-local-instance(1), stop-instance(1),
stop-local-instance(1)
asadmin(1M)
delete-message-security-provider

enables administrators to delete a message security provider

**Synopsis**

delete-message-security-provider [--help] [--target target]  
--layer message_layer  
provider_name

**Description**
The delete-message-security-provider subcommand enables administrators to delete a message security provider.

In terms of what happens when this subcommand is run, the provider-config sub-element for the given message layer (message-security-config element of domain.xml) is deleted. The domain.xml file specifies parameters and properties to the GlassFish Server. The options specified in the list below apply to attributes within the message-security-config and provider-config sub-elements of the domain.xml file.

If the message-layer (message-security-config attribute) does not exist, it is created, and then the provider-config is created under it.

This command is supported in remote mode only.

**Options**

If an option has a short option name, then the short option precedes the long option name. Short options have one dash whereas long options have two dashes.

---help

Displays the help text for the subcommand.

--target

Specifies the target from which you are deleting the message security provider. Valid values are

**server**

Deletes the message security provider from the default server instance server and is the default value

**domain**

Deletes the message security provider from the domain.

**cluster_name**

Deletes the message security provider from every server instance in the cluster.

**instance_name**

Deletes the message security provider from a particular sever instance.

--layer

The message-layer from which the provider has to be deleted. The default value is HttpServlet.
Operands

`provider_name`
The name of the provider used to reference the `provider-config` element.

Examples

**Example 1 Deleting a message security provider**
The following example shows how to delete a message security provider for a client.

```
asadmin> delete-message-security-provider
--layer SOAP mySecurityProvider
```

Exit Status

0
command executed successfully

1
error in executing the command

See Also

`create-message-security-provider(1), list-message-security-providers(1)`
`asadmin(1M)`
**delete-network-listener**

removes a network listener

**Synopsis**

```plaintext
delete-network-listener [--help]
(--target target)
listener-name
```

**Description**

The `delete-network-listener` subcommand removes the specified network listener. This subcommand is supported in remote mode only.

**Options**

--help
-?
Displays the help text for the subcommand.

--target
Deletes the network listener only from the specified target. Valid values are as follows:

- **server**
  Deletes the network listener from the default server instance. This is the default value.

- **configuration-name**
  Deletes the network listener from the specified configuration.

- **cluster-name**
  Deletes the network listener from all server instances in the specified cluster.

- **standalone-instance-name**
  Deletes the network listener from the specified standalone server instance.

**Operands**

`listener-name`
The name of the network listener to be deleted.

**Examples**

**Example 1 Deleting a Network Listener**

The following command deletes the network listener named `sampleListener`:

```plaintext
asadmin> delete-network-listener sampleListener
Command delete-network-listener executed successfully.
```
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-network-listener(1), list-network-listeners(1)
asadmin(1M)
**delete-node-config**

deletes a node that is not enabled for remote communication

**Synopsis**

deleート-node-config [--help] node-name

**Description**
The `delete-node-config` subcommand deletes a node that is not enabled for remote communication from the domain. This subcommand does not require the Distributed Component Object Model (DCOM) remote protocol or secure shell (SSH) to be configured.

This subcommand can delete only a node that is not enabled for remote communication. The subcommand for deleting other types of nodes depends on the type of the node:

- A node that is enabled for remote communication over DCOM must be deleted by using the `delete-node-dcom(1)` subcommand.
- A node that is enabled for remote communication over SSH must be deleted by using the `delete-node-ssh(1)` subcommand.

To determine whether a node is enabled for remote communication, use the `list-nodes(1)` subcommand.

No GlassFish Server instances must reside on the node that is being deleted. Otherwise, the subcommand fails. Before running this subcommand, delete any instances that reside on the node by using, for example, the `delete-instance(1)` subcommand or the `delete-local-instance(1)` subcommand.

---

**Note:** The predefined node `localhost-domain` cannot be deleted.

This subcommand is supported in remote mode only.

**Options**

--help
-?
Displays the help text for the subcommand.

**Operands**

*node-name*
The name of the node to delete. The node must not be enabled for communication over DCOM or SSH. Otherwise, an error occurs.
Examples

Example 1 Deleting a Node That Is Not Enabled for Remote Communication
This example deletes the node sj03, which is not enabled for remote communication.

```bash
asadmin> delete-node-config sj03
```

Command delete-node-config executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-node-config(1), create-node-dcom(1), delete-node-ssh(1),
delete-instance(1), delete-local-instance(1), install-node(1),
install-node-dcom(1), install-node-ssh(1), list-nodes(1), uninstall-node(1),
uninstall-node-dcom(1), uninstall-node-ssh(1), update-node-config(1),
update-node-dcom(1), update-node-ssh(1)

asadmin(1M)
delete-node-dcom

deletes a node that is enabled for communication over DCOM

Synopsis

delete-node-dcom [--help] [--uninstall={false|true}] [--force={false|true}] node-name

Description

The delete-node-dcom subcommand deletes a node that is enabled for communication over the Distributed Component Object Model (DCOM) remote protocol from the domain. The DCOM protocol is available only on Windows systems. This subcommand does not require DCOM to be configured.

This subcommand can delete only a node that is enabled for communication over DCOM. The subcommand for deleting other types of nodes depends on the type of the node:

- A node that is enabled for communication over secure shell (SSH) must be deleted by using the delete-node-ssh(1) subcommand.
- A node that is not enabled for communication must be deleted by using the delete-node-config(1) subcommand.

To determine whether a node is enabled for communication over DCOM, use the list-nodes(1) subcommand.

No GlassFish Server instances must reside on the node that is being deleted. Otherwise, the subcommand fails. Before running this subcommand, delete any instances that reside on the node by using, for example, the delete-instance(1) subcommand or the delete-local-instance(1) subcommand.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--uninstall

Specifies whether the GlassFish Server software is uninstalled from host that the node represents.

Possible values are as follows:

false
The GlassFish Server software is not uninstalled from the host (default).

ture
The GlassFish Server software is uninstalled from the host. By default, if any node except the predefined node localhost-domain resides on any host from which GlassFish Server software is being uninstalled, the subcommand fails. To uninstall
the GlassFish Server software from a host on which user-defined nodes reside, set the --force option to true. If the --force option is true, the subcommand removes the entire content of the parent of the base installation directory.

**--force**
If --uninstall is true, specifies whether the subcommand uninstalls the GlassFish Server software from a host even if a user-defined node resides on the host. Possible values are as follows:

**false**
If a user-defined node resides on a host, the software is not uninstalled and the subcommand fails (default).

If the --force option is false, the subcommand removes only the GlassFish Server software files. Other content if the parent of the base installation directory, such as configuration files, are not removed.

**true**
The subcommand uninstalls the GlassFish Server software from the host even if a user-defined node resides on the host.
If the --force option is true, the subcommand removes the entire content of the parent of the base installation directory.

**Operands**

node-name
The name of the node to delete. The node must enabled for communication over DCOM. Otherwise, an error occurs.

**Examples**

**Example 1 Deleting a Node That Is Enabled for Communication Over DCOM**
This example deletes the node xkyd, which is enabled for communication over DCOM.

```
asadmin> delete-node-dcom xkyd
Command delete-node-dcom executed successfully.
```

**Exit Status**

0
command executed successfully

1
error in executing the command

**See Also**

create-node-dcom(1), delete-node-config(1), delete-node-ssh(1), delete-instance(1), delete-local-instance(1), install-node-dcom(1), list-nodes(1), uninstall-node-dcom(1), update-node-dcom(1)
asadmin(1M)
delete-node-ssh

deletes a node that is enabled for communication over SSH

Synopsis

delete-node-ssh [--help]
    [--uninstall={false|true}] [--force={false|true}]
    node-name

Description

The delete-node-ssh subcommand deletes a node that is enabled for communication
over secure shell (SSH) from the domain. This subcommand does not require SSH to
be configured.

This subcommand can delete only a node that is enabled for communication over SSH.
The subcommand for deleting other types of nodes depends on the type of the node:

- A node that is enabled for communication over the Distributed Component Object
  Model (DCOM) remote protocol must be deleted by using the
    delete-node-dcom(1) subcommand.
- A node that is not enabled for remote communication must be deleted by using
  the delete-node-config(1) subcommand.

To determine whether a node is enabled for communication over SSH, use the
list-nodes(1) subcommand.

No GlassFish Server instances must reside on the node that is being deleted.
Otherwise, the subcommand fails. Before running this subcommand, delete any
instances that reside on the node by using, for example, the delete-instance(1)
subcommand or the delete-local-instance(1) subcommand.

This subcommand is supported in remote mode only.

Options

  --help
  -?
  Displays the help text for the subcommand.

  --uninstall
  Specifies whether the GlassFish Server software is uninstalled from host that the node
  represents.

  Possible values are as follows:

  false
      The GlassFish Server software is not uninstalled from the host (default).

  true
      The GlassFish Server software is uninstalled from the host. By default, if any node
      except the predefined node localhost-domain resides on any host from which
      GlassFish Server software is being uninstalled, the subcommand fails. To uninstalled
      the GlassFish Server software from a host on which user-defined nodes reside, set
the --force option to true. If the --force option is true, the subcommand removes the entire content of the parent of the base installation directory.

---force
If --uninstall is true, specifies whether the subcommand uninstalls the GlassFish Server software from a host even if a user-defined node resides on the host. Possible values are as follows:

false
If a user-defined node resides on a host, the software is not uninstalled and the subcommand fails (default).

If the --force option is false, the subcommand removes only the GlassFish Server software files. Other content if the parent of the base installation directory, such as configuration files, are not removed.

true
The subcommand uninstalls the GlassFish Server software from the host even if a user-defined node resides on the host.

If the --force option is true, the subcommand removes the entire content of the parent of the base installation directory.

Operands

node-name
The name of the node to delete. The node must enabled for communication over SSH. Otherwise, an error occurs.

Examples

Example 1 Deleting a Node That Is Enabled for Communication Over SSH
This example deletes the node eg1, which is enabled for communication over SSH.

asadmin> delete-node-ssh eg1
Command delete-node-ssh executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-node-dcom(1), create-node-ssh(1), delete-node-config(1),
delete-node-dcom(1), delete-instance(1), delete-local-instance(1),
install-node(1), install-node-dcom(1), install-node-ssh(1), list-nodes(1),
uninstall-node(1), uninstall-node-dcom(1), uninstall-node-ssh(1),
update-node-dcom(1), update-node-ssh(1)
asadmin(1M)
delete-password-alias

deleates a password alias

Synopsis

delete-password-alias [--help]

aliasname

Description

This subcommand deletes a password alias.

Options

--help
-?
Displays the help text for the subcommand.

Operands

aliasname

This is the name of the substitute password as it appears in domain.xml.

Examples

Example 1 Deleting a Password Alias

asadmin> delete-password-alias

jmspassword-alias

Command delete-password-alias executed successfully

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-password-alias(1), list-password-aliases(1), update-password-alias(1)
asadmin(1M)
delete-profiler

removes the profiler element

Synopsis

delete-profiler [--help] [--target target_name]

Description

The delete-profiler subcommand deletes the profiler element in the Java configuration. Only one profiler can exist at a time. If you attempt to create a profiler while one already exists, an error message is displayed and the existing profiler must be deleted.

For changes to take effect, the server must restarted.

This command is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target profiler element which you are deleting. Valid values are

server
Deletes the profiler element for the default server instance server and is the default value.

configuration_name
Deletes the profiler element for the named configuration.

cluster_name
Deletes the profiler element for every server instance in the cluster.

instance_name
Deletes the profiler element for a particular server instance.

Examples

Example 1 Deleting a Profile
This example deletes the profiler named sample_profiler.

asadmin> delete-profiler sample_profiler
Command delete-profiler executed successfully
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-profiler(1)

asadmin(1M)
delete-protocol

removes a protocol

Synopsis

default http-1

Description

The delete-protocol subcommand removes the specified protocol. This subcommand is supported in remote mode only.

Options

--help
-?  Displays the help text for the subcommand.

--target
Delete the protocol only from the specified target. Valid values are as follows:

  server
  Deletes the protocol from the default server instance. This is the default value.

  configuration-name
  Deletes the protocol from the specified configuration.

  cluster-name
  Deletes the protocol from all server instances in the specified cluster.

  standalone-instance-name
  Deletes the protocol from the specified standalone server instance.

Operands

protocol-name
The name of the protocol to be deleted.

Examples

Example 1 Deleting a Protocol
The following command deletes the protocol named http-1:

  asadmin> delete-protocol http-1
  Command delete-protocol executed successfully.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-protocol(1), list-protocols(1)
asadmin(1M)
delete-protocol-filter

removes a protocol filter

Synopsis

delete-protocol-filter [--help]
--protocol protocol-name
[--target server]
protocol-filter-name

Description

The delete-protocol-filter subcommand removes the specified protocol filter. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--protocol-name
The name of the associated protocol.

--target
Deletes the protocol filter only from the specified target. Valid values are as follows:

server
Deletes the protocol filter from the default server instance. This is the default value.

configuration-name
Deletes the protocol filter from the specified configuration.

cluster-name
Deletes the protocol filter from all server instances in the specified cluster.

standalone-instance-name
Deletes the protocol filter from the specified standalone server instance.

Operands

protocol-filter-name
The name of the protocol filter to be deleted.

Examples

Example 1 Deleting a Protocol Filter
The following command deletes the protocol filter named http1-filter:
asadmin> delete-protocol-filter --protocol http1 http1-filter
Command delete-protocol-filter executed successfully.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-protocol-filter(1), list-protocol-filters(1)

asadmin(1M)
delete-protocol-finder

removes a protocol finder

Synopsis

```
delte-protocol-finder [--help]
  --protocol protocol-name
  [--target server]
  protocol-finder-name
```

Description

The delete-protocol-finder subcommand removes the specified protocol finder. This subcommand is supported in remote mode only.

Options

```
--help
-?
Displays the help text for the subcommand.

--protocol-name
The name of the associated protocol.

--target
Deletes the protocol finder only from the specified target. Valid values are as follows:
  server
  Deletes the protocol finder from the default server instance. This is the default value.
  configuration-name
  Deletes the protocol finder from the specified configuration.
  cluster-name
  Deletes the protocol finder from all server instances in the specified cluster.
  standalone-instance-name
  Deletes the protocol finder from the specified standalone server instance.
```

Operands

```
protocol-finder-name
The name of the protocol finder to be deleted.
```

Exit Status

```
0
command executed successfully

1
error in executing the command
```
See Also

create-protocol-finder(1), list-protocol-finders(1)

asadmin(1M)
delete-resource-adapter-config

deletes the resource adapter configuration

Synopsis

deleteresource-adapter-config [--help] raname

Description

The delete-resource-adapter-config subcommand deletes the configuration information for the connector module.

This command is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option is deprecated.

Operands

raname
Specifies the connector module name.

Examples

Example 1 Deleting a Resource Adapter Configuration
This example deletes the configuration information for ral.

asadmin> delete-resource-adapter-config ral
Command delete-resource-adapter-config executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-resource-adapter-config(1), list-resource-adapter-configs(1)
asadmin(1M)
delete-resource-ref

removes a reference to a resource

Synopsis

delete-resource-ref [--help] [--target target] reference_name

Description

The delete-resource-ref subcommand removes from a cluster or an unclustered server instance a reference to a resource (for example, a JDBC resource). This effectively results in the removal of the resource from the JNDI tree of the targeted instance or cluster.

The target instance or instances making up the cluster need not be running or available for this subcommand to succeed. If one or more instances are not available, they will no longer load the resource in the JNDI tree the next time they start.

Removal of the reference does not result in removal of the resource from the domain. The resource is removed only by the delete subcommand for that resource (for example, delete-jdbc-resource).

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target from which you are removing the resource reference. Valid values are

server
Removes the resource reference from the default server instance server and is the default value.

cluster_name
Removes the resource reference from every server instance in the cluster.

instance_name
Removes the resource reference from the named unclustered server instance.

Operands

reference_name
The name or JNDI name of the resource.
Examples

Example 1  Removing a Reference to a Resource
This example removes a reference to the JMS destination resource jms/Topic on the cluster cluster1.

```bash
asadmin> delete-resource-ref --target cluster1 jms/Topic
resource-ref jms/Topic deleted successfully.
Command delete-resource-ref executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-resource-ref(1), list-resource-refs(1)
asadmin(1M)
delete-schedule

deletes an existing schedule

Synopsis

delete-schedule [--help]
schedule-name

Description

The delete-schedule subcommand deletes an existing schedule.

Note: You cannot delete a schedule that is in use by a domain backup configuration.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

schedule-name
Specifies the name of the schedule to delete.

Examples

Example 1 Deleting a Schedule
This example deletes the quarterly schedule.

asadmin> delete-schedule quarterly
Command delete-schedule executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-schedule(1), list-schedules(1)
asadmin(1M)
delete-ssl

deletes the SSL element in the selected HTTP listener, IIOP listener, or IIOP service.

Synopsis

dele-ssl [--help]
[--target target]
--type listener_or_service_type
listener_id

Description

The delete-ssl subcommand deletes the SSL element in the selected HTTP listener, IIOP listener, or IIOP service.

The listener_id is not required if the --type is iiop-service.

This subcommand is supported in remote mode only.

Options

If an option has a short option name, then the short option precedes the long option name. Short options have one dash whereas long options have two dashes.

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target on which you are configuring the ssl element. The following values are valid:

server
Specifies the server in which the iiop-service or HTTP/IIOP listener is to be unconfigured for SSL.

config
Specifies the configuration that contains the HTTP/IIOP listener or iiop-service for which SSL is to be unconfigured.

cluster
Specifies the cluster in which the HTTP/IIOP listener or iiop-service is to be unconfigured for SSL. All the server instances in the cluster will get SSL unconfigured for the respective listener or iiop-service.

instance
Specifies the instance in which the HTTP/IIOP listener or iiop-service is to be unconfigured for SSL.

--type
The type of service or listener for which the SSL is deleted. The type must be one of the following types:

- http-listener
Operands

- `iiop-listener`
- `iiop-service`

Example 1 Deleting an SSL element from an HTTP listener

The following example shows how to delete an SSL element from an HTTP listener named `http-listener-1`.

```
asadmin> delete-ssl
--type http-listener http-listener-1
Command delete-ssl executed successfully.
```

Exit Status

- 0: command executed successfully
- 1: error in executing the command

See Also

- `create-ssl(1)`
- `asadmin(1M)`
delete-system-property

removes a system property of the domain, configuration, cluster, or server instance, one at a time

Synopsis

delete-system-property [--help] [--target target_name ] [property_name]

Description

The delete-system-property subcommand deletes a system property of a domain, configuration, cluster, or server instance. Make sure that the system property is not referenced elsewhere in the configuration before deleting it.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target on which you are deleting the system properties. The valid targets for this subcommand are instance, cluster, configuration, domain, and server. Server is the default option.

Operands

property_name
The name of the system property to remove.

Examples

Example 1 Deleting a System Property
This example deletes the system property named http-listener-port.

asadmin> delete-system-property http-listener-port
Command delete-system-property executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand
See Also

create-system-properties(1), list-system-properties(1)

asadmin(1M)
**delete-threadpool**

removes a thread pool

**Synopsis**

```
delete-threadpool [--help] [--target target] threadpool-id
```

**Description**

Removes the thread pool with the specified ID. This subcommand is supported in remote mode only.

**Options**

```
--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target from which you are removing the thread pool.
Valid values are as follows:

server
Deletes the thread pool for the default GlassFish Server instance server and is the default value.

configuration-name
Deletes the thread pool for the named configuration.

cluster-name
Deletes the thread pool for every instance in the cluster.

instance-name
Deletes the thread pool for a particular instance.
```

**Operands**

```
threadpool-id
An ID for the work queue, for example, thread-pool1, threadpool-2, and so forth.
```

**Examples**

**Example 1 Deleting a Thread Pool**

This example deletes threadpool-1.

```
asadmin> delete-threadpool threadpool-1
Command delete-threadpool executed successfully
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-threadpool(1), list-threadpools(1)
asadmin(1M)
delete-transport

removes a transport

Synopsis

delete-transport [--help]
                    [--target target]
                    transport-name

Description

The delete-transport subcommand removes the specified transport. This
subcommand is supported in remote mode only.

Options

--help
  -?
  Displays the help text for the subcommand.

--target
  Deletes the transport only from the specified target. Valid values are as follows:

  server
  Deletes the transport from the default server instance. This is the default value.

  configuration-name
  Deletes the transport from the specified configuration.

  cluster-name
  Deletes the transport from all server instances in the specified cluster.

  standalone-instance-name
  Deletes the transport from the specified standalone server instance.

Operands

transport-name
  The name of the transport to be deleted.

Examples

Example 1 Deleting a Transport
The following command deletes the transport named http1-trans:

asadmin> delete-transport http1-trans
Command delete-transport executed successfully.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-transport(1), list-transports(1)
asadmin(1M)
delete-virtual-server

removes a virtual server

Synopsis

delete-virtual-server [--help]

[--target target] virtual-server-id

Description

The delete-virtual-server subcommand removes the virtual server with the specified virtual server ID. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Deletes the virtual server only from the specified target. Valid values are as follows:

  server
  Deletes the virtual server from the default server instance. This is the default value.

  configuration-name
  Deletes the virtual server from the specified configuration.

  cluster-name
  Deletes the virtual server from all server instances in the specified cluster.

  standalone-instance-name
  Deletes the virtual server from the specified standalone server instance.

Operands

virtual-server-id
The unique identifier for the virtual server to be deleted.

Examples

Example 1 Deleting a Virtual Server
The following command deletes the virtual server named sample_vs1:

asadmin> delete-virtual-server sample_vs1
Command delete-virtual-server executed successfully.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-virtual-server(1), list-virtual-servers(1)
asadmin(1M)
deploy

deploys the specified component

Synopsis

deploy [--help]
[--force={false|true}]
[--virtualservers virtual_servers]
[--contextroot context_root]
[--precompilejsp={false|true}]
[--verify={false|true}]
[--name component_name]
[--upload={true|false}]
[--retrieve local_dirpath]
[--dbvendorname dbvendorname]
[--createtables={true|false}|--dropandcreatetables={true|false}]
[-- uniquetablenames={true|false}]
[--deploymentplan deployment_plan]
[--enabled={true|false}]
[--generatermistubs={false|true}]
[--availabilityenabled={false|true}]
[--asyncreplication={true|false}]
[--lbenabled={true|false}]
[--keepstate={false|true}]
[--libraries jar_file[,jar_file]*]
[--target target]
[--type pkg-type]
[--properties(name=value)[.:name=value]]
[file_archive|filepath]

Description

The deploy subcommand deploys applications to the server. Applications can be enterprise applications, web applications, Enterprise JavaBeans (EJB) modules, connector modules, and application client modules. If the component is already deployed or already exists, it is forcibly redeployed if the --force option is set to true (default is false).

The --createtables and --dropandcreatetables options are boolean flags and therefore can take the values of true or false. These options are only used during deployment of CMP beans that have not been mapped to a database (that is, no sun-cmp-mappings.xml descriptor is provided in the module's META-INF directory). They are ignored otherwise.

The --createtables and --dropandcreatetables options are mutually exclusive; only one should be used. If drop and/or create tables fails, the deployment does not fail; a warning message is provided in the log file.

This subcommand is supported in remote mode only.
Options

--help
Displays the help text for the subcommand.

--force
If set to true, redeploy the component even if the specified component has already been deployed or already exists. Default is false.

--virtualservers
One or more virtual server IDs. Multiple IDs are separated by commas.

--contextroot
Valid only if the archive is a web module. It is ignored for other archive types; defaults to filename without extension.

--precompilejsp
By default this option does not allow the JSP to be precompiled during deployment. Instead, JSPs are compiled during runtime. Default is false.

--verify
If set to true and the required verifier packages are installed from the Update Tool, the syntax and semantics of the deployment descriptor is verified. Default is false.

--name
Name of the deployable component.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter or number. It can contain alphanumeric characters plus underscore (_), dash (-), and period (.) characters. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.

--upload
Specifies whether the subcommand uploads the file to the DAS. In most situations, this option can be omitted.

Valid values are as follows:

false
The subcommand does not upload the file and attempts to access the file through the specified file name. If the DAS cannot access the file, the subcommand fails.

For example, the DAS might be running as a different user than the administration user and does not have read access to the file. In this situation, the subcommand fails if the --upload option is false.

ture
The subcommand uploads the file to the DAS over the network connection.

The default value depends on whether the DAS is on the host where the subcommand is run or is on a remote host.

- If the DAS is on the host where the subcommand is run, the default is false.
- If the DAS is on a remote host, the default is true.

If a directory filepath is specified, this option is ignored.
--retrieve
Retrieves the client stub JAR file from the server machine to the local directory.

--dbvendorname
Specifies the name of the database vendor for which tables are created. Supported values include db2, mssql, mysql, oracle, derby, javadb, postgresql, and sybase. These values are case-insensitive. If not specified, the value of the database-vendor-name attribute in glassfish-ejb-jar.xml is used. If no value is specified, a connection is made to the resource specified by the jndi-name subelement of the cmp-resource element in the glassfish-ejb-jar.xml file, and the database vendor name is read. If the connection cannot be established, or if the value is not recognized, SQL-92 compliance is presumed.

--createtables
If specified as true, creates tables at deployment of an application with unmapped CMP beans. If specified as false, tables are not created. If not specified, the value of the create-tables-at-deploy entry in the cmp-resource element of the glassfish-ejb-jar.xml file determines whether or not tables are created. No unique constraints are created for the tables.

--dropandcreatetables
If specified as true when the component is redeployed, the tables created by the previous deployment are dropped before creating the new tables. Applies to deployed applications with unmapped CMP beans. Preexisting tables will not be dropped on the initial deployment of an application or on a deployment that follows an explicit undeploy. If specified as false, tables are neither dropped nor created. If not specified, the tables are dropped if the drop-tables-at-undeploy entry in the cmp-resource element of the glassfish-ejb-jar.xml file is set to true, and the new tables are created if the create-tables-at-deploy entry in the cmp-resource element of the glassfish-ejb-jar.xml file is set to true.

--uniquetablenames
Guarantees unique table names for all the beans and results in a hash code added to the table names. This is useful if you have an application with case-sensitive bean names. Applies to applications with unmapped CMP beans.

--deploymentplan
Deploys the deployment plan, which is a JAR file that contains GlassFish Server descriptors. Specify this option when deploying a pure EAR file. A pure EAR file is an EAR without GlassFish Server descriptors.

--enabled
Allows users to access the application. If set to false, users will not be able to access the application. This option enables the application on the specified target instance or cluster. If you deploy to the target domain, this option is ignored, since deploying to the domain doesn’t deploy to a specific instance or cluster. The default is true.

--generatermistubs
If set to true, static RMI-IIOP stubs are generated and put into the client.jar. If set to false, the stubs are not generated. Default is false.

--availabilityenabled
This option controls whether high-availability is enabled for web sessions and for stateful session bean (SFSB) checkpointing and potentially passivation. If set to false (default) all web session saving and SFSB checkpointing is disabled for the specified application, web application, or EJB module. If set to true, the specified application or
module is enabled for high-availability. Set this option to true only if high availability is configured and enabled at higher levels, such as the server and container levels.

--asyncreplication
This option controls whether web session and SFSB states for which high availability is enabled are first buffered and then replicated using a separate asynchronous thread. If set to true (default), performance is improved but availability is reduced. If the instance where states are buffered but not yet replicated fails, the states are lost. If set to false, performance is reduced but availability is guaranteed. States are not buffered but immediately transmitted to other instances in the cluster.

--lbenabled
This option controls whether the deployed application is available for load balancing. The default is true.

--keepstate
This option controls whether web sessions, SFSB instances, and persistently created EJB timers are retained between redeployments.

The default is false. This option is supported only on the default server instance, named server. It is not supported and ignored for any other target.

Some changes to an application between redeployments prevent this feature from working properly. For example, do not change the set of instance variables in the SFSB bean class.

For web applications, this feature is applicable only if in the glassfish-web-app.xml file the persistence-type attribute of the session-manager element is file.

For stateful session bean instances, the persistence type without high availability is set in the server (the sfssb-persistence-type attribute) and must be set to file, which is the default and recommended value.

If any active web session, SFSB instance, or EJB timer fails to be preserved or restored, none of these will be available when the redeployment is complete. However, the redeployment continues and a warning is logged.

To preserve active state data, GlassFish Server serializes the data and saves it in memory. To restore the data, the class loader of the newly redeployed application deserializes the data that was previously saved.

--libraries
A comma-separated list of library JAR files. Specify the library JAR files by their relative or absolute paths. Specify relative paths relative to domain-dir/lib/applibs. The libraries are made available to the application in the order specified.

--target
Specifies the target to which you are deploying. Valid values are:

server
Deploys the component to the default server instance server and is the default value.

domain
Deploys the component to the domain. If domain is the target for an initial deployment, the application is deployed to the domain, but no server instances or clusters reference the application. If domain is the target for a redeployment (the --force option is set to true), and dynamic reconfiguration is enabled for the clusters or server instances that reference the application, the referencing clusters
or server instances automatically get the new version of the application. If
redeploying, and dynamic configuration is disabled, the referencing clusters or
server instances do not get the new version of the application until the clustered or
standalone server instances are restarted.

**cluster_name**
Deploys the component to every server instance in the cluster.

**instance_name**
Deploys the component to a particular stand-alone server instance.

**--type**
The packaging archive type of the component that is being deployed. Possible values
are as follows:

- **osgi**
  The component is packaged as an OSGi Alliance bundle.

The **--type** option is optional. If the component is packaged as a regular archive, omit
this option.

**--properties or --property**
Optional keyword-value pairs that specify additional properties for the deployment.
The available properties are determined by the implementation of the component that
is being deployed or redeployed. The **--properties** option and the **--property** option
are equivalent. You can use either option regardless of the number of properties that
you specify.

You can specify the following properties for a deployment:

**jar-signing-alias**
Specifies the alias for the security certificate with which the application client
container JAR file is signed. Java Web Start will not run code that requires elevated
permissions unless it resides in a JAR file signed with a certificate that the user's
system trusts. For your convenience, GlassFish Server signs the JAR file
automatically using the certificate with this alias from the domain's keystore. Java
Web Start then asks the user whether to trust the code and displays the GlassFish
Server certificate information. To sign this JAR file with a different certificate, add
the certificate to the domain keystore, then use this property. For example, you can
use a certificate from a trusted authority, which avoids the Java Web Start prompt,
or from your own company, which users know they can trust. Default is **s1as**, the
alias for the self-signed certificate created for every domain.

**java-web-start-enabled**
Specifies whether Java Web Start access is permitted for an application client
module. Default is true.

**compatibility**
Specifies the GlassFish Server release with which to be backward compatible in
terms of JAR visibility requirements for applications. The only allowed value is **v2**,
which refers to Sun GlassFish Enterprise Server version 2 or Sun Java System
Application Server version 9.1 or 9.1.1. The Java EE 6 platform specification
imposes stricter requirements than Java EE 5 did on which JAR files can be visible
to various modules within an EAR file. In particular, application clients must not
have access to EJB JAR files or other JAR files in the EAR file unless references use
the standard Java SE mechanisms (extensions, for example) or the Java EE
library-directory mechanism. Setting this property to **v2** removes these Java EE 6
restrictions.
keepSessions=(false|true)
Superseded by the --keepstate option.

If the --force option is set to true, this property can by used to specify whether active sessions of the application that is being redeployed are preserved and then restored when the redeployment is complete. Applies to HTTP sessions in a web container. Default is false.

false
Active sessions of the application are not preserved and restored (default).

true
Active sessions of the application are preserved and restored.

If any active session of the application fails to be preserved or restored, none of the sessions will be available when the redeployment is complete. However, the redeployment continues and a warning is logged.

To preserve active sessions, GlassFish Server serializes the sessions and saves them in memory. To restore the sessions, the class loader of the newly redeployed application deserializes any sessions that were previously saved.

preserveAppScopedResources
If set to true, preserves any application-scoped resources and restores them during redeployment. Default is false.

Other available properties are determined by the implementation of the component that is being redeployed.

Operands

file_archive|filepath
The path to the archive that contains the application that is being deployed. This path can be a relative path or an absolute path.

The archive can be in either of the following formats:

- An archive file, for example, /export/JEE_apps/hello.war.
  If the --upload option is set to true, this is the path to the deployable file on the local client machine. If the --upload option is set to false, this is the path to the file on the server machine.

- A directory that contains the exploded format of the deployable archive. This is the path to the directory on the server machine.
  If you specify a directory, the --upload option is ignored.

Examples

Example 1 Deploying an Enterprise Application
This example deploys the enterprise application packaged in the Cart.ear file to the default server instance server. You can use the --target option to deploy to a different server instance or to a cluster.

asadmin> deploy Cart.ear
Application deployed successfully with name Cart.
Command deploy executed successfully
Example 2  Deploying a Web Application With the Default Context Root
This example deploys the web application in the hello.war file to the default server instance server. You can use the --target option to deploy to a different server instance or to a cluster.

asadmin> deploy hello.war
Application deployed successfully with name hello.
Command deploy executed successfully

Example 3  Forcibly Deploying a Web Application With a Specific Context Root
This example forcibly deploys the web application in the hello.war file. The context root of the deployed web application is greetings. If the application has already been deployed, it is redeployed.

asadmin> deploy --force=true --contextroot greetings hello.war
Application deployed successfully with name hello.
Command deploy executed successfully

Example 4  Deploying an Enterprise Bean
This example deploys a component based on the EJB specification (enterprise bean) with CMP and creates the database tables used by the bean.

This example uses the --target option. The target in this example is an existing cluster, cluster1.

asadmin> deploy --createtables=true --target cluster1 EmployeeEJB.jar
Application deployed successfully with name EmployeeEJB.
Command deploy executed successfully

Example 5  Deploying a Connector Module
This example deploys a connector module that is packaged in an RAR file.

This example uses the --target option. The target in this example is an existing standalone server instance that does not belong to a cluster.

asadmin> deploy --target myinstance jdbcra.rar
Application deployed successfully with name jdbcra.
Command deploy executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

redeploy(1), list-components(1), undeploy(1)
asadmin(1M)

Oracle GlassFish Server Application Deployment Guide
deploydir

deploys an exploded format of application archive

Synopsis

deploydir [--help]
[--force={false|true}]
[--virtualservers virtual_servers]
[--contextroot context_root]
[--verify={false|true}]
[--precompilejsp={false|true}]
[--name component-name]
[--retrieve local_dirpath]
[--uniquetablenames={true|false}]
[--dbvendorname dbvendorname]
[--createtables={false|true}|--dropandcreatetables={false|true}]
[--deploymentplan deployment_plan]
[--enabled={true|false}]
[--generatemistubs={false|true}]
[--availabilityenabled={false|true}]
[--asyncreplication={true|false}]
[--lbenabled={true|false}]
[--keepstate={false|true}]
[--libraries jar_file[,jar_file]*]
[--target target]
[--type pkg-type]
[--properties(name=value)[(:name=value)*]]
dirpath

Description

Note: The deploydir subcommand is deprecated. Use the deploy subcommand instead.

The deploydir subcommand deploys an application directly from a development directory. The appropriate directory hierarchy and deployment descriptors conforming to the Java EE specification must exist in the deployment directory.

Directory deployment is for advanced developers only. Do not use deploydir in production environments. Instead, use the deploy subcommand. Directory deployment is only supported on localhost, that is, the client and server must reside on the same machine. For this reason, the only values for the --host option are:

• localhost
• The value of the $HOSTNAME environment variable
• The IP address of the machine

If the --uniquetablenames, --createtables, and --dropandcreatetables options are not specified, the entries in the deployment descriptors are used.

The --force option makes sure the component is forcefully (re)deployed even if the specified component has already been deployed or already exists. Set the --force
option to false for an initial deployment. If the specified application is running and the
--force option is set to false, the subcommand fails.
This subcommand is supported in remote mode only.

Options

--help
Displays the help text for the subcommand.

--force
If set to true, redeploys the component even if the specified component has already
been deployed or already exists. Default is false.

--virtualservers
One or more virtual server IDs. Multiple IDs are separated by commas.

--contextroot
Valid only if the archive is a web module. It is ignored for other archive types; defaults
to filename without extension.

--precompilejsp
By default this option does not allow the JSP to be precompiled during deployment.
Instead, JSPs are compiled during runtime. Default is false.

--verify
If set to true and the required verifier packages are installed from the Update Tool, the
syntax and semantics of the deployment descriptor is verified. Default is false.

--name
Name of the deployable component.
The name can include an optional version identifier, which follows the name and is
separated from the name by a colon (:). The version identifier must begin with a letter
or number. It can contain alphanumeric characters plus underscore (_), dash (-), and
period (.) characters. For more information about module and application versions,
see "Module and Application Versions" in Oracle GlassFish Server Application
Deployment Guide.

--retrieve
Retrieves the client stub JAR file from the server machine to the local directory.

--dbvendorname
Specifies the name of the database vendor for which tables are created. Supported
values include db2, mssql, mysql, oracle, derby, javadb, postgresql, and sybase.
These values are case-insensitive. If not specified, the value of the
database-vendor-name attribute in glassfish-ejb-jar.xml is used. If no value is
specified, a connection is made to the resource specified by the jndi-name subelement
of the cmp-resource element in the glassfish-ejb-jar.xml file, and the database
vendor name is read. If the connection cannot be established, or if the value is not
recognized, SQL-92 compliance is presumed.

--createtables
If specified as true, creates tables at deployment of an application with unmapped
CMP beans. If specified as false, tables are not created. If not specified, the value of the
create-tables-at-deploy entry in the cmp-resource element of the
glassfish-ejb-jar.xml file determines whether or not tables are created. No unique constraints are created for the tables.

**--dropandcreatedtables**
If specified as true when the component is redeployed, the tables created by the previous deployment are dropped before creating the new tables. Applies to deployed applications with unmapped CMP beans. Preexisting tables will not be dropped on the initial deployment of an application or on a deployment that follows an explicit undeploy. If specified as false, tables are neither dropped nor created. If not specified, the tables are dropped if the drop-tables-at-undeploy entry in the cmp-resource element of the glassfish-ejb-jar.xml file is set to true, and the new tables are created if the create-tables-at-deploy entry in the cmp-resource element of the glassfish-ejb-jar.xml file is set to true.

**--uniquetablenames**
 Guarantees unique table names for all the beans and results in a hash code added to the table names. This is useful if you have an application with case-sensitive bean names. Applies to applications with unmapped CMP beans.

**--deploymentplan**
Deploys the deployment plan, which is a JAR file that contains GlassFish Server descriptors. Specify this option when deploying a pure EAR file. A pure EAR file is an EAR without GlassFish Server descriptors.

**--enabled**
Allows users to access the application. If set to false, users will not be able to access the application. This option enables the application on the specified target instance or cluster. If you deploy to the target domain, this option is ignored, since deploying to the domain doesn’t deploy to a specific instance or cluster. The default is true.

**--generatermistubs**
If set to true, static RMI-IIOP stubs are generated and put into the client.jar. If set to false, the stubs are not generated. Default is false.

**--availabilityenabled**
This option controls whether high-availability is enabled for web sessions and for stateful session bean (SFSB) checkpointing and potentially passivation. If set to false (default) all web session saving and SFSB checkpointing is disabled for the specified application, web application, or EJB module. If set to true, the specified application or module is enabled for high-availability. Set this option to true only if high availability is configured and enabled at higher levels, such as the server and container levels.

**--asyncreplication**
This option controls whether web session and SFSB states for which high availability is enabled are first buffered and then replicated using a separate asynchronous thread. If set to true (default), performance is improved but availability is reduced. If the instance where states are buffered but not yet replicated fails, the states are lost. If set to false, performance is reduced but availability is guaranteed. States are not buffered but immediately transmitted to other instances in the cluster.

**--lbenabled**
This option controls whether the deployed application is available for load balancing. The default is true.

**--keepstate**
This option controls whether web sessions, SFSB instances, and persistently created EJB timers are retained between redeployments.
The default is false. This option is supported only on the default server instance, named server. It is not supported and ignored for any other target.

Some changes to an application between redeployments prevent this feature from working properly. For example, do not change the set of instance variables in the SFSB bean class.

For web applications, this feature is applicable only if in the glassfish-web-app.xml file the persistence-type attribute of the session-manager element is file.

For stateful session bean instances, the persistence type without high availability is set in the server (the sfsb-persistence-type attribute) and must be set to file, which is the default and recommended value.

If any active web session, SFSB instance, or EJB timer fails to be preserved or restored, none of these will be available when the redeployment is complete. However, the redeployment continues and a warning is logged.

To preserve active state data, GlassFish Server serializes the data and saves it in memory. To restore the data, the class loader of the newly redeployed application deserializes the data that was previously saved.

--libraries
A comma-separated list of library JAR files. Specify the library JAR files by their relative or absolute paths. Specify relative paths relative to domain-dir/lib/applibs. The libraries are made available to the application in the order specified.

--target
Specifies the target to which you are deploying. Valid values are:

server
Deploys the component to the default server instance server and is the default value.

domain
Deploys the component to the domain. If domain is the target for an initial deployment, the application is deployed to the domain, but no server instances or clusters reference the application. If domain is the target for a redeployment (the --force option is set to true), and dynamic reconfiguration is enabled for the clusters or server instances that reference the application, the referencing clusters or server instances automatically get the new version of the application. If redeploying, and dynamic configuration is disabled, the referencing clusters or server instances do not get the new version of the application until the clustered or standalone server instances are restarted.

cluster_name
Deploys the component to every server instance in the cluster.

instance_name
Deploys the component to a particular stand-alone server instance.

--type
The packaging archive type of the component that is being deployed. Possible values are as follows:

osgi
The component is packaged as an OSGi Alliance bundle.

The --type option is optional. If the component is packaged as a regular archive, omit this option.
--properties OR --property
Optional keyword-value pairs that specify additional properties for the deployment. The available properties are determined by the implementation of the component that is being deployed or redeployed. The --properties option and the --property option are equivalent. You can use either option regardless of the number of properties that you specify.

You can specify the following properties for a deployment:

**jar-signing-alias**
Specifies the alias for the security certificate with which the application client container JAR file is signed. Java Web Start will not run code that requires elevated permissions unless it resides in a JAR file signed with a certificate that the user’s system trusts. For your convenience, GlassFish Server signs the JAR file automatically using the certificate with this alias from the domain's keystore. Java Web Start then asks the user whether to trust the code and displays the GlassFish Server certificate information. To sign this JAR file with a different certificate, add the certificate to the domain's keystore, then use this property. For example, you can use a certificate from a trusted authority, which avoids the Java Web Start prompt, or from your own company, which users know they can trust. Default is s1as, the alias for the self-signed certificate created for every domain.

**java-web-start-enabled**
Specifies whether Java Web Start access is permitted for an application client module. Default is true.

**compatibility**
Specifies the GlassFish Server release with which to be backward compatible in terms of JAR visibility requirements for applications. The only allowed value is v2, which refers to Sun GlassFish Enterprise Server version 2 or Sun Java System Application Server version 9.1 or 9.1.1. The Java EE 6 platform specification imposes stricter requirements than Java EE 5 did on which JAR files can be visible to various modules within an EAR file. In particular, application clients must not have access to EJB JAR files or other JAR files in the EAR file unless references use the standard Java SE mechanisms (extensions, for example) or the Java EE library-directory mechanism. Setting this property to v2 removes these Java EE 6 restrictions.

**keepSessions={false|true}**
Superseded by the --keepstate option.

If the --force option is set to true, this property can be used to specify whether active sessions of the application that is being redeployed are preserved and then restored when the redeployment is complete. Applies to HTTP sessions in a web container. Default is false.

    false
    Active sessions of the application are *not* preserved and restored (default).

    true
    Active sessions of the application are preserved and restored.

If any active session of the application fails to be preserved or restored, none of the sessions will be available when the redeployment is complete. However, the redeployment continues and a warning is logged.
To preserve active sessions, GlassFish Server serializes the sessions and saves them in memory. To restore the sessions, the class loader of the newly redeployed application deserializes any sessions that were previously saved.

**preserveAppScopedResources**
If set to `true`, preserves any application-scoped resources and restores them during redeployment. Default is `false`.

Other available properties are determined by the implementation of the component that is being redeployed.

**Operands**

**dirpath**
Path to the directory containing the exploded format of the deployable archive. This is the path to the directory on the server machine.

**Examples**

**Example 1 Deploying an Application From a Directory**
In this example, the exploded application to be deployed is in the
/home/temp/sampleApp directory. Because the `--force` option is set to true, if an application of that name already exists, the application is redeployed.

```bash
asadmin> deploydir --force=true --precompilejsp=true /home/temp/sampleApp
Application deployed successfully with name sampleApp.
WARNING : deploydir command deprecated. Please use deploy command instead.
Command deploydir executed successfully
```

**Exit Status**

0  
subcommand executed successfully

1  
error in executing the subcommand

**See Also**

`deploy(1), redeploy(1), undeploy(1)`

`asadmin(1M)`

*Oracle GlassFish Server Application Deployment Guide*
disable

disables the component

Synopsis

disable [--help] [--target target_name] component_name

Description

The disable subcommand immediately disables the specified deployed component. If the component has not been deployed, an error message is returned. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target on which you are disabling the component. Valid values are:

server
Disable the component on the default server instance server and is the default value.

domain_name
Disable the component on the named domain.

cluster_name
Disable the component on every server instance in the cluster.

instance_name
Disable the component on a particular clustered or stand-alone server instance.

Operands

component_name
name of the component to be disabled.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter or number. It can contain alphanumeric characters plus underscore (_), dash (-), and period (.) characters. To disable multiple versions, you can use an asterisk (*) as a wildcard character. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.
Examples

Example 1 Disabling a Component
This example disables the deployed component sampleApp.

asadmin> disable sampleApp
Command disable executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

deploy(1), undeploy(1), enable(1)
asadmin(1M)

Oracle GlassFish Server Application Deployment Guide
disable-backup-config

disables automatic backups defined by a domain backup configuration

Synopsis

disable-backup-config [--help]  
backup-config-name

Description

The disable-backup-config subcommand sets the autobackupenabled option of a domain backup configuration to false, thus disabling automatic backups for the domain backup configuration.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

backup-config-name
The name of the domain backup configuration to disable.

Examples

Example 1 Disabling Automatic Backups
This example disables automatic backups for the weekly-backup domain backup configuration. The example shows list-backup-configs commands before and after the disable-backup-config command to demonstrate that the autobackupenabled value changes from true to false.

asadmin> list-backup-configs --long weekly-backup
Name of Backup Config :weekly-backup
Auto Backup Enabled   :true
Schedule              :weekly
Recycle Limit         :5
Config Only backup    :true
Active Backup Enabled :false
Backup Directory      :null
Last Backup Attempt   :
Last Successful Backup :

Schedule Details:
NAME    SECOND  MINUTE  HOUR  DAY OF WEEK  DAY OF MONTH  MONTH  YEAR
weekly  0       0       0     Sun          *             *      *

asadmin> disable-backup-config weekly-backup
Command disable-backup-config executed successfully.

asadmin> list-backup-configs --long weekly-backup
Name of Backup Config : weekly-backup
Auto Backup Enabled   : false
Schedule              : weekly
Recycle Limit         : 5
Config Only backup    : true
Active Backup Enabled : false
Backup Directory      : null
Last Backup Attempt   :
Last Successful Backup:

Schedule Details:
NAME    SECOND  MINUTE  HOUR  DAY OF WEEK  DAY OF MONTH  MONTH  YEAR
weekly   0       0       0     Sun          *             *      *

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-backup-config(1), delete-backup-config(1), enable-backup-config(1),
list-backup-configs(1)

asadmin(1M)
disable-http-lb-application

disables an application managed by a load balancer

Synopsis

disable-http-lb-application [--help] [--timeout 30]  
--name application_name target

Description

The disable disable-http-lb-application subcommand disables an application for load balancing. The disabled application goes offline for load balancing with minimal impact to users. Disabling an application gives a finer granularity of control than disabling a server instance and is most useful when a cluster is hosting multiple independent applications.

Once the application is disabled and the changes have been applied to the load balancer, new requests for the application are not forwarded to the target. Existing sessions continue to access the application until the timeout is reached. This process is known as quiescing.

If an application is deployed across multiple clusters, use this subcommand to disable it in one cluster while leaving it enabled in others.

If an application is deployed to a single server instance, use this subcommand to disable it in that instance while leaving the instance itself enabled.

---

Note: This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

---

Options

--help

Displays the help text for the subcommand.

--timeout

The timeout (in minutes) to wait before disabling the specified application. This time allows for the graceful shutdown (quiescing) of the specified application. The default value is 30 minutes. The minimum value is 1 minute.

--name

The name of the application to be disabled.

Operands

target

This operand specifies the server instance or cluster on which to disable the application. Valid values are:

- cluster_name: The name of a target cluster.
- instance_name- The name of a target server instance.

Examples

Example 1  Disabling an Application for Load Balancing
This example, disables an application for load balancing

asadmin> disable-http-lb-application --name webapps-simple mycluster

Command disable-http-lb-application executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

enable-http-lb-application(1)

asadmin(1M)
disable-http-lb-server

disables a server or cluster managed by a load balancer

Synopsis


Description

The disable-http-lb-server subcommand disables a standalone server or cluster of
servers for load balancing. The disabled server instance or cluster goes offline for load
balancing with a minimum impact to users.

Once the target has been disabled and the changes have been applied to the load
balancer, the load balancer stops assigning new requests to the target. Session requests
with sessions created before disabling the target continue to be assigned to that target
until the timeout is reached. This process is known as quiescing.

Changes are applied the load balancer automatically. You can also manually export the
configuration using export-http-lb-config and copy it to the load balancer.

Note: This subcommand is only applicable to Oracle GlassFish
Server. This subcommand is not applicable to GlassFish Server

Options

--help
-?
Displays the help text for the subcommand.

--timeout

The timeout (in minutes) to wait before disabling the specified target. This time allows
for the graceful shutdown (quiescing) of the specified target. The default value is 30
minutes. The minimum value is 1 minute.

Operands

target This operand specifies which server instances and clusters to disable. Valid values are:

- cluster_name- The name of a target cluster.
- instance_name- The name of a target server instance.

Examples

Example 1 Disabling a Cluster for Load Balancing
This example disables load balancing for a cluster named mycluster.

asadmin> disable-http-lb-server mycluster
Command disable-http-lb-server executed successfully.

**Exit Status**

0
subcommand executed successfully

1
error in executing the subcommand

**See Also**

create-http-lb-ref(1), enable-http-lb-server(1)

asadmin(1M)
disable-monitoring

disables monitoring for the server or for specific monitorable modules

Synopsis

disable-monitoring [--help] [--modules module-name][:module-name]*

Description

The disable-monitoring subcommand is used to turn off monitoring for GlassFish Server or for particular modules during runtime. Changes are dynamic, that is, server restart is not required.

Running the disable-monitoring subcommand without the --module option disables the monitoring service by setting the monitoring-enabled attribute of the monitoring-service element to false. The individual modules retain their monitoring levels, but no monitoring data is generated because the entire monitoring service is disabled.

This subcommand used with the --modules option disables monitoring for a module by setting the monitoring level to OFF. The status of the monitoring service is not affected. For a list of monitorable modules, see the --modules option in this help page.

An alternative method for disabling monitoring is to use the set subcommand. In this case, the server must be restarted for changes to take effect.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--modules
Disables the specified module or modules by setting the monitoring level to OFF. Multiple modules are separated by : (colon). Monitorable modules include connector-connection-pool, connector-service, ejb-container, http-service, jdbc-connection-pool, jersey, jpa, jms-service, jvm, security, thread-pool, transaction-service, web-container, and web-services-container. Additional modules can be listed by using the get subcommand.

Operands

Examples

Example 1 Disabling the Monitoring Service for GlassFish Server
This example disables monitoring for GlassFish Server in general by setting the enable-monitoring flag to false (default is true).
asadmin> **disable-monitoring**
Command disable-monitoring executed successfully

**Example 2 Disabling Monitoring for the Web and EJB Containers**
This example disables monitoring for specific containers. Their monitoring levels will be set to OFF.

asadmin> **disable-monitoring --modules web-container:ejb-container**
Command disable-monitoring executed successfully

**Exit Status**

0
subcommand executed successfully

1
error in executing the subcommand

**See Also**

**enable-monitoring(1), monitor(1), list(1), get(1), set(1)**

**monitoring(5ASC)**

**asadmin(1M)**

"Administering the Monitoring Service" in *Oracle GlassFish Server Administration Guide*
disable-secure-admin

disables secure admin if it is already enabled.

Synopsis

disable-secure-admin [--help]

Description

The disable-secure-admin subcommand disables secure admin if it is already enabled.

Note: You must restart any running servers in the domain after you enable or disable secure admin. It is simpler to enable or disable secure admin with only the DAS running, then restart the DAS, and then start any other instances.

Options

--help
-?
Displays the help text for the subcommand.

Examples

Example 1 Disabling secure admin for a domain
The following example shows how to disable secure admin for a domain.

asadmin> disable-secure-admin
server-config
default-config
c

Command disable-secure-admin executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

enable-secure-admin(1)
asadmin(1M)
disable-secure-admin-internal-user

Instructs the GlassFish Server DAS and instances to not use the specified admin user to authenticate with each other and to authorize admin operations.

Synopsis

disable-secure-admin-internal-user
   [--help]
   admin-username

Description

The `disable-secure-admin-internal-user` subcommand disables secure admin from using the username (instead of SSL certificates) to authenticate the DAS and instances with each other and to authorize admin operations.

Options

--help
-?
 Displays the help text for the subcommand.

Operands

`admin-username`

The admin user name that GlassFish Server should not use to authenticate the DAS and instances with each other and to authorize admin operations.

Examples

Example 1 Disabling a user name for secure admin

The following example disables secure admin from using username tester to authenticate the DAS and instances with each other and to authorize admin operations.

asadmin> disable-secure-admin-internal-user tester

Command disable-secure-admin-internal-user executed successfully.

Exit Status

0
subcommand executed successfully
1
error in executing the subcommand

See Also

enable-secure-admin(1)
enable-secure-admin-internal-user(1)
asadmin(1M)
disable-secure-admin-principal

disables the certificate for authorizing access in secure administration.

Synopsis

disable-secure-admin-principal
[--help]
--alias aliasname | DN

Description

The disable-secure-admin-principal subcommand disables the certificate as being valid for authorizing access as part of secure administration.

You must specify either the --alias option, or the DN.

Options

--help
-? Displays the help text for the subcommand.

--alias

The alias name of the certificate in the truststore. GlassFish Server looks up the certificate in the truststore using that alias and, if found, disables the corresponding DN as being valid for secure administration. Because alias-name must be an alias associated with a certificate currently in the truststore, you may find it most useful for self-signed certificates.

Operands

DN

The distinguished name of the certificate, specified as a comma-separated list in quotes. For example, "CN=system.amer.oracle.com,OU=GlassFish,O=Oracle Corporation,L=Santa Clara,ST=California,C=US".

Examples

Example 1 Disables trust of a DN for secure administration

The following example shows how to disable trust of a DN for authorizing access in secure administration.

asadmin> disable-secure-admin-principal
*CN=system.amer.oracle.com,OU=GlassFish,
O=Oracle Corporation,L=Santa Clara,ST=California,C=US"

Command disable-secure-admin-principal executed successfully.
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

enable-secure-admin(1)
enable-secure-admin-principal(1)
asadmin(1M)
enable

enables the component

Synopsis

enable [--help] [--target target_name] component_name

Description

The `enable` subcommand enables the specified deployed component. If the component is already enabled, then it is re-enabled. If it has not been deployed, then an error message is returned.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target on which you are enabling the component. Valid values are:

   server
   Enables the default server instance server and is the default value.

   domain_name
   Enables the named domain.

   cluster_name
   Enables every server instance in the cluster.

   instance_name
   Enables a particular clustered or stand-alone server instance.

Operands

component_name
name of the component to be enabled.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter or number. It can contain alphanumeric characters plus underscore (_), dash (-), and period (.) characters. For more information about module and application versions, see “Module and Application Versions” in Oracle GlassFish Server Application Deployment Guide.

At most one version of a module or application can be enabled on a server instance. All other versions are disabled. Enabling one version automatically disables all others.
Examples

Example 1  Enabling a Component
This example enables the disabled component, `sampleApp`.

```
asadmin> enable sampleApp
Command enable executed successfully
```

Exit Status

```
0
subcommand executed successfully
```

```
1
error in executing the subcommand
```

See Also

`deploy(1), undeploy(1), disable(1)`

`asadmin(1M)`

`Oracle GlassFish Server Application Deployment Guide`
enable-backup-config

enables automatic backups defined by a domain backup configuration

Synopsis

enable-backup-config [--help]
backup-config-name

Description

The `enable-backup-config` subcommand sets the `autobackupenabled` option of a domain backup configuration to `true`, thus enabling automatic backups for the domain backup configuration.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

backup-config-name
Specifies the name of the domain backup configuration to enable.

Examples

Example 1 Enabling Automatic Backups

This example enables automatic backups for the `weekly-backup` domain backup configuration. The example shows `list-backup-configs` commands before and after the `disable-backup-config` command to demonstrate that the `autobackupenabled` value changes from `false` to `true`.

asadmin> list-backup-configs --long weekly-backup
Name of Backup Config :weekly-backup
Auto Backup Enabled   :false
Schedule              :weekly
Recycle Limit         :5
Config Only backup    :true
Active Backup Enabled :false
Backup Directory      :null
Last Backup Attempt   :
Last Successful Backup :
Schedule Details:
NAME    SECOND  MINUTE  HOUR  DAY OF WEEK  DAY OF MONTH  MONTH  YEAR
weekly  0       0       0     Sun          *             *      *

asadmin> enable-backup-config weekly-backup
Command enable-backup-config executed successfully.

asadmin> list-backup-configs --long weekly-backup
Name of Backup Config : weekly-backup
Auto Backup Enabled : true
Schedule : weekly
Recycle Limit : 5
Config Only backup : true
Active Backup Enabled : false
Backup Directory : null
Last Backup Attempt :
Last Successful Backup :

Schedule Details:
NAME SECOND MINUTE HOUR DAY OF WEEK DAY OF MONTH MONTH YEAR
weekly 0 0 0 Sun * * *

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-backup-config(1), delete-backup-config(1), disable-backup-config(1), list-backup-configs(1)
asadmin(1M)
enable-http-lb-application

enables a previously-disabled application managed by a load balancer

Synopsis

enable-http-lb-application [--help] --name application_name target

Description

The enable-http-lb-application subcommand enables load balancing for applications deployed on a standalone instance or cluster. You can enable load balancing for an application on all instances in a cluster, or on a single standalone server instance. By default, load balancing is enabled for applications.

Note: This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

Options

--help
-?
Displays the help text for the subcommand.

--name
The name of the application to be enabled.

Operands

target
This operand specifies on which server instance or cluster to enable the application. Valid values are:

- cluster_name- The name of a target cluster.
- instance_name- The name of a target server instance.

Examples

Example 1 Enabling Load Balancing for an Application
This example enables an application named webapps-simple to use load balancing on a cluster named mycluster.

asadmin> enable-http-lb-application --name webapps-simple mycluster
Command enable-http-lb-application executed successfully.
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

disable-http-lb-application(1)
asadmin(1M)
enable-http-lb-server

enables a previously disabled sever or cluster managed by a load balancer

Synopsis

enable-http-lb-server [--help] target

Description

The enable-http-lb-server subcommand enables a standalone server instance or cluster for load balancing. By default, load balancing is enabled for instances and clusters.

Note: This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

Options

--help
-?  Displays the help text for the subcommand.

Operands

target

This operand specifies which server instances and clusters to enable. Valid values are:

- cluster_name- The name of a target cluster.
- instance_name- The name of a target server instance.

Examples

Example 1 Enabling a Cluster for Load Balancing

This example enables load balancing for a cluster named mycluster.

asadmin> enable-http-lb-server mycluster

Command enable-http-lb-server executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand
See Also

create-http-lb-ref(1), disable-http-lb-server(1)
asadmin(1M)
enable-monitoring

enables monitoring for the server or for specific monitorable modules

Synopsis

enable-monitoring [--help]
[-mbean={false|true}]
[-dtrace={true|false}]
[-modules modules=[level][:module=[level]]]*
[-pid pid]
[-options options={true|false}]

Description

The enable-monitoring subcommand is used to turn on monitoring for GlassFish Server or for particular modules during runtime. Changes are dynamic, that is, server restart is not required.

By default, the monitoring service is enabled, that is, the monitoring-enabled attribute of the monitoring-service element is true. However, the default monitoring level for individual modules is OFF. This subcommand used with the --modules option can enable monitoring for a given module by setting the monitoring level to HIGH or LOW. If level is not specified when running the subcommand, the level defaults to HIGH.

The specific meanings of HIGH or LOW are determined by the individual containers. For a list of monitorable modules, see the --modules option in this help page.

An alternative method for enabling monitoring is to use the set subcommand. In this case, the server must be restarted for changes to take effect.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--mbean
Enables Mbean monitoring. Default value is false.

--dtrace
Only usable if the DTrace Monitoring module is present. Enables Oracle Solaris DTrace monitoring. Default value is false.

--modules
Enables specified module or modules by indicating monitoring level. Valid levels are OFF, HIGH, LOW. If level is not specified, the default setting is HIGH. Multiple modules are separated by : (colon). Monitorable modules include connector-connection-pool, connector-service, ejb-container, http-service, jdbc-connection-pool, jersey, jpa, jms-service, jvm, security, thread-pool, transaction-service, web-container, and web-services-container. Additional modules can be listed by using the get subcommand.
**--pid**
Specifies the GlassFish Server JVM process identifier (PID). When monitoring is enabled, the `btrace-agent` is attached, based on the specified PID. Need to specify only in exceptional cases when the system cannot determine the PID. In this situation, the subcommand prompts for the PID of the corresponding GlassFish Server process.

**--options**
Sets the following `btrace-agent` options:

- **debug**
  Enables debugging for BTrace. Default value is false.

**Examples**

**Example 1  Enabling the Monitoring Service for GlassFish Server**
This example enables monitoring for GlassFish Server in general by setting the `enable-monitoring` flag to `true` (default is `true`).

```
asadmin> enable-monitoring
Command enable-monitoring executed successfully
```

**Example 2  Enabling Monitoring for the Web and EJB Containers**
This example enables monitoring for specific containers by setting their monitoring levels.

```
asadmin> enable-monitoring --modules web-container=LOW:ejb-container=HIGH
Command enable-monitoring executed successfully
```

**Example 3  Turning on Debugging for Monitoring**
This example turns on debugging.

```
asadmin> enable-monitoring --options debug=true
Command enable-monitoring executed successfully
```

**Exit Status**

- **0**
  subcommand executed successfully

- **1**
  error in executing the subcommand

**See Also**

- `disable-monitoring(1)`, `monitor(1)`, `list(1)`, `get(1)`, `set(1)`
- `monitoring(5ASC)`
- `asadmin(1M)`

/Administering the Monitoring Service” in *Oracle GlassFish Server Administration Guide*
enable-secure-admin

enables secure admin (if it is not already enabled), optionally changing the alias used for DAS-to-instance admin messages or the alias used for instance-to-DAS admin messages.

Synopsis

enable-secure-admin [--help]
[--adminalias=alias]
[--instancealias=alias]

Description

The enable-secure-admin subcommand causes the DAS and the instances in the domain to use SSL certificates for encrypting the messages they send to each other. This subcommand also allows the DAS to accept administration messages from remote admin clients such as the asadmin utility and IDEs.

Note: You must restart any running servers in the domain after you enable or disable secure admin. It is simpler to enable or disable secure admin with only the DAS running, then restart the DAS, and then start any other instances.

By default, when secure admin is enabled the DAS and the instances use these SSL certificates to authenticate to each other as security "principals" and to authorize admin access. The --asadminalias value indicates to the DAS which SSL certificate it should use to identify itself to the instances. The --instancealias value determines for instances which SSL certificate they should use to identify themselves to the DAS.

The enable-secure-admin subcommand fails if any administrative user in the domain has a blank password.

Alternatively, you can use the enable-secure-admin-internal-user subcommand to cause the servers to identify themselves using a secure admin user name and password.

Options

--help
-?
Displays the help text for the subcommand.

--adminalias

The alias that refers to the SSL/TLS certificate on the DAS. This alias is used by the DAS to identify itself to instances. The default value is s1as.

--instancealias

The alias that refers to the SSL/TLS certificate on the instances. This alias is used by the instances to identify themselves to the DAS. The default value is glassfish-instance.
Examples

Example 1 Enabling secure admin for a domain
The following example shows how to enable secure admin for a domain using an admin alias adtest and an instance alias intest

asadmin> enable-secure-admin --adminalias adtest --instancealias intest
server-config
default-config

Command enable-secure-admin executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

disable-secure-admin(1)
enable-secure-admin-principal(1)
enable-secure-admin-internal-user(1)
asadmin(1M)
enable-secure-admin-internal-user

Instructs the GlassFish Server DAS and instances to use the specified admin user and the password associated with the password alias to authenticate with each other and to authorize admin operations.

Synopsis

```
enable-secure-admin-internal-user
[--help]
[--passwordalias pwdaliasname]
admin-username
```

Description

The `enable-secure-admin-internal-user` subcommand instructs all servers in the domain to authenticate to each other, and to authorize admin operations submitted to each other, using an existing admin username and password rather than SSL certificates. This generally means that you must:

1. Create a valid admin user.
   
   ```
   asadmin> create-file-user --authrealmname admin-realm --groups
   asadmin newAdminUsername
   ```

2. Create a password alias for the just-created password.
   
   ```
   asadmin> create-password-alias passwordAliasName
   ```

3. Use that user name and password for inter-process authentication and admin authorization.
   
   ```
   asadmin> enable-secure-admin-internal-user
   --passwordalias passwordAliasName
   newAdminUsername
   ```

If GlassFish Server finds at least one secure admin internal user, then if secure admin is enabled GlassFish Server processes will not use SSL authentication and authorization with each other and will instead use username password pairs.

If secure admin is enabled, all GlassFish Server processes continue to use SSL encryption to secure the content of the admin messages, regardless of how they authenticate to each other.

Most users who use this subcommand will need to set up only one secure admin internal user. As a general practice, you should not use the same user name and password pair for internal admin communication and for admin user login.

If you set up more than one secure admin internal user, you should not make any assumptions about which user name and password pair GlassFish Server will choose to use for any given admin request.
Options

--help
-?
Displays the help text for the subcommand.

--passwordalias
The password alias for the user that GlassFish Server should use for internally
authenticating and authorizing the DAS to instances and the instances to the DAS.

Operands

admin-username
The admin user name that GlassFish Server should use for internally authenticating
and authorizing the DAS to instances and the instances to the DAS.

Examples

Example 1  Specifying a user name and password for secure admin
The following example allows secure admin to use a user name and password alias for
authentication and authorization between the DAS and instances, instead of
certificates.

asadmin> enable-secure-admin-internal-user
--passwordalias passwordAliasName
newAdminUsername

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

enable-secure-admin(1)
disable-secure-admin-internal-user(1)
asadmin(1M)
enable-secure-admin-principal

Instructs GlassFish Server, when secure admin is enabled, to accept admin requests from clients identified by the specified SSL certificate.

Synopsis

enable-secure-admin-principal
[--help]
--alias aliasname | DN

Description

The enable-secure-admin-principal subcommand instructs GlassFish Server to accept admin requests when accompanied by an SSL certificate with the specified distinguished name (DN). If you use the "--alias aliasname" form, then GlassFish Server looks in its truststore for a certificate with the specified alias and uses the DN associated with that certificate. Otherwise, GlassFish Server records the value you specify as the DN.

You must specify either the --alias option, or the DN.

You can run enable-secure-admin-principal multiple times so that GlassFish Server accepts admin requests from a client sending a certificate with any of the DNs you specify.

When you run enable-secure-admin, GlassFish Server automatically records the DNs for the admin alias and the instance alias, whether you specify those values or use the defaults. You do not need to run enable-secure-admin-principal yourself for those certificates. Other than these certificates, you must run enable-secure-admin-principal for any other DN that GlassFish Server should authorize to send admin requests. This includes DNs corresponding to trusted certificates (those with a certificate chain to a trusted authority.)

Options

--help
-?
Displays the help text for the subcommand.

--alias
The alias name of the certificate in the trust store. GlassFish Server looks up certificate in the trust store using that alias and, if found, stores the corresponding DN as being valid for secure administration. Because alias-name must be an alias associated with a certificate currently in the trust store, you may find it most useful for self-signed certificates.

Operands

DN
The distinguished name of the certificate, specified as a comma-separated list in quotes. For example, "CN=system.amer.oracle.com,OU=GlassFish,0=Oracle Corporation,L=Santa Clara,ST=California,C=US".
Examples

Example 1  Trusting a DN for secure administration
The following example shows how to specify a DN for authorizing access in secure administration.

asadmin> enable-secure-admin-principal
"CN=system.amer.oracle.com,OU=GlassFish,
O=Oracle Corporation,L=Santa Clara,ST=California,C=US"

Command enable-secure-admin-principal executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

enable-secure-admin(1)
disable-secure-admin-principal(1)
asadmin(1M)
export

marks a variable name for automatic export to the environment of subsequent commands in multimode

Synopsis

export [--help] [variable-name=value [variable-name=value] *]

Description

In multimode, the export subcommand marks an environment variable for automatic export to the environment of subsequent commands. All subsequent commands use the variable name value as specified unless you exit multimode, or use the unset subcommand to unset the variable. If only the variable name is specified, the current value of that variable name is displayed.

If the export subcommand is used without any arguments, a list of all the exported variables and their values is displayed. Exported shell environment variables set prior to invoking the asadmin utility are imported automatically and set as exported variables within asadmin. Environment variables that are not exported cannot be read by the asadmin utility.

This subcommand is supported in local mode only.

Options

--help
-
Displays the help text for the subcommand.

Operands

variable-name=value

Variable name and value for automatic export to the environment to be used by subsequent commands.

Examples

**Example 1 Listing the Environment Variables That Are Set**

This example lists the environment variables that have been set.

```bash
asadmin> export
AS_ADMIN_USER = admin
AS_ADMIN_HOST = bluestar
AS_ADMIN_PREFIX = server1.jms-service
AS_ADMIN_PORT = 8000
Command export executed successfully
```

**Example 2 Setting an Environment Variable**

This example sets the AS_ADMIN_HOST environment variable to bluestar.
asadmin> export AS_ADMIN_HOST=bluestar
Command export executed successfully

Example 3 Setting Multiple Environment Variables
This example sets a number of environment variables for the multimode environment.

asadmin> export AS_ADMIN_HOST=bluestar AS_ADMIN_PORT=8000
AS_ADMIN_USER=admin AS_ADMIN_PREFIX=server1.jms-service
Command export executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

unset(1), multimode(1)
asadmin(1M)
**export-http-lb-config**

exports the load balancer configuration or load balancer to a file

**Synopsis**

```bash
export-http-lb-config [-help] --config config_name | --lbname load_balancer_name
|--target target| [--retrieve=false] [file_name]
```

**Description**

The `export-http-lb-config` subcommand exports a load balancer configuration or load balancer into a file that the load balancer plug-in can use. The default file name is `loadbalancer.xml`, but you can specify a different name. Once exported, you manually copy the exported file to the load balancer plug-in location before configuration changes are applied. The `--target` option makes it possible to generate a `loadbalancer.xml` for clusters or standalone instances without having to manually create `lb-config` or `load-balancer` elements in the target's `domain.xml`.

To apply changes to the load balancer without manually copying the configuration file, configure the load balancer to automatically apply changes with `create-http-lb`. If you use the `create-http-lb` subcommand, you do not need to use `export-http-lb-config`.

**Note:** This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

**Options**

- **--help**
  - Displays the help text for the subcommand.

- **--config**
  - Specifies which load balancer configuration to export.
  
  Specify either a load balancer configuration or a load balancer. Specifying both results in an error.

- **--lbname**
  - Specifies the load balancer to export.

  Specify either a load balancer configuration or a load balancer. Specifying both results in an error.

- **--retrieve**
  - If set to `true`, retrieves the `loadbalancer.xml` file from the remote machine. The default is `false`.
--target
Specifies the target to which the load balancer configuration will be exported. If a target is not specified, the load balancer configuration is exported to the location specified with file_name.

Operands

file_name
Specifies the file name and location of the exported configuration.

- If you specify a directory (relative or absolute) but not a file name, the file named loadbalancer.xml.load_balancer_config_name is created in the specified directory. On Microsoft Windows systems the path must be in quotes.
- If you specify a file name in a relative or absolute path, the file is created with the name you specify in the directory you specify.
- If you specify a file name but do not specify a directory, the file is created with that name in the current working directory.
- If you do not specify this operand, the default value is a file named loadbalancer.xml.load_balancer_config_name created in the domain-dir/generated directory.

target
Specifies the target to which the configuration will be exported.

Valid values are:

- cluster_name- Specifies a cluster and its server instances.
- stand-alone_instance_name- Specifies a specific server instance.

Examples

Example 1 Exporting a Load Balancer Configuration on UNIX
The following example exports a load balancing configuration named mycluster-http-lb-config to a file named loadbalancer.xml in the /Sun/AppServer directory.

asadmin> export-http-lb-config --config mycluster-http-lb-config /Sun/AppServer/loadbalancer.xml
Command export-http-lb-config executed successfully.

Example 2 Exporting a Load Balancer Configuration on Windows
The following example exports a load balancing configuration named mycluster-http-lb-config to a file named loadbalancer.xml in the C:\Sun\AppServer directory on a Microsoft Windows system.

asadmin> export-http-lb-config --config mycluster-http-lb-config "C:\Sun\AppServer\loadbalancer.xml"
Command export-http-lb-config executed successfully.

Exit Status

0
subcommand executed successfully
error in executing the subcommand

See Also

create-http-lb(1), create-http-lb-config(1), list-http-lb-configs(1)

asadmin(1M)
export-sync-bundle

exports the configuration data of a cluster or standalone instance to an archive file

Synopsis

export-sync-bundle [--help]
--target target
[--retrieve={false|true}]
[file-name]

Description

The export-sync-bundle subcommand exports the configuration data of a cluster or standalone instance to an archive file. The archive file can then be used with the import-sync-bundle(1) subcommand to restore the configuration data.

Importing an instance's configuration data transfers the data to a host for an instance without the need for the instance to be able to communicate with the domain administration server (DAS). Importing an instance's configuration data is typically required for the following reasons:

■ To reestablish the instance after an upgrade
■ To synchronize the instance manually with the DAS when the instance cannot contact the DAS

The subcommand creates an archive that contains the following files and directories in the current domain directory:

■ All the files in the following directories:
  – config
  – docroot

■ The entire contents of the following directories and their subdirectories:
  – applications
  – config/target, where target is the cluster or standalone instance for which configuration data is being exported
  – generated
  – lib

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
The cluster or standalone instance for which to export configuration data. The --target option is required.
This option must not specify a clustered GlassFish Server instance. If this option specifies a clustered instance, an error occurs. To export configuration data for a clustered instance, specify the name of the cluster of which the instance is a member, not the instance.

--retrieve
Specifies whether the archive file is downloaded from the DAS host to the host where the subcommand is run.

Possible values are as follows:

true
The archive file is downloaded to the host where the subcommand is run.

false
The archive file is not downloaded and remains on the DAS host (default).

Operands

file-name
The file name and location of the archive file to which to export the data.

The default depends on the setting of the --retrieve option:

- If --retrieve is false, the default is sync/target-sync-bundle.zip in the current domain directory.
- If --retrieve is true, the default is target-sync-bundle.zip in the current working directory.

target is the cluster or standalone instance that the --target option specifies.

If a relative path is specified, the directory to which the path is appended depends on the setting of the --retrieve option:

- If --retrieve is false, the path is appended to the config subdirectory of the current domain directory.
- If --retrieve is true, the path is appended to the current working directory.

If an existing directory is specified without a filename, the file name of the archive file is target-sync-bundle.zip, where target is the cluster or standalone instance that the --target option specifies.

Examples

Example 1 Exporting the Configuration Data of a Cluster
This example exports the configuration data of the cluster pmdcluster.

asadmin> export-sync-bundle --target=pmdcluster

Command export-sync-bundle executed successfully.

Exit Status

0
command executed successfully
1
error in executing the command

See Also

import-sync-bundle(1)
asadmin(1M)
flush-connection-pool

reinitializes all connections established in the specified connection pool

Synopsis

flush-connection-pool [--help] pool_name
[--appname application [--modulename module]]

Description

The flush-connection-pool subcommand resets a JDBC connection pool or a
connector connection pool to its initial state. Any existing live connections are
destroyed, which means that the transactions associated with these connections are
lost. The subcommand then recreates the initial connections for the pool, and restores
the pool to its steady pool size.

This subcommand is supported in remote mode only.

Application Scoped Resources

The flush-connection-pool subcommand can target resources that are scoped to a
specific application or module, as defined in the glassfish-resources.xml for the
GlassFish domain.

- To reference the jndi-name for an application scoped resource, perform the lookup
  using the java:app prefix.
- To reference the jndi-name for a module scoped resource, perform the lookup
  using the java:module prefix.

The jndi-name for application-scoped-resources or module-scoped-resources are specified
using the format java:app/jdbc/myDataSource or
java:module/jdbc/myModuleLevelDataSource. This naming scope is defined in the Java
EE 6 Specification (http://download.oracle.com/javaee/6/api/).

Options

--help
-?
Displays the help text for the subcommand.

--appname
Name of the application in which the application scoped resource is defined.

--modulename
Name of the module in which the module scoped resource is defined.

Operands

pool_name
Name of the connection pool to be reinitialized.
Flush a Connection Pool

This example reinitializes the JDBC connection pool named __TimerPool.

```
asadmin> flush-connection-pool __TimerPool
Command flush-connection-pool executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

list-connector-connection-pools(1), list-jdbc-connection-pools(1)

asadmin(1M)
flush-jmsdest

purges messages in a JMS destination.

Synopsis

flush-jmsdest [--help]  
--desttype {topic|queue}  
[--target target]  
destname

Description

The flush-jmsdest subcommand purges the messages from a physical destination in the server's Java Message Service (JMS) configuration.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

--help
-?
Displays the help text for the subcommand.

--desttype
This option indicates the type of physical destination from which you want to purge messages. The supported destination types are topic and queue.

--target
Purges messages from the physical destination only for the specified target. Valid values are as follows:

server
Purges messages from the physical destination for the default server instance. This is the default value.

configuration-name
Purges messages from the physical destination in the specified configuration.

cluster-name
Purges messages from the physical destination for every server instance in the specified cluster.

instance-name
Purges messages from the physical destination for the specified server instance.

Operands

dest_name
The unique identifier of the JMS destination to be purged.
Examples

Example 1 Purging messages from a physical destination
The following subcommand purges messages from the queue named PhysicalQueue.

asadmin> flush-jmsdest --desttype queue PhysicalQueue
Command flush-jmsdest executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jmsdest(1), list-jmsdest(1), create-jmsdest(1)
asadmin(1M)
freeze-transaction-service

freezes the transaction subsystem

Synopsis

freeze-transaction-service [--help] [--target target]

Description

The freeze-transaction-service subcommand freezes the transaction subsystem, preventing the transaction manager from starting, completing, or changing the state of all in-flight transactions. Invoke this command before rolling back any in-flight transactions. Invoking this subcommand on an already frozen transaction subsystem has no effect. Restarting the server unfreezes the transaction subsystem. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target on which you are freezing the transaction service. Valid values are:

  server
  Freezes the transaction service for the default server instance server and is the default value.

  configuration_name
  Freezes the transaction service for all server instances that use the named configuration.

  cluster_name
  Freezes the transaction service for every server instance in the cluster.

  instance_name
  Freezes the transaction service for a particular server instance.

Examples

Example 1 Using freeze-transaction-service

% asadmin freeze-transaction-service
Command freeze-transaction-service executed successfully

Exit Status

0
command executed successfully
1
error in executing the command

See Also

unfreeze-transaction-service(1), rollback-transaction(1),
recover-transactions(1)
asadmin(1M)

"Administering Transactions" in Oracle GlassFish Server Administration Guide

"Transactions" in The Java EE 6 Tutorial
generate-jvm-report

shows the JVM machine statistics for a given target instance

Synopsis

generate-jvm-report [--help] [--type=jvm-statistic-type] [--target target]

Description

The generate-jvm-report subcommand creates a report that shows the threads (dump of stack trace), classes, memory, or loggers for a given target instance, including the domain administration server (DAS). If a type is not specified, a summary report is generated. This subcommand only provides statistics for the GlassFish Server instance processes. This subcommand provides an alternative to sending Ctrl+Break or kill -3 signals to GlassFish Server processes to obtain a stack trace for processes that are hanging.

The information in the report is obtained from managed beans (MBeans) and MXBeans that are provided in the Java Platform, Standard Edition (Java SE) or JDK software with which GlassFish Server is being used.

If GlassFish Server is running in the Java Runtime Environment (JRE) software from JDK release 6 or Java SE 6, additional information is provided. For example:

- System load on the available processors
- Object monitors that are currently held or requested by a thread
- Lock objects that a thread is holding, for example, ReentrantLock objects and ReentrantReadWriteLock objects

If the JRE software cannot provide this information, the report contains the text NOT AVAILABLE.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target for which you are showing JVM machine statistics.

Valid values are as follows:

  server
  Specifies the DAS (default).

  instance-name
  Specifies a GlassFish Server instance.

  cluster-name
  Specifies a cluster.
**configuration-name**
Specifies a named configuration.

**--type**
The type of report that is to be generated. Default is *summary*.

**summary**
Displays summary information about the threads, classes, and memory (default).

**memory**
Provides information about heap and non-heap memory consumption, memory pools, and garbage collection statistics for a given target instance.

**class**
Provides information about the class loader for a given target instance.

**thread**
Provides information about threads running and the thread dump (stack trace) for a given target instance.

**log**
Provides information about the loggers that are registered in the Virtual Machine for the Java platform (Java Virtual Machine or JVM machine).1

**Examples**

**Example 1 Obtaining Summary Information for the JVM Machine**
This example shows a partial listing of a report that is generated if no type is specified. This same report is generated if the *summary* type is specified.

```
asadmin> generate-jvm-report
Operating System Information:
  Name of the Operating System: SunOS
  Binary Architecture name of the Operating System: sparc, Version: 5.10
  Number of processors available on the Operating System: 32
  System load on the available processors for the last minute: 7.921875.
(Sum of running and queued runnable entities per minute)
General Java Runtime Environment Information for the VM: 64097@sr1-usca-22
... 
  sun.desktop = gnome
  sun.io.unicode.encoding = UnicodeBig
  sun.java.launcher = SUN_STANDARD
  sun.jnu.encoding = ISO646-US
  sun.management.compiler = HotSpot Client Compiler
  sun.os.patch.level = unknown
  user.dir = /home/thisuser/GlassFish/glassfishv3/glassfish/domains/mydomain4/config
  user.home = /home/thisuser
  user.language = en
  user.name = thisuser
  user.timezone = US/Pacific
Command generate-jvm-report executed successfully
```

**Example 2 Obtaining Information for a Particular JVM Machine Type**
This example generates a report that shows information on the class loader.

1 The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform.
asadmin> **generate-jvm-report --type=class**
Class loading and unloading in the Java Virtual Machine:
Number of classes currently loaded in the Java Virtual Machine: 3,781
Number of classes loaded in the Java Virtual Machine since the startup: 3,868
Number of classes unloaded from the Java Virtual Machine: 87
Just-in-time (JIT) compilation information in the Java Virtual Machine:
Java Virtual Machine compilation monitoring allowed: true
Name of the Just-in-time (JIT) compiler: HotSpot Client Compiler
Total time spent in compilation: 0 Hours 0 Minutes 4 Seconds
Command generate-jvm-report executed successfully.

**Exit Status**

0
subcommand executed successfully

1
generate-jvm-report error in executing the subcommand

**See Also**

create-jvm-options(1), delete-jvm-options(1), list-jvm-options(1)
asadmin(1M)
get

gets the values of configurable or monitorable attributes

Synopsis

get [--help] [--monitor=[true|false]]
(dotted-attribute--name)+

Description

The get subcommand uses dotted names to get the names and values of configurable or monitorable attributes for GlassFish Server elements.

You can use the list(1) subcommand to display the dotted names that represent individual server components and subsystems. For example, a dotted name might be server.applications.web-module. Attributes from the monitoring hierarchy are read-only, but configuration attributes can be modified using the set(1) subcommand. For more detailed information on dotted names, see the dotted-names(5ASC) help page.

Note: Characters that have special meaning to the shell or command interpreter, such as * (asterisk), should be quoted or escaped as appropriate to the shell, for example, by enclosing the argument in quotes. In multimode, quotes are needed only for arguments that include spaces, quotes, or backslash.

The following list shows common usage of the get subcommand with the * (asterisk):

- get * Or get *.*
  Gets all values on all dotted name prefixes.

- get domain* Or get domain*.*
  Gets all values on the dotted names that begin with domain.

- get *config*.**
  Gets all values on the dotted names that match *config*.**.

- get domain.j2ee-applications.*.ejb-module.*.**
  Gets all values on all EJB modules of all applications.

- get *web-modules.*.*
  Gets all values on all web modules whether in an application or standalone.

- get *.*.*.*
  Gets all values on all dotted names that have four parts.

Options

--help
-?
Displays the help text for the subcommand.
--monitor or -m
Defaults to false. If set to false, the configurable attribute values are returned. If set to true, the monitorable attribute values are returned.

Operands

dotted-attribute-name
Identifies the attribute name in the dotted notation. At least one dotted name attribute is required. The dotted notation is the syntax used to access attributes of configurable entities.

Examples

Example 1 Getting the Attributes of a Configurable Element
This example gets the attributes of listener.http-listener-1.

server.http-service.http-listener.http-listener-1.address = 0.0.0.0
Command get executed successfully.

Example 2 Getting Monitorable Objects
This example gets the configuration attributes for setting the monitoring level and shows whether they are enabled (LOW or HIGH) or disabled (OFF). The jvm component is enabled for monitoring.

asadmin> get server.monitoring-service.module-monitoring-levels.*
server.monitoring-service.module-monitoring-levels.connector-connection-pool=OFF
server.monitoring-service.module-monitoring-levels.connector-service=OFF
server.monitoring-service.module-monitoring-levels.d-trace=OFF
server.monitoring-service.module-monitoring-levels.ejb-container=OFF
server.monitoring-service.module-monitoring-levels.http-service=OFF
server.monitoring-service.module-monitoring-levels.jdbc-connection-pool=OFF
server.monitoring-service.module-monitoring-levels.jms-service=OFF
server.monitoring-service.module-monitoring-levels.jvm=HIGH
server.monitoring-service.module-monitoring-levels.orb=OFF
server.monitoring-service.module-monitoring-levels.thread-pool=OFF
server.monitoring-service.module-monitoring-levels.transaction-service=OFF
server.monitoring-service.module-monitoring-levels.web-container=OFF
Command get executed successfully.

Example 3 Getting Attributes and Values for a Monitorable Object
This example gets all attributes and values of the jvm monitorable object.

asadmin> get --monitor server.jvm.*
server.jvm.HeapSize_Current = 45490176
server.jvm.HeapSize_Description = Describes JvmHeapSize
server.jvm.HeapSize_HighWaterMark = 45490176
server.jvm.HeapSize_LastSampleTime = 1063217002433
server.jvm.HeapSize_LowWaterMark = 0
server.jvm.HeapSize_LowerBound = 0
server.jvm.HeapSize_Name = JvmHeapSize
server.jvm.HeapSize_StartTime = 1063238840055
server.jvm.HeapSize_Unit = bytes
server.jvm.HeapSize_UpperBound = 531628032
server.jvm.UpTime_Count = 1063238840100
server.jvm.UpTime_Description = Describes JvmUpTime
server.jvm.UpTime_LastSampleTime = 1-63238840070
server.jvm.UpTime_Name = JvmUpTime
server.jvm.UpTime_StartTime = 1063217002430
server.jvm.UpTime_Unit = milliseconds
Command get executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

list(1), set(1)
dotted-names(5ASC)
asadmin(1M)

Oracle GlassFish Server Administration Guide
get-client-stubs

retrieves the application JAR files needed to launch the application client.

Synopsis

get-client-stubs [--help] --appname application_name local_directory_path

Description

The get-client-stubs subcommand copies the required JAR files for an AppClient standalone module or each AppClient module in an application from the server machine to the local directory. Each client's generated JAR file is retrieved, along with any required supporting JAR files. The client JAR file name is of the form app-nameClient.jar. Before executing the get-client-stubs subcommand, you must deploy the application or module. The generated client JAR file is useful for running the application using the appclient utility. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--appname
The name of the application or stand-alone client module.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter or number. It can contain alphanumeric characters plus underscore (_), dash (-), and period (.) characters. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.

Operands

local_directory_path
The path to the local directory where the client stub JAR file should be stored.

Examples

Example 1 Using get-client-stubs

asadmin> get-client-stubs --appname myapplication /sample/example
Command get-client-stubs executed successfully

Exit Status

0
command executed successfully
1
error in executing the command

See Also

deploy(1), redeploy(1), undeploy(1)
applclient(1M), asadmin(1M), package-applclient(1M)

Oracle GlassFish Server Application Deployment Guide
get-health

provides information on the cluster health

Synopsis

get-health [--help] cluster_name

Description

The get-health subcommand gets information about the health of the cluster. Note that if the group management service (GMS) is not enabled, the basic information about whether the server instances in this cluster are running or not running is not returned. For each server instance, one of the following states is reported: not started, started, stopped, rejoined, or failed. This subcommand is available in remote mode only.

Options

--help

Displays the help text for the subcommand.

Operands

cluster_name

The name of the cluster for which you want the health information. This subcommand prompts you for the cluster name if you don't specify it.

Examples

Example 1  Checking the health of server instances in a cluster

asadmin> get-health cluster1
instance1 started since Wed Sep 29 16:32:46 EDT 2010
instance2 started since Wed Sep 29 16:32:45 EDT 2010
Command get-health executed successfully.

Exit Status

0

command executed successfully

1

error in executing the command

See Also

validate-multicast(1)
asadmin(1M)
import-sync-bundle

imports the configuration data of a clustered instance or standalone instance from an archive file

Synopsis

import-sync-bundle [--help]
--instance instance-name
[--nodedir node-dir] [--node node-name]
file-name

Description

The import-sync-bundle subcommand imports the configuration data of a clustered instance or standalone instance from an archive file that was created by the export-sync-bundle(1) subcommand.

You must run this subcommand on the host where the instance resides. To contact the domain administration server (DAS), this subcommand requires the name of the host where the DAS is running. If a nondefault port is used for administration, this subcommand also requires the port number. You must provide this information through the --host option and the --port option of the asadmin(1M) utility.

Importing an instance’s configuration data transfers the data to a host for an instance without the need for the instance to be able to communicate with the DAS. Importing an instance’s configuration data is typically required for the following reasons:

■ To reestablish the instance after an upgrade
■ To synchronize the instance manually with the domain administration server (DAS) when the instance cannot contact the DAS

The subcommand imports an instance’s configuration data by performing the following operations:

■ Creating or updating the instance's files and directories
■ Attempting to register the instance with the DAS

If the attempt to register the instance with the DAS fails, the subcommand does not fail. Instead, the subcommand displays a warning that the attempt failed. The warning contains the command to run to register the instance with the DAS.

The import-sync-bundle subcommand does not contact the DAS to determine the node on which the instance resides. If the node is not specified as an option of the subcommand, the subcommand determines the node from the DAS configuration in the archive file.

This subcommand is supported in local mode only.

Options

--help
-?

Displays the help text for the subcommand.
import-sync-bundle

--instance
The instance for which configuration data is being imported. The instance must already exist in the DAS configuration. The archive file from which the data is being imported must contain data for the specified instance.

--nodedir
The directory that contains the instance’s node directory. The instance’s files are stored in the instance’s node directory. The default is `as-install/nodes`.

--node
The node on which the instance resides. If this option is omitted, the subcommand determines the node from the DAS configuration in the archive file.

Operands

file-name
The name of the file, including the path, that contains the archive file to import. This operand is required.

Examples

Example 1 Importing Configuration Data for a Clustered Instance
This example imports the configuration for the clustered instance `ymli2` on the node `sj02` from the archive file `/export/glassfish3/glassfish/domains/domain1/sync/ymlcluster-sync-bundle.zip.

The command is run on the host `sj02`, which is the host that the node `sj02` represents. The DAS is running on the host `sr04` and uses the default HTTP port for administration.

```
  sj02# asadmin --host sr04 import-sync-bundle --node sj02 --instance ymli2
  /export/glassfish3/glassfish/domains/domain1/sync/ymlcluster-sync-bundle.zip

  Command import-sync-bundle executed successfully.
```

Exit Status

0
command executed successfully

1
error in executing the command

See Also

export-sync-bundle(1)

asadmin(1M)
install-node

installs GlassFish Server software on specified SSH-enabled hosts

Synopsis

install-node [--help]
[--archive archive]
[--create={false|true}] [--save={false|true}]
[--installdir as-install-parent]
[--sshport ssh-port] [--sshuser ssh-user]
[--sshkeyfile ssh-keyfile]
[--force={false|true}]
host-list

Description

The install-node subcommand installs GlassFish Server software on the hosts that are specified as the operand of the subcommand. This subcommand requires secure shell (SSH) to be configured on the host where the subcommand is run and on each host where the GlassFish Server software is being installed.

If necessary, the subcommand creates a ZIP archive of the GlassFish Server software from the installation where this subcommand is run. The archive does not contain the domains directory or the nodes directory. These directories are synchronized from the domain administration server (DAS) when instances on nodes that represent the hosts are created and started. The subcommand does not delete the archive after installing the GlassFish Server software from the archive on the specified hosts.

If multiple hosts are specified, the configuration of the following items is the same on all hosts:

- Base installation directory
- SSH port
- SSH user
- SSH key file

If the SSH key file does not exist on a host where the GlassFish Server software is to be installed, the subcommand runs interactively and prompts for a password. To enable the subcommand to run noninteractively, the following conditions must be met:

- The --interactive option of the asadmin(1M) utility must be false.
- The --passwordfile option of the asadmin utility must specify a password file.
- The password file must contain the AS_ADMIN_SSHPASSWORD entry.

The subcommand does not modify the configuration of the DAS.

Note: This subcommand is equivalent to the install-node-ssh(1) subcommand. To install GlassFish Server software on hosts that use the Distributed Component Object Model (DCOM) remote protocol, use the install-node-dcom(1) subcommand.
This subcommand is supported in local mode only.

**Options**

--help
-?
Displays the help text for the subcommand.

--archive
The absolute path to the archive file of the GlassFish Server software that is to be installed. If no archive file is specified, the subcommand creates an archive from the installation of GlassFish Server software from which the subcommand is run. This archive is created in the home directory of the user that is running the command.

--create
Specifies whether the subcommand should create an archive file of the GlassFish Server software to install.

  false
  No archive file is created. The subcommand installs the software from the existing archive file that the --archive option specifies (default).

  true
  The subcommand creates an archive file from the installation of GlassFish Server software from which the subcommand is run.

--save
Specifies whether the archive file from which the software is installed is saved after installation.

  false
  The archive file is not saved. The subcommand deletes the file after installing the software (default).

  true
  The archive file is saved.

--installdir
The absolute path to the parent of the base installation directory where the GlassFish Server software is to be installed on each host, for example, /export/glassfish3/. If the directory does not exist, the subcommand creates the directory.

The user that is running this subcommand must have write access to the specified directory. Otherwise, an error occurs.

To overwrite an existing an installation of the GlassFish Server software, set the --force option to true. If the directory already contains an installation and the --force option is false, an error occurs.

The default is the parent of the base installation directory of the GlassFish Server software on the host where this subcommand is run.

--sshport
The port to use for SSH connections to the host where the GlassFish Server software is to be installed. The default is 22.
--sshuser
The user on the host where the GlassFish Server software is to be installed that is to run the process for connecting through SSH to the host. The default is the user that is running this subcommand. To ensure that the DAS can read this user's SSH private key file, specify the user that is running the DAS process.

--sshkeyfile
The absolute path to the SSH private key file for user that the --sshuser option specifies. This file is used for authentication to the sshd daemon on the host.

The user that is running this subcommand must be able to reach the path to the key file and read the key file.

The default is a key file in the user's .ssh directory. If multiple key files are found, the subcommand uses the following order of preference:

1. id_rsa
2. id_dsa
3. identity

--force
Specifies whether the subcommand overwrites an existing installation of the GlassFish Server software in the directory that the --installdir option specifies. Possible values are as follows:

false
The existing installation is not overwritten (default).

true
The existing installation is overwritten.

Operands

host-list
A space-separated list of the names of the hosts where the GlassFish Server software is to be installed.

Examples

Example 1 Installing GlassFish Server Software at the Default Location
This example installs GlassFish Server software on the hosts sj03.example.com and sj04.example.com at the default location.

asadmin> install-node sj03.example.com sj04.example.com
Created installation zip /home/gfuser/glassfish2339538623689073993.zip
Successfully connected to gfuser@sj03.example.com using keyfile /home/gfuser/.ssh/id_rsa
Copying /home/gfuser/glassfish2339538623689073993.zip (81395008 bytes) to sj03.example.com:/export/glassfish3
Installing glassfish2339538623689073993.zip into sj03.example.com:/export/glassfish3
Removing sj03.example.com:/export/glassfish3/glassfish2339538623689073993.zip
Fixing file permissions of all files under sj03.example.com:/export/glassfish3/bin
Successfully connected to gfuser@sj04.example.com using keyfile /home/gfuser/.ssh/id_rsa
Copying /home/gfuser/glassfish2339538623689073993.zip (81395008 bytes) to sj04.example.com:/export/glassfish3
Installing glassfish2339538623689073993.zip into sj04.example.com:/export/glassfish3
Removing sj04.example.com:/export/glassfish3/glassfish2339538623689073993.zip
Fixing file permissions of all files under sj04.example.com:/export/glassfish3/bin
Command install-node executed successfully

Exit Status

0
command executed successfully
1
error in executing the command

See Also

install-node-dcom(1), install-node-ssh(1), uninstall-node(1),
uninstall-node-dcom(1), uninstall-node-ssh(1)
asadmin(1M)
install-node-dcom

installs GlassFish Server software on specified DCOM-enabled hosts

Synopsis

install-node-dcom [--help]
[--archive archive]
[--create={false|true}] [--save={false|true}]
[--installdir as-install-parent]
[--windowsuser windows-user] [--windowsdomain windows-domain]
[--force={false|true}]
host-list

Description

The install-node-dcom subcommand installs GlassFish Server software on the hosts that are specified as the operand of the subcommand. This subcommand requires the Distributed Component Object Model (DCOM) remote protocol to be configured on the host where the subcommand is run and on each host where the GlassFish Server software is being installed. The DCOM protocol is available only on Windows systems.

If necessary, the subcommand creates a ZIP archive of the GlassFish Server software from the installation where this subcommand is run. The archive does not contain the domains directory or the nodes directory. These directories are synchronized from the domain administration server (DAS) when instances on nodes that represent the hosts are created and started. The subcommand does not delete the archive after installing the GlassFish Server software from the archive on the specified hosts.

If multiple hosts are specified, the configuration of the following items is the same on all hosts:

- Base installation directory
- Windows user
- Windows domain

By default, the subcommand runs interactively and prompts for a password to authenticate the Windows user on each host where the GlassFish Server software is being installed. To enable the subcommand to run noninteractively, the following conditions must be met:

- The --interactive option of the asadmin(1M) utility must be false.
- The --passwordfile option of the asadmin utility must specify a password file.
- The password file must contain the AS_ADMIN_WINDOWSPASSWORD entry.

The subcommand does not modify the configuration of the DAS.

This subcommand is supported in local mode only.

---

Note: To install GlassFish Server software on hosts that use secure shell (SSH), use the install-node-ssh(1) subcommand.
Options

--help
-?
Displays the help text for the subcommand.

--archive
The absolute path to the archive file of the GlassFish Server software that is to be installed. If no archive file is specified, the subcommand creates an archive from the installation of GlassFish Server software from which the subcommand is run. This archive is created in the home directory of the user that is running the command.

--create
Specifies whether the subcommand should create an archive file of the GlassFish Server software to install.

false
No archive file is created. The subcommand installs the software from the existing archive file that the --archive option specifies (default).

true
The subcommand creates an archive file from the installation of GlassFish Server software from which the subcommand is run.

--save
Specifies whether the archive file from which the software is installed is saved after installation.

false
The archive file is not saved. The subcommand deletes the file after installing the software (default).

true
The archive file is saved.

--installdir
The absolute path to the parent of the base installation directory where the GlassFish Server software is to be installed on each host, for example, C:\glassfish3. If the directory does not exist, the subcommand creates the directory.

The user that is running this subcommand must have write access to the specified directory. Otherwise, an error occurs.

To overwrite an existing installation of the GlassFish Server software, set the --force option to true. If the directory already contains an installation and the --force option is false, an error occurs.

The default is the parent of the base installation directory of the GlassFish Server software on the host where this subcommand is run.

--windowsuser
-w
The user on the host where the GlassFish Server software is to be installed that is to run the process for connecting through DCOM to the host. The default is the user that is running this subcommand.
The name of the Windows domain that contains the user that the --windowsuser option specifies. The default is the name of the host on which the subcommand is run.

Specifies whether the subcommand overwrites an existing installation of the GlassFish Server software in the directory that the --installdir option specifies. Possible values are as follows:

false
The existing installation is not overwritten (default).

true
The existing installation is overwritten.

**Operands**

*host-list*
A space-separated list of the names of the hosts where the GlassFish Server software is to be installed.

**Examples**

**Example 1 Installing GlassFish Server Software at the Default Location**

This example installs GlassFish Server software on the hosts wpmdl1.example.com and wpmdl2.example.com at the default location.

Some lines of output are omitted from this example for readability.

```
asadmin> install-node-dcom wpmdl1.example.com wpmdl2.example.com
Created installation zip C:\glassfish8107276692860773166.zip
Copying 85760199 bytes..........................................................
....................................
WROTE FILE TO REMOTE SYSTEM: C:/glassfish3/glassfish_install.zip and
C:/glassfish3/unpack.bat
Output from Windows Unpacker:

C:\Windows\system32>C:
C:\Windows\system32>cd "C:\glassfish3"
C:\glassfish3>jar xvf glassfish_install.zip
inflated: bin/asadmin
inflated: bin/asadmin.bat
inflated: glassfish/bin/appclient
inflated: glassfish/bin/appclient.bat
inflated: glassfish/bin/appclient.js
inflated: glassfish/bin/asadmin
inflated: glassfish/bin/asadmin.bat
...
inflated: mq/lib/props/broker/default.properties
inflated: mq/lib/props/broker/install.properties

Command install-node-dcom executed successfully.
```
Exit Status

0
command executed successfully

1
error in executing the command

See Also

install-node-ssh(1), uninstall-node-dcom(1)

asadmin(1M)
install-node-ssh

installs GlassFish Server software on specified SSH-enabled hosts

Synopsis

install-node-ssh [--help]
[--archive archive]
[--create={false|true}] [--save={false|true}]
[--installdir as-install-parent]
[--sshport ssh-port] [--sshuser ssh-user]
[--sshkeyfile ssh-keyfile]
[--force={false|true}]
host-list

Description

The install-node-ssh subcommand installs GlassFish Server software on the hosts that are specified as the operand of the subcommand. This subcommand requires secure shell (SSH) to be configured on the host where the subcommand is run and on each host where the GlassFish Server software is being installed.

If necessary, the subcommand creates a ZIP archive of the GlassFish Server software from the installation where this subcommand is run. The archive does not contain the domains directory or the nodes directory. These directories are synchronized from the domain administration server (DAS) when instances on nodes that represent the hosts are created and started. The subcommand does not delete the archive after installing the GlassFish Server software from the archive on the specified hosts.

If multiple hosts are specified, the configuration of the following items is the same on all hosts:

- Base installation directory
- SSH port
- SSH user
- SSH key file

If the SSH key file does not exist on a host where the GlassFish Server software is to be installed, the subcommand runs interactively and prompts for a password. To enable the subcommand to run noninteractively, the following conditions must be met:

- The --interactive option of the asadmin(1M) utility must be false.
- The --passwordfile option of the asadmin utility must specify a password file.
- The password file must contain the AS_ADMIN_SSHPASSWORD entry.

The subcommand does not modify the configuration of the DAS.
This subcommand is supported in local mode only.

Options

--help
-?
Displays the help text for the subcommand.

--archive
The absolute path to the archive file of the GlassFish Server software that is to be installed. If no archive file is specified, the subcommand creates an archive from the installation of GlassFish Server software from which the subcommand is run. This archive is created in the home directory of the user that is running the command.

--create
Specifies whether the subcommand should create an archive file of the GlassFish Server software to install.

false
No archive file is created. The subcommand installs the software from the existing archive file that the --archive option specifies (default).

true
The subcommand creates an archive file from the installation of GlassFish Server software from which the subcommand is run.

--save
Specifies whether the archive file from which the software is installed is saved after installation.

false
The archive file is not saved. The subcommand deletes the file after installing the software (default).

true
The archive file is saved.

--installdir
The absolute path to the parent of the base installation directory where the GlassFish Server software is to be installed on each host, for example, /export/glassfish3/. If the directory does not exist, the subcommand creates the directory.

The user that is running this subcommand must have write access to the specified directory. Otherwise, an error occurs.

To overwrite an existing an installation of the GlassFish Server software, set the --force option to true. If the directory already contains an installation and the --force option is false, an error occurs.

The default is the parent of the base installation directory of the GlassFish Server software on the host where this subcommand is run.

--sshport
The port to use for SSH connections to the host where the GlassFish Server software is to be installed. The default is 22.
**--sshuser**
The user on the host where the GlassFish Server software is to be installed that is to run the process for connecting through SSH to the host. The default is the user that is running this subcommand. To ensure that the DAS can read this user's SSH private key file, specify the user that is running the DAS process.

**--sshkeyfile**
The absolute path to the SSH private key file for user that the **--sshuser** option specifies. This file is used for authentication to the sshd daemon on the host.

The user that is running this subcommand must be able to reach the path to the key file and read the key file.

The default is a key file in the user's .ssh directory. If multiple key files are found, the subcommand uses the following order of preference:

1. id_rsa
2. id_dsa
3. identity

**--force**
Specifies whether the subcommand overwrites an existing installation of the GlassFish Server software in the directory that the **--installdir** option specifies. Possible values are as follows:

- **false**
  The existing installation is not overwritten (default).

- **true**
  The existing installation is overwritten.

### Operands
**host-list**
A space-separated list of the names of the hosts where the GlassFish Server software is to be installed.

### Examples

**Example 1 Installing GlassFish Server Software at the Default Location**
This example installs GlassFish Server software on the hosts sj03.example.com and sj04.example.com at the default location.

```
asadmin> install-node-ssh sj03.example.com sj04.example.com
Created installation zip /home/gfuser/glassfish2339538623689073993.zip
Successfully connected to gfuser@sj03.example.com using keyfile /home/gfuser/.ssh/id_rsa
Copying /home/gfuser/glassfish2339538623689073993.zip (81395008 bytes) to sj03.example.com:/export/glassfish3
Installing glassfish2339538623689073993.zip into sj03.example.com:/export/glassfish3
Removing sj03.example.com:/export/glassfish3/glassfish2339538623689073993.zip
Fixing file permissions of all files under sj03.example.com:/export/glassfish3/bin
Successfully connected to gfuser@sj04.example.com using keyfile /home/gfuser/.ssh/id_rsa
Copying /home/gfuser/glassfish2339538623689073993.zip (81395008 bytes) to sj04.example.com:/export/glassfish3
```


install-node-ssh

Installing glassfish2339538623689073993.zip into sj04.example.com:/export/glassfish3
Removing sj04.example.com:/export/glassfish3/glassfish2339538623689073993.zip
Fixing file permissions of all files under sj04.example.com:/export/glassfish3/bin
Command install-node-ssh executed successfully

Exit Status

0
command executed successfully

1
error in executing the command

See Also

install-node(1), install-node-dcom(1), uninstall-node(1), uninstall-node-ssh(1)
asadmin(1M)
jms-ping

checks if the JMS service is up and running

Synopsis

jms-ping [--help]
[-- target target]

Description

The jms-ping subcommand checks if the Java Message Service (JMS) service (also known as the JMS provider) is up and running. When you start the GlassFish Server, the JMS service starts by default.

The jms-ping subcommand pings only the default JMS host within the JMS service. It displays an error message when it is unable to ping a built-in JMS service.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

--help
-?  
Displays the help text for the subcommand.

--target

Specifies the target for which the operation is to be performed. Valid values are as follows:

  server  
Pings the JMS service for the default server instance. This is the default value

  configuration-name  
Pings the JMS service for all clusters using the specified configuration.

  cluster-name  
Pings the JMS service for the specified cluster.

  instance-name  
Pings the JMS service for the specified server instance.

Examples

Example 1 Verifying that the JMS service is running
The following subcommand checks to see if the JMS service is running on the default server.

    asadmin> jms-ping
    JMS-ping command executed successfully
    Connector resource test_jms_adapter created.
    Command jms-ping executed successfully.
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jms-host(1), list-jms-hosts(1), delete-jms-host(1)
asadmin(1M)
list

lists configurable or monitorable elements

Synopsis

list [--help] [--monitor={false|true}] [dotted-parent-attribute-name]

Description

The list subcommand lists configurable and monitorable attributes of GlassFish Server.

The output of the list subcommand is a list of the dotted names that represent individual server components and subsystems. For example, server.applications.web-module. After you know the particular component or subsystem, you can then use the get subcommand to access any attributes, and the set subcommand to modify configurable attributes.

The following rules apply to dotted names in a list subcommand:

---

**Note:** Characters that have special meaning to the shell or command interpreter, such as * (asterisk), should be quoted or escaped as appropriate to the shell, for example, by enclosing the argument in quotes. In multimode, quotes are needed only for arguments that include spaces, quotes, or backslash.

---

- Any list subcommand that has a dotted name that is not followed by a wildcard (*) lists the current node's immediate children. For example, the following command lists all immediate children belonging to the server node:

  asadmin> list server

- Any list subcommand that has a dotted name followed by a wildcard(*) lists a hierarchical tree of child nodes from the current node. For example, the following command lists all child nodes of applications and their subsequent child nodes, and so on:

  asadmin> list server.applications.*

- Any list subcommand that has a dotted name preceded or followed by a wildcard (*) of the form *dotted name or dottedname* lists all nodes and their child nodes that match the regular expression created by the provided matching pattern.

For detailed information about dotted names, see the dotted-names(5ASC) help page.

Options

--help

Displays the help text for the subcommand.
--monitor
-m
Defaults to false. If set to false, the configurable attribute values are returned. If set to true, the monitorable attribute values are returned.

Operands

dotted-parent-element-name
Configurable or monitorable element name

Examples

Example 1 Listing Dotted Names of Configurable Elements
This example lists the elements that can be configured.

asadmin> list *
applications
configs
configs.config.server-config
configs.config.server-config.admin-service
configs.config.server-config.admin-service.das-config
configs.config.server-config.admin-service.jms-connector.system
configs.config.server-config.admin-service.property.adminConsoleContextRoot
configs.config.server-config.admin-service.property.adminConsoleDownloadLocation
configs.config.server-config.admin-service.property.ipsRoot
configs.config.server-config.ejb-container
configs.config.server-config.ejb-container.ejb-timer-service
configs.config.server-config.http-service
configs.config.server-config.http-service.access-log
configs.config.server-config.http-service.virtual-server.__asadmin
configs.config.server-config.http-service.virtual-server.server
configs.config.server-config.iioip-service
configs.config.server-config.iioip-service.iioip-listener.SSL
configs.config.server-config.iioip-service.iioip-listener.SSL.ssl
configs.config.server-config.iioip-service.iioip-listener.SSL_MUTUALAUTH
configs.config.server-config.iioip-service.iioip-listener.SSL_MUTUALAUTH.ssl
configs.config.server-config.iioip-service.orb-listener-1
configs.config.server-config.iioip-service.orb
configs.config.server-config.java-config
configs.config.server-config.jms-service
configs.config.server-config.jms-service.jms-host.default_JMS_host
...
property.administrative.domain.name
resources
resources.jdbc-connection-pool.DerbyPool
resources.jdbc-connection-pool.DerbyPool.property.DatabaseName
resources.jdbc-connection-pool.DerbyPool.property.Password
resources.jdbc-connection-pool.DerbyPool.property.PortNumber
resources.jdbc-connection-pool.DerbyPool.property.User
resources.jdbc-connection-pool.DerbyPool.property.connectionAttributes
resources.jdbc-connection-pool.DerbyPool.property.serverName
resources.jdbc-connection-pool.__TimerPool
resources.jdbc-connection-pool.__TimerPool.property.connectionAttributes
resources.jdbc-connection-pool.__TimerPool.property.databaseName
resources.jdbc-resource.jdbc/__TimerPool
resources.jdbc-resource.jdbc/__default
servers
servers.server.server
servers.server.server.resource-ref.jdbc/__TimerPool
Example 2  Listing Attributes of a Configurable Element
This example lists the attributes of the web container.

```bash
asadmin> list configs.config.server-config.web-container
configs.config.server-config.web-container
configs.config.server-config.web-container.session-config
Command list executed successfully.
```

Example 3  Listing Dotted Names of Monitorable Objects
This example lists the names of the monitorable objects that are enabled for monitoring.

```bash
asadmin> list --monitor *
server.jvm
server.jvm.class-loading-system
server.jvm.compilation-system
server.jvm.garbage-collectors
server.jvm.garbage-collectors.Copy
server.jvm.garbage-collectors.MarkSweepCompact
server.jvm.memory
server.jvm.operating-system
server.jvm.runtime
server.network
server.network.admin-listener
server.network.admin-listener.connections
server.network.admin-listener.file-cache
server.network.admin-listener.keep-alive
server.network.admin-listener.thread-pool
server.network.http-listener-1
server.network.http-listener-1.connections
server.network.http-listener-1.file-cache
server.network.http-listener-1.keep-alive
server.network.http-listener-1.thread-pool
server.transaction-service
Command list executed successfully.
```

Exit Status

- 0
  subcommand executed successfully

- 1
  error in executing the subcommand

See Also

- `get(1)`, `set(1)`
- `dotted-names(5ASC)`
- `asadmin(1M)`

*Oracle GlassFish Server Administration Guide*
list-admin-objects

gets all the administered objects

Synopsis

list-admin-objects [--help] [target]

Description

The list-admin-objects subcommand lists all the administered objects. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
The target for which administered objects are to be listed. Valid values are as follows:

server
Lists the administered objects on the default server instance. This is the default value.

configuration-name
Lists the administered objects in the specified configuration.

cluster-name
Lists the administered objects on all server instances in the specified cluster.

instance-name
Lists the administered objects on a specified server instance.

Examples

Example 1 Listing Administered Objects
This example lists all the administered objects.

asadmin> list-admin-objects
jms/samplequeue
jms/anotherqueue
Command list-admin-objects executed successfully

Exit Status

0
subcommand executed successfully
error in executing the subcommand

See Also

create-admin-object(1), delete-admin-object(1)
asadmin(1M)
list-application-refs

lists the existing application references

Synopsis

list-application-refs [--help] [--long={false|true}] [target]

Description

The list-application-refs subcommand lists all application references in a cluster or an unclustered server instance. This effectively lists all the modules deployed on the specified target (for example, J2EE applications, Web modules, and enterprise bean modules).

If multiple versions of a module or application are deployed, this subcommand lists all versions. To list which version is enabled, set the --long option to true. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.

The target instance or instances making up the cluster need not be running or available for this subcommand to succeed.

This subcommand is supported in remote mode only.

Options

--help

Displays the help text for the subcommand.

--long

If true, displays whether each module or application listed is enabled. The default is false.

Operands

**target**

The target for which you are listing the application references. Valid values are

- server- Specifies the default server instance as the target. server is the name of the default server instance and is the default value.

- cluster_name- Specifies a certain cluster as the target.

- instance_name- Specifies a certain server instance as the target.

Examples

**Example 1 Listing Application References**

The following example lists the application references for the unclustered server instance NewServer.

asadmin> list-application-refs NewServer
ClientSessionMDBApp
MEjbApp
__ejb_container_timer_app
Command list-application-refs executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-application-ref(1), delete-application-ref(1)
asadmin(1M)

Oracle GlassFish Server Application Deployment Guide
**list-applications**

lists deployed applications

**Synopsis**

list-applications [--help]
[--long={false|true}] [--resources] [--subcomponents]
[--type type] [target]

**Description**

The `list-applications` subcommand lists deployed Java EE applications and the type of each application that is listed.

If the `--type` option is not specified, all applications are listed. If the type option is specified, you must specify a type. The possible types are listed in the Options section of this help page.

If multiple versions of a module or application are deployed, this subcommand lists all versions. To list which version is enabled, set the `--long` option to true. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.

This subcommand is supported in remote mode only.

**Options**

**--help**

Displays the help text for the subcommand.

**--long**

If true, displays whether each module or application listed is enabled. The default is false.

**--resources**

Lists the application-scoped resources for each application. If the `--subcomponents` option is also used, lists the application-scoped resources for each component within the application.

**--subcomponents**

Lists the subcomponents of each application. The subcomponents listed depend on the application type. For example, for a Java EE application (EAR file), modules are listed. For a web application, servlets and JSP pages are listed. For an EJB module, EJB subcomponents are listed.

**--type**

Specifies the type of the applications that are to be listed. The options are as follows:

- application
- appclient
- connector
- ejb
- web
- webservice

If no type is specified, all applications are listed.

Operands

--target
This is the name of the target upon which the subcommand operates. The valid values are as follows:

server
Lists the applications for the default server instance server and is the default value.

domain
Lists the applications for the domain.

cluster_name
Lists the applications for the cluster.

instance_name
Lists the applications for a particular stand-alone server instance.

Examples

Example 1 Listing the Web Applications

asadmin> list-applications --type web
hellojsp <web>
Command list-applications executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

list-components(1), list-sub-components(1), show-component-status(1)

asadmin(1M)

Oracle GlassFish Server Application Deployment Guide
list-audit-modules

gets all audit modules and displays them

Synopsis

list-audit-modules [--help] [target]

Description

The list-audit-modules subcommand lists all the audit modules. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
Specifies the target on which you are listing the audit modules. Valid values are as follows:

server
Lists the audit modules for the default server instance server and is the default value.

configuration_name
Lists the audit modules for the named configuration.

cluster_name
Lists the audit modules for every server instance in the cluster.

instance_name
Lists the audit modules for a particular server instance.

Examples

Example 1 Listing Audit Modules
asadmin> list-audit-modules
sampleAuditModule1
sampleAuditModule2
Command list-audit-modules executed successfully

Exit Status

0
command executed successfully
1
error in executing the command

See Also

create-audit-module(1), delete-audit-module(1)
asadmin(1M)
list-auth-realms

lists the authentication realms

Synopsis

list-auth-realms [--help] [target]

Description

The list-auth-realms subcommand lists the authentication realms. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
The name of the target for which you want to list the authentication realms.

server
Lists the realms for the default server instance server and is the default value.

collection_name
Lists the realms for the named configuration.

collection_name
Lists the realms for every server instance in the cluster.

instance_name
Lists the realms for a particular server instance.

Examples

Example 1 Listing authentication realms

asadmin> list-auth-realms
file
ldap
certificate
db
Command list-auth-realms executed successfully

Where file, ldap, certificate, and db are the available authentication realms.

Exit Status

0

command executed successfully
error in executing the command

See Also

create-auth-realm(1), delete-auth-realm(1)
asadmin(1M)
**list-backup-configs**

lists existing domain backup configurations

**Synopsis**

```
list-backup-configs [--help]
[--long={false|true}] [backup-config-name]
```

**Description**

The `list-backup-configs` subcommand lists existing domain backup configurations. This subcommand is supported in remote mode only.

**Options**

--help
-?
Displays the help text for the subcommand.

--long
-l
Provides detailed information about the listed domain backup configurations. The default value is false.

If the --long option is specified and a `backup-config-name` operand is provided, information about the schedule used by the given domain backup configuration is also provided.

**Operands**

`backup-config-name`
Restricts the listing to the named domain backup configuration.

If the --long option is specified and a `backup-config-name` operand is provided, information about the schedule used by the given domain backup configuration is also provided.

**Examples**

**Example 1 Listing Backup Configurations**

This example provides detailed information about all existing domain backup configurations.

```
asadmin> list-backup-configs --long
Name of Backup Config :monthly-backup
Auto Backup Enabled   :true
Schedule               :monthly
Recycle Limit          :12
Config Only backup     :false
Active Backup Enabled  :false
```
Backup Directory        :null
Last Backup Attempt     :
Last Successful Backup  :
Name of Backup Config   :weekly-backup
Auto Backup Enabled     :true
Schedule                :weekly
Recycle Limit           :5
Config Only backup      :true
Active Backup Enabled   :false
Backup Directory        :null
Last Backup Attempt     :
Last Successful Backup  :

Command list-backup-configs executed successfully.

**Exit Status**

<table>
<thead>
<tr>
<th>Exit Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>subcommand executed successfully</td>
</tr>
<tr>
<td>1</td>
<td>error in executing the subcommand</td>
</tr>
</tbody>
</table>

**See Also**

create-backup-config(1), delete-backup-config(1), disable-backup-config(1), enable-backup-config(1), asadmin(1M)
list-backups

lists all backups

Synopsis

list-backups [--help]
[--help=({false|true})]
[--domaindir domain-root-dir]
[--backupdir backup-directory]
[--backupconfig backup-config-name]
[domain-name]

Description

The list-backups subcommand displays information about domain backups. This subcommand is supported in local mode only in GlassFish Server Open Source Edition, and is support in local mode and remote mode in Oracle GlassFish Server.

Options

--help
-?
Displays the help text for the subcommand.

--long
-l
Displays detailed information about each backup.
The default value is false.

--domaindir
Specifies the domain root directory, the parent directory of the domain upon which the command will operate.
The default value is as-install/domains.

--backupdir
Specifies the directory under which backup files are stored.
The default value is as-install/domains/domain-dir/backups. If the domain is not in the default location, the location is domain-dir/backups.

--backupconfig
(Supported only in Oracle GlassFish Server.) Restricts the listing of backup files in the backup directory to those for the specified domain backup configuration.

Operands

domain-name
Specifies the domain for which backups are listed.
This operand is optional if only one domain exists in the GlassFish Server installation.
Examples

**Example 1 Listing Domain Backups**
This example provides detailed information about backups in the default domain.

```
list-backups --long
```

```
Description : domain1 backup created on 2010_12_01 by user adminuser
GlassFish Version : Oracle GlassFish Server 3.1
Backup User : adminuser
Backup Date : Wed Dec 01 09:22:45 PST 2010
Domain Name : domain1
Backup Type : full
Backup Config Name : 
Backup Filename (origin) : /glassfish3/glassfish/domains/domain1/backups/domain1_2010_12_01_v00001.zip
Domain Directory : /glassfish3/glassfish/domains/domain1

Description : domain1 backup created on 2010_12_16 by user adminuser
GlassFish Version : Oracle GlassFish Server 3.1
Backup User : adminuser
Backup Date : Thu Dec 16 09:23:48 PST 2010
Domain Name : domain1
Backup Type : full
Backup Config Name : 
Backup Filename (origin) : /glassfish3/glassfish/domains/domain1/backups/domain1_2010_12_16_v00001.zip
Domain Directory : /glassfish3/glassfish/domains/domain1
```

Command list-backups executed successfully.

**Exit Status**

```
0
```

- subcommand executed successfully

```
1
```

- error in executing the subcommand

**See Also**

backup-domain(1), restore-domain(1)
list-clusters

lists existing clusters in a domain

Synopsis

list-clusters [--help] [target]

Description

The list-clusters subcommand lists existing clusters in a domain. The list can be filtered by cluster, instance, node, or configuration. For each cluster that is listed, the subcommand indicates whether the cluster is running.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
Filters the list of clusters by specifying the target for which the clusters are to be listed. Valid values are as follows:

domain
Lists all clusters in the domain (default).

cluster-name
Lists only the specified cluster.

instance-name
Lists the cluster of which the specified instance is a member.

node-name
Lists the clusters that contain an instance that resides on the specified node. For example, if instance pmdii in cluster pmdc and instance ymlii in cluster ymlc reside on node n1, pmdc and ymlc are listed.

configuration-name
Lists all clusters that contain instances whose configuration is defined by the named configuration.

Examples

Example 1  Listing All Clusters in a Domain
This example lists all clusters in the current domain.

asadmin> list-clusters
pmdclust not running
ymlclust not running

Command list-clusters executed successfully.

Example 2 Displaying the Status of a Cluster
This example displays status of the cluster ymlclust, which is not running.

```
asadmin> list-clusters ymlclust
ymlclust not running
```

Command list-clusters executed successfully.

Example 3 Listing All Clusters That Are Associated With a Node
This example lists the clusters that contain an instance that resides on the node sj02.

```
asadmin> list-clusters sj02
ymlclust not running
```

Command list-clusters executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-cluster(1), delete-cluster(1), start-cluster(1), stop-cluster(1)

asadmin(1M)
list-commands

lists available commands

Synopsis

list-commands [--help]
[--localonly={false|true}] [--remoteonly={false|true}]

pattern-list

Description

The list-commands subcommand lists the asadmin subcommands.
By default, the list-commands subcommand displays a list of local subcommands
followed by a list of remote subcommands. You can specify that only remote
subcommands or only local subcommands are listed and that only subcommands
whose names contain a specified text string are listed.

This subcommand is supported in local mode and remote mode.

Options

--help
-?
Displays the help text for the subcommand.

--localonly
If this option is set to true, only local commands are listed. Default is false.
If this option is set to true, the --remoteonly option must be set to false. Otherwise, an
error occurs.

--remoteonly
If this option is set to true, only remote commands are listed. Default is false.
If this option is set to true, the --localonly option must be set to false. Otherwise, an
error occurs.

Operands

pattern-list
A space-separated list of text strings on which to filter the list of subcommands. Only
the subcommands that contain any one of the specified text strings is listed.

Examples

Example 1  Listing the Local Subcommands
This example lists only the local subcommands.

asadmin> list-commands --localonly=true
*********** Local Commands ***********
change-admin-password
change-master-password
create-domain
create-service
delete-domain
export
help
list-commands
list-domains
login
monitor
multimode
restart-domain
start-database
start-domain
stop-database
stop-domain
unset
verify-domain-xml
version
Command list-commands executed successfully.

Example 2  Filtering the Subcommands That Are Listed
This example lists only the subcommands whose names contain the text configure or set.

asadmin> list-commands configure set
********** Local Commands **********
setup-ssh
unset

********** Remote Commands **********
configure-jms-cluster set-log-levels
configure-lb-weight set-web-context-param
configure-ldap-for-admin set-web-env-entry
set unset-web-context-param
set-log-attributes unset-web-env-entry

Command list-commands executed successfully.

Example 3  Listing All Subcommands
This example first displays a list of the local subcommands, followed by a partial list of the remote subcommands.

asadmin> list-commands
********** Local Commands **********
change-admin-password
change-master-password
create-domain
create-service
delete-domain
export
help
list-commands
list-domains
login
monitor
multimode
restart-domain
start-database
start-domain
stop-database
stop-domain
unset
verify-domain-xml
version

********** Remote Commands **********

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__locations</td>
<td>enable</td>
</tr>
<tr>
<td>add-resources</td>
<td>enable-monitoring</td>
</tr>
<tr>
<td>configure-ldap-for-admin</td>
<td>flush-jmsdest</td>
</tr>
<tr>
<td>create-admin-object</td>
<td>freeze-transaction-service</td>
</tr>
<tr>
<td>create-audit-module</td>
<td>generate-jvm-report</td>
</tr>
<tr>
<td>create-auth-realm</td>
<td>get</td>
</tr>
<tr>
<td>create-connector-connection-pool</td>
<td>get-client-stubs</td>
</tr>
<tr>
<td>create-connector-resource</td>
<td>get-host-and-port</td>
</tr>
<tr>
<td>create-connector-security-map</td>
<td>jms-ping</td>
</tr>
<tr>
<td>create-connector-work-security-map</td>
<td>list</td>
</tr>
<tr>
<td>create-custom-resource</td>
<td>list-admin-objects</td>
</tr>
<tr>
<td>create-file-user</td>
<td>list-app-refs</td>
</tr>
<tr>
<td>create-http</td>
<td>list-applications</td>
</tr>
<tr>
<td>create-http-listener</td>
<td>list-audit-modules</td>
</tr>
<tr>
<td>create-iiop-listener</td>
<td>list-auth-realms</td>
</tr>
<tr>
<td>create-javamail-resource</td>
<td>list-components</td>
</tr>
<tr>
<td>create-jdbc-connection-pool</td>
<td>list-connector-connection-pools</td>
</tr>
<tr>
<td>create-jdbc-resource</td>
<td>list-connector-resources</td>
</tr>
<tr>
<td>create-jms-host</td>
<td>list-connector-security-maps</td>
</tr>
<tr>
<td>create-jms-resource</td>
<td>list-connector-work-security-maps</td>
</tr>
<tr>
<td>create-jmsdest</td>
<td>list-containers</td>
</tr>
<tr>
<td>create-jndi-resource</td>
<td>list-custom-resources</td>
</tr>
<tr>
<td>create-jvm-options</td>
<td>list-file-groups</td>
</tr>
<tr>
<td>create-lifecycle-module</td>
<td>list-file-users</td>
</tr>
<tr>
<td>create-message-security-provider</td>
<td>list-http-listeners</td>
</tr>
<tr>
<td>create-network-listener</td>
<td>list-iio-listeners</td>
</tr>
<tr>
<td>create-password-alias</td>
<td>list-javamail-resources</td>
</tr>
<tr>
<td>create-profiler</td>
<td>list-jdbc-connection-pools</td>
</tr>
<tr>
<td>create-protocol</td>
<td>list-jdbc-resources</td>
</tr>
<tr>
<td>create-resource-adapter-config</td>
<td>list-jms-hosts</td>
</tr>
<tr>
<td>create-resource-ref</td>
<td>list-jms-resources</td>
</tr>
<tr>
<td>create-ssl</td>
<td>list-jmsdest</td>
</tr>
<tr>
<td>create-system-properties</td>
<td>list-jndi-entries</td>
</tr>
<tr>
<td>create-threadpool</td>
<td>list-jndi-resources</td>
</tr>
<tr>
<td>create-transport</td>
<td>list-jvm-options</td>
</tr>
<tr>
<td>create-virtual-server</td>
<td>list-lifecycle-modules</td>
</tr>
<tr>
<td>delete-admin-object</td>
<td>list-logger-levels</td>
</tr>
<tr>
<td>delete-audit-module</td>
<td>list-message-security-providers</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

list-components(1), list-containers(1), list-modules(1)
asadmin(1M)
list-components

lists deployed components

Synopsis

list-components [--help]
[-long=(false|true)] [--resources] [--subcomponents]
[--type type] [target]

Description

Note: The list-components subcommand is deprecated. Use the list-applications subcommand instead.

The list-components subcommand lists all deployed Java EE components.

If the --type option is not specified, all components are listed. If the type option is specified, you must specify a type. The possible types are listed in the Options section in this help page.

If multiple versions of a module or application are deployed, this subcommand lists all versions. To list which version is enabled, set the --long option to true. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--long
If true, displays whether each module or application listed is enabled. The default is false.

--resources
Lists the application-scoped resources for each component. If the --subcomponents option is also used, lists the application-scoped resources for each component within an application.

--subcomponents
Lists the subcomponents of each component. The subcomponents listed depend on the component type. For example, for a Java EE application (EAR file), modules are listed. For a web application, servlets and JSP pages are listed. For an EJB module, EJB subcomponents are listed.

--type
Specifies the type of the components that are to be listed. The options are as follows:
list-components

- application
- appclient
- connector
- ejb
- web
- webservice

If no type is specified, all components are listed.

Operands

**target**

This is the name of the target upon which the subcommand operates. The valid values are:

- server
  
  Lists the components for the default server instance server and is the default value.

- domain
  
  Lists the components for the domain.

- cluster_name
  
  Lists the components for the cluster.

- instance_name
  
  Lists the components for a particular stand-alone server instance.

Examples

**Example 1 Listing Components**

This example lists the connector components. (cciblackbox-tx.rar was deployed.)

```
asadmin> list-components --type connector
cciblackbox-tx <connector>
Command list-components executed successfully
```

Exit Status

0

subcommand executed successfully

1

error in executing the subcommand

See Also

list-applications(1), show-component-status(1)
asadmin(1M)

Oracle GlassFish Server Application Deployment Guide
list-configs

lists named configurations

Synopsis

list-configs [--help] [target]

Description

The list-configs subcommand lists named configurations in the configuration of the domain administration server (DAS). The list can be filtered by cluster, instance, or named configuration.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
Filters the list of configurations. Valid values are as follows:

  domain
  Lists all named configurations in the current domain.

  cluster-name
  Lists the named configuration that defines the configuration of instances in the specified cluster.

  instance-name
  Lists the named configuration that defines the configuration of the specified instance.

  configuration-name
  Lists the specified named configuration. Use this option to determine whether a named configuration exits.

Examples

Example 1 Listing All Named Configurations in a Domain
This example lists all named configurations in the current domain.

asadmin> list-configs
server-config
default-config
pmdclust-config
pmdsharedconfig
pmdcpinst-config
list-configs

yamlclust-config
il1-config
il2-config

Command list-configs executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

copy-config(1), delete-config(1)
asaadmin(1M)
configuration(5ASC)
list-connector-connection-pools

lists the existing connector connection pools

Synopsis

list-connector-connection-pools [--help]

Description

The list-connector-connection-pools subcommand lists the existing connector connection pools that have been created.

This subcommand is supported in remote mode only.

Options

--help
-
Displays the help text for the subcommand.

Examples

Example 1 Listing the Connector Connection Pools
This example lists the existing connector connection pools.

asadmin> list-connector-connection-pools
jms/qConnPool
Command list-connector-connection-pools executed successfully

Exit Status

0
subcommand executed successfully

1
ter error in executing the subcommand

See Also

create-connector-connection-pool(1), delete-connector-connection-pool(1),
ping-connection-pool(1)
asadmin(1M)
list-connector-resources

lists all connector resources

Synopsis

list-connector-resources [--help] [target]

Description

The list-connector-resources subcommand lists all connector resources.
This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
The target for which the connector resources are to be listed. Valid values are as
follows:

  server
  Lists the connector resources on the default server instance. This is the default
  value.

  domain
  Lists the connector resources for the domain.

  cluster-name
  Lists the connector resources on all server instances in the specified cluster.

  instance-name
  Lists the connector resources on a specified server instance.

Examples

Example 1 Listing Connector Resources
This example lists all existing connector resources.

asadmin> list-connector-resources
jms/qConnFactory
Command list-connector-resources executed successfully.

Exit Status

0
subcommand executed successfully
error in executing the subcommand

See Also

create-connector-resource(1), delete-connector-resource(1)
asadmin(1M)
list-connector-security-maps

lists the security maps belonging to the specified connector connection pool

Synopsis

list-connector-security-maps [--help] [--securitymap securitymap] [--verbose={false|true}] [--target target] pool-name

Description

The list-connector-security-maps subcommand lists the security maps belonging to the specified connector connection pool.

For this subcommand to succeed, you must have first created a connector connection pool using the create-connector-connection-pool subcommand.

This subcommand is supported in remote mode only.

Options

--help
-
-?
Displays the help text for the subcommand.

--securitymap

Specifies the name of the security map contained within the connector connection pool from which the identity and principals should be listed. With this option, --verbose is redundant.

--verbose

If set to true, returns a list including the identity, principals, and security name. The default is false.

--target

Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

Operands

pool-name

Name of the connector connection pool for which you want to list security maps.

Examples

Example 1 Listing the Connector Security Maps
This example lists the existing connector security maps for the pool named connector-Pool1.

asadmin> list-connector-security-maps connector-Pool1
securityMap1
Command list-connector-security-maps executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-connector-security-map(1), delete-connector-security-map(1),
update-connector-security-map(1)
asaadmin(1M)
list-connector-work-security-maps

lists the work security maps belonging to the specified resource adapter

Synopsis

list-connector-work-security-maps [--help] [--securitymap securitymap] resource_adapter_name

Description

The list-connector-work-security-maps subcommand lists the work security maps belonging to the specified resource adapter.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--securitymap

Specifies the name of the security map contained within the resource adapter from which the identity and principals should be listed.

Operands

resource_adapter_name

The name of the resource adapter for which you want to list security maps.

Examples

Example 1 Listing Connector Work Security Maps

This example lists the current connector work security maps for the resource adapter named my_resource_adapter.

asadmin> list-connector-work-security-maps my_resource_adapter
workSecurityMap1: EIS principal=eis-principal-2, mapped principal=server-principal-2
workSecurityMap1: EIS principal=eis-principal-1, mapped principal=server-principal-1
workSecurityMap2: EIS principal=eis-principal-1, mapped principal=server-principal-1
Command list-connector-work-security-maps executed successfully.

Exit Status

0
subcommand executed successfully
error in executing the subcommand

See Also

create-connector-work-security-map(1), delete-connector-work-security-map(1),
update-connector-work-security-map(1)

asadmin(1M)
**list-containers**

lists application containers

**Synopsis**

```
list-containers [--help]
```

**Description**

The `list-containers` subcommand displays a list of application containers. This subcommand is supported in remote mode only.

**Options**

```
--help
-?
```

Displays the help text for the subcommand.

**Examples**

**Example 1 Listing the Application Containers**

This example lists the current application containers.

```
asadmin> list-containers
List all known application containers
Container : grizzly
Container : ejb
Container : webservices
Container : ear
Container : appclient
Container : connector
Container : jpa
Container : web
Container : osgi
Container : security
Container : webbeans

Command list-containers executed successfully.
```

**Exit Status**

- **0** subcommand executed successfully
- **1** error in executing the subcommand

**See Also**

`list-commands(1), list-components(1), list-modules(1)`
asadmin(1M)
list-custom-resources

gerets all custom resources

Synopsis

list-custom-resources [--help] [target]

Description

The list-custom-resources subcommand lists the custom resources. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

**target**
This operand specifies the location of the custom resources. Valid targets are:

  * server
    Lists the resources on the default server instance. This is the default value
  * domain
    Lists the resources in the domain.
  * cluster_name
    Lists the resources for every server instance in the cluster.
  * instance_name
    Lists the resources for a particular server instance.

Examples

**Example 1 Listing Custom Resources**
This example lists the current custom resources.

asadmin> list-custom-resources
sample_custom_resource01
sample_custom_resource02
Command list-custom-resources executed successfully.

**Example 2 Using the list-custom-resources command with a target**
The following example displays the usage of this command.

asadmin> list-custom-resources --user admin --passwordfile passwords.txt --host plum --port 4848 target6
sample_custom_resource03
sample_custom_resource04
Command list-custom-resources executed successfully.

**Exit Status**

- **0**
  subcommand executed successfully

- **1**
  error in executing the subcommand

**See Also**

create-custom-resource(1), delete-custom-resource(1)
asadmin(1M)
list-domains

lists the domains in the specified directory

Synopsis

list-domains [--help] [--domaindir domaindir]

Description

The list-domains subcommand lists the domains in the specified domains directory. If the domains directory is not specified, the domains in the default directory are listed. If there is more than one domains directory, the --domaindir option must be specified. The status of each domain is included.

This subcommand is supported in local mode only.

Options

--help
-?
Displays the help text for the subcommand.

--domaindir
The directory where the domains are to be listed. If specified, the path must be accessible in the files stem. If not specified, the domains in the domain root directory are listed. The default location of the domain root directory is as-install/domains.

Examples

Example 1 Listing Domains
This example lists the domains in the default directory.

asadmin> list-domains
Name: domain1 Status: Running
Name: domain2 Status: Not running
Name: domain4 Status: Running, restart required to apply configuration changes
Command list-domains executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-domain(1), delete-domain(1), start-domain(1), stop-domain(1),
asadmin(1M)
list-file-groups

lists file groups

Synopsis

list-file-groups [--help] [--name username] [--authrealmname auth_realm_name] [--target target]

Description

The list-file-groups subcommand lists the file users and groups supported by the file realm authentication. This subcommand lists available groups in the file user. If the --name option is not specified, all groups are listed.

This subcommand is supported in remote mode only.

Options

--help

Displays the help text for the subcommand.

--name

Identifies the name of the file user for whom the groups will be listed.

--authrealmname

The name of the authentication realm for which to list available groups.

--target

This option specifies which configurations you can list. Valid targets are:

server
Lists the file groups in the current server. This is the default value.

cluster_name
Lists the file groups in a cluster.

instance_name
Lists the file groups for a particular instance.

Examples

Example 1 Listing Groups in all File Realms

This example list all file realm groups defined for the server.

asadmin> list-file-groups
staff
manager
Command list-file-groups executed successfully
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-file-user(1), update-file-user(1), create-file-user(1),
list-file-users(1)
asadmin(1M)
list-file-users

lists the file users

Synopsis

list-file-users [--help] [--authrealmname auth_realm_name] [target]

Description

The list-file-users subcommand displays a list of file users supported by file realm authentication.

Options

--help
-?
Displays the help text for the subcommand.

--authrealmname
Lists only the users in the specified authentication realm.

Operands

target
Specifies the target for which you want to list file users. The following values are valid:

server
Lists the file users on the default server instance. This is the default value.

configuration_name
Lists the file users in the specified configuration.

cluster_name
Lists the file users on all server instances in the specified cluster.

instance_name
Lists the file users on a specified server instance.

Examples

Example 1 Listing Users in a Specific File Realm
The following example lists the users in the file realm named sample_file_realm.

asadmin> list-file-users --authrealmname sample_file_realm
sample_user05
sample_user08
sample_user12
Command list-file-users executed successfully
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-file-user(1), delete-file-user(1), update-file-user(1),
list-file-groups(1)

asadmin(1M)
list-http-lb-configs

lists load balancer configurations

Synopsis

list-http-lb-configs [--help] [target]

Description

The list-http-lb-configs subcommand lists the load balancer configurations. List them all or list them by the cluster or server instance they reference.

Note: This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
Lists the load balancers by target. Valid values are:

- cluster_name- The name of a target cluster.
- instance_name- The name of a target server instance.

Examples

Example 1 Listing Load Balancer Configurations Without a Target
This example lists all load balancer configurations defined for all GlassFish Server clusters and instances.

asadmin> list-http-lb-configs

mycluster-http-lb-config server1instlb
Command list-http-lb-configs executed successfully.

Example 2 Listing Load Balancer Configurations for a Specific Target
This example lists the load balancer configuration defined for a cluster named mycluster.

asadmin> list-http-lb-configs mycluster

mycluster-http-lb-config
Command list-http-lb-configs executed successfully.

**Exit Status**

0
subcommand executed successfully

1
error in executing the subcommand

**See Also**

delete-http-lb-config(1), create-http-lb-config(1)

asadmin(1M)
list-http-lbs

lists load balancers

Synopsis

list-http-lbs [--help]

Description

Use the list-http-lbs subcommand to list physical load balancers.

Note: This subcommand is only applicable to Oracle GlassFish Server. This subcommand is not applicable to GlassFish Server Open Source Edition.

Options

--help
-?
Displays the help text for the subcommand.

Examples

Example 1 Listing Physical Load Balancers for a Domain
This example lists all physical load balancers defined for a domain.

asadmin> list-http-lbs

lb1
lb2
Command list-http-lbs executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-http-lb(1), delete-http-lb(1)

asadmin(1M)
list-http-listeners

lists the existing network listeners

Synopsis

list-http-listeners [--help] [target]

Description

The list-http-listeners subcommand lists the existing network listeners. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
Restricts the listing to network listeners for a specified target. Valid values are as follows:

server
Lists the network listeners for the default server instance. This is the default value.

configuration-name
Lists the network listeners for the specified configuration.

cluster-name
Lists the network listeners for all server instances in the specified cluster.

instance-name
Lists the network listeners for the specified server instance.

Examples

Example 1  Listing Network Listeners
The following command lists all the network listeners for the server instance:

asadmin> list-http-listeners
http-listener-1
http-listener-2
admin-listener
Command list-http-listeners executed successfully.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-http-listener(1), delete-http-listener(1)
asadmin(1M)
list-iiop-listeners

lists the existing IIOP listeners

Synopsis

list-iiop-listeners [--help] [target]

Description

The list-iiop-listeners subcommand lists the existing IIOP listeners. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
This operand specifies the target for which the IIOP listeners are to be listed. Valid values are:

  server
  Lists the listeners in the default server instance server and is the default value.

  configuration_name
  Lists the listeners in the specified configuration.

  cluster_name
  Lists the listeners in the specified cluster.

  instance_name
  Lists the listeners in a particular server instance.

Examples

Example 1 Using the list-iiop-listeners subcommand
The following command lists all the IIOP listeners for the server instance:

asadmin> list-iiop-listeners
orb-listener-1
SSL
SSL_MUTUALAUTH
sample_iiop_listener
Command list-iiop-listeners executed successfully.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-iiop-listener(1), delete-iiop-listener(1)
asadmin(1M)
list-instances

lists GlassFish Server instances in a domain

Synopsis

list-instances [--help] [--timeoutmsec timeout] [--long={false|true}] [--nostatus={false|true}] [--standaloneonly={false|true}] [target]

Description

The list-instances subcommand lists GlassFish Server instances in a domain. The list can be filtered by cluster, instance, node, or configuration.

The subcommand displays every GlassFish Server instance in the specified target, regardless of how each instance was created. For example, this subcommand lists instances that were created by using the create-instance(1) subcommand and by using the create-local-instance(1) subcommand.

By default, the subcommand indicates whether each instance that is listed is running. Options of this subcommand control the information that is displayed for each instance.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--timeoutmsec
The time limit in milliseconds for determining the status of instances. The default is 2,000, which is equivalent to 2 seconds.

--long
-1
Specifies whether detailed information is displayed for each instance that is listed.

The --long option and --nostatus option are mutually exclusive. If both options are specified in the same command, an error occurs.

Valid values are as follows:

true
The following details are displayed for each instance that is listed:

- The name of the instance
- The name of the host where the instance's node resides
- The HTTP port on which the instance listens for administration requests
- The process identifier (PID) of the instance process or -1 if the instance is not running
The name of the cluster of which the instance is a member, if any

- The state of the instance, which is running or not running

When an instance is listed, some configuration changes in the domain administration server (DAS) for the instance might not have been applied to the instance itself. In this situation, the commands that are required to apply the changes are listed adjacent to the state of the instance. The maximum number of commands that are listed for an instance is 10.

false
Only the name of the instance and an indication of whether the instance is running are displayed (default). The length of time that the instance has been running is not displayed.

--nostatus
Specifies whether information about whether instances are running is suppressed.

The --long option and --nostatus option are mutually exclusive. If both options are specified in the same command, an error occurs.

Valid values are as follows:

true
Information about whether instances are running is suppressed. Only the name of each instance is displayed.

false
Information about whether instances are running is displayed (default).

--standaloneonly
Specifies whether only standalone instances are listed.

The --standaloneonly option and the target operand and are mutually exclusive. If both the --standaloneonly option and the target operand are specified in the same command, an error occurs.

Valid values are as follows:

true
Only standalone instances are listed.

false
All instances in the specified target are listed (default).

Operands

target
Filters the list of GlassFish Server instances by specifying the target for which instances are listed.

The target operand and the --standaloneonly option are mutually exclusive. If both the target operand and the --standaloneonly option are specified in the same command, an error occurs.

Valid values are as follows:

domain
Lists all instances in the domain (default).
**cluster-name**
Lists the instances that are members of the specified cluster.

**instance-name**
Lists only the specified instance.

**node-name**
Lists the instances that reside on the specified node.

**configuration-name**
Lists all instances whose configuration is defined by the specified named configuration.

**Examples**

**Example 1 Listing Basic Information About All GlassFish Server Instances in a Domain**
This example lists the name and status of all GlassFish Server instances in the current domain.

```
asadmin> list-instances
pmd-i-sj02 running
yml-i-sj02 running
pmd-i-sj01 running
yml-i-sj01 running
pmdsa1 not running
Command list-instances executed successfully.
```

**Example 2 Listing Detailed Information About All GlassFish Server Instances in a Domain**
This example lists detailed information about all GlassFish Server instances in the current domain.

```
asadmin> list-instances --long=true
NAME        HOST       PORT   PID    CLUSTER     STATE
pmd-i-sj01  sj01       24848  31310  pmdcluster   running
yml-i-sj01  sj01       24849  25355  ymlcluster   running
pmdsa1      localhost  24848  -1     ---          not running
pmd-i-sj02  sj02       24848  22498  pmdcluster   running
yml-i-sj02  sj02       24849  20476  ymlcluster   running
ymlsa1      localhost  24849  -1     ---          not running
Command list-instances executed successfully.
```

**Example 3 Displaying the Status of an Instance**
This example displays status of the instance pmd-i-sj01, which is running.

```
asadmin> list-instances pmd-i-sj01
pmd-i-sj01 running
Command list-instances executed successfully.
```

**Example 4 Listing Only Standalone Instances in a Domain**
This example lists only the standalone instances in the current domain.

```
asadmin> list-instances --standaloneonly=true
```

Oracle GlassFish Server 3.1.2 Reference Manual
pmdsal not running
Command list-instances executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-instance(1), create-local-instance(1)

asadmin(1M)
list-jacc-providers

enables administrators to list JACC providers defined for a domain

Synopsis

list-jacc-providers [--help] [target]

Description

The list-jacc-providers subcommand enables administrators to list the JACC providers defined for a domain. JACC providers are defined as jacc-provider elements in the security-service element in the domain's domain.xml file. JACC providers can be created using the GlassFish Server Admin Console or the create-jacc-provider subcommand.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target

Specifies the target for which you want to list JACC providers. The following values are valid:

server
Lists the JACC providers on the default server instance. This is the default value.

configuration_name
Lists the JACC providers in the specified configuration.

cluster_name
Lists the JACC providers on all server instances in the specified cluster.

instance_name
Lists the JACC providers on a specified server instance.

Examples

Example 1 Listing JACC providers
The following example shows how to list JACC providers for the default domain.

asadmin> list-jacc-providers
default
simple
testJACC

Command list-jacc-providers executed successfully.
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jacc-provider(1), delete-jacc-provider(1)
asadmin(1M)
list-javamail-resources

lists the existing JavaMail session resources

Synopsis

list-javamail-resources [--help] [target]

Description

The list-javamail-resources subcommand lists the existing JavaMail session resources.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
This operand specifies the target for which the JavaMail session resources are to be listed. Valid values are:

server
Lists the resources for the default server instance. This is the default value.

domain
Lists the resources for the domain.

cluster_name
Lists the resources for the specified cluster.

instance_name
Lists the resources for a particular server instance.

Examples

Example 1 Listing JavaMail Resources
This example lists the JavaMail session resources for the server instance.

asadmin> list-javamail-resources
mail/MyMailSession
Command list-javamail-resources executed successfuly.

Exit Status

0
subcommand executed successfully
error in executing the subcommand

See Also

create-javamail-resource(1), delete-javamail-resource(1)
asadmin(1M)
### list-jdbc-connection-pools

lists all JDBC connection pools

#### Synopsis

```shell
list-jdbc-connection-pools [--help]
```

#### Description

The `list-jdbc-connection-pools` subcommand lists the current JDBC connection pools.

This subcommand is supported in the remote mode only.

#### Options

- `--help`  
- `?-?
  Displays the help text for the subcommand.

#### Examples

**Example 1 Listing the JDBC Connection Pools**

This example lists the existing JDBC connection pools.

```shell
asadmin> list-jdbc-connection-pools
sample_derby_pool
__TimerPool
Command list-jdbc-connection-pools executed successfully.
```

#### Exit Status

- **0**  
  subcommand executed successfully

- **1**  
  error in executing the subcommand

#### See Also

- `create-jdbc-connection-pool(1)`, `delete-jdbc-connection-pool(1)`
- `asadmin(1M)`
**list-jdbc-resources**

lists all JDBC resources

**Synopsis**

list-jdbc-resources [--help] [target target]

**Description**

The `list-jdbc-resources` subcommand displays a list of the existing JDBC resources. This subcommand is supported in remote mode only.

**Options**

--help

-?

Displays the help text for the subcommand.

**Operands**

--target

This operand specifies which JDBC resources you can list. Usage of this operand is optional. Valid values are:

  server
  Lists the JDBC resources in the current server and is the default.

  domain
  Lists the JDBC resources in the current domain.

  cluster_name
  Lists the JDBC resources in a cluster.

  instance_name
  Lists the JDBC resources for a particular instance.

**Examples**

**Example 1 Listing the JDBC Resources**

This example lists the current JDBC resources.

```bash
asadmin> list-jdbc-resources
jdbc/DerbyPool
Command list-jdbc-resources executed successfully.
```

**Exit Status**

0

subcommand executed successfully
error in executing the subcommand

**See Also**

create-jdbc-resource(1), delete-jdbc-resource(1)

asadmin(1M)
list-jmsdest

lists the existing JMS physical destinations

Synopsis

list-jmsdest [--help]
[--desttype type]
[target]

Description

The list-jmsdest subcommand lists the Java Message Service (JMS) physical destinations.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

--help
-?
Displays the help text for the subcommand.

--desttype
The type of JMS destination to be listed. Valid values are topic and queue.

Operands

target
Restricts the listing to physical destinations for a specified target. Valid values are as follows:

server
Lists the physical destinations for the default server instance. This is the default value.

configuration-name
Lists the physical destinations in the specified configuration.

cluster-name
Lists the physical destinations for every server instance in the specified cluster.

instance-name
Lists the physical destinations for the specified server instance.

Examples

Example 1 Listing all physical destinations
The following subcommand lists all the physical destinations.

asadmin> list-jmsdest
PhysicalQueue
PhysicalTopic
Command list-jmsdest executed successfully.

Example 2  Listing all physical destinations of a specified type
The following subcommand lists all physical topics.

```
asadmin> list-jmsdest --desttype topic
PhysicalTopic
Command list-jmsdest executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jmsdest(1), delete-jmsdest(1), flush-jmsdest(1)
asadmin(1M)
list-jms-hosts

lists the existing JMS hosts

Synopsis

list-jms-hosts [--help]
[--target target]

Description

The list-jms-hosts subcommand lists the existing Java Message Service (JMS) hosts for the JMS service.

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

--help
-?
Displays the help text for the subcommand.

--target
Restricts the listing to JMS hosts for a specified target. Valid values are as follows:

  server
  Lists the JMS hosts for the default server instance. This is the default value.

  configuration-name
  Lists the JMS hosts for the specified configuration.

  cluster-name
  Lists the JMS hosts for all server instances in the specified cluster.

  instance-name
  Lists the JMS hosts for the specified server instance.

Examples

Example 1  Listing all JMS hosts
The following subcommand lists the JMS hosts for the JMS service.

asadmin> list-jms-hosts
default_JMS_host
MyNewHost
Command list-jms-hosts executed successfully.

Exit Status

0
subcommand executed successfully
error in executing the subcommand

See Also

create-jms-host(1), delete-jms-host(1), jms-ping(1)
asadmin(1M)
list-jms-resources

lists the JMS resources

Synopsis

list-jms-resources [--help]
 [--restype type]
 [target]

Description

The list-jms-resources subcommand lists the existing Java Message Service (JMS) resources (destination and connection factory resources).

This subcommand is supported in remote mode only. Remote asadmin subcommands require a running domain administration server (DAS).

Options

--help
-?
Displays the help text for the subcommand.

--restype

Operands

target
Restricts the listing to resources for a specified target. Valid values are as follows:

server
Lists the resources for the default server instance. This is the default value.

domain
Lists the resources for the domain.

cluster-name
Lists the resources for every server instance in the specified cluster.

instance-name
Lists the resources for the specified server instance.

Examples

Example 1 Listing all JMS resources
The following subcommand lists all JMS resources.

asadmin> list-jms-resources
jms/Queue
jms/ConnectionFactory
jms/DurableConnectionFactory
jms/Topic
Command list-jms-resources executed successfully.

Example 2 Listing JMS resources of a specified type
The following subcommand lists all javax.jms.ConnectionFactory resources.

asadmin> list-jms-resources --restype javax.jms.ConnectionFactory
jms/ConnectionFactory
jms/DurableConnectionFactory
Command list-jms-resources executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jms-resource(1), delete-jms-resource(1)
asadmin(1M)
list-jndi-entries

browses and queries the JNDI tree

Synopsis

list-jndi-entries [--help]
[--context context_name]
[target]

Description

Use this subcommand to browse and query the JNDI tree.
The subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--context

The name of the JNDI context or subcontext. If context is not specified, all entries in
the naming service are returned. If context (such as ejb) is specified, all those entries
are returned.

Operands

target
This operand specifies the JNDI tree to browse.
Possible values are as follows:

server
Browses the JNDI tree for the default GlassFish Server instance (default). The
default instance is the domain administration server (DAS).

domain
Browses the JNDI tree for the current domain.

cluster-name
Browses the JNDI tree for the specified cluster.

instance-name
Browses the JNDI tree for the specified GlassFish Server instance.

Examples

Example 1  Browsing the JNDI Tree
This example browses the JNDI tree for the default GlassFish Server instance.

asadmin> list-jndi-entries
Command list-jndi-entries executed successfully.

**Exit Status**

0
subcommand executed successfully

1
error in executing the subcommand

**See Also**

create-jndi-resource(1), delete-jndi-resource(1), list-jndi-resources(1)
asadmin(1M)
list-jndi-resources

lists all existing JNDI resources

Synopsis

list-jndi-resources [--help] [target]

Description

The list-jndi-resources subcommand identifies all existing JNDI resources. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
The target for which the JNDI resources are to be listed. Valid values are as follows:

server
Lists the JNDI resources on the default server instance. This is the default value.

configuration-name
Lists the JNDI resources for the specified configuration.

cluster-name
Lists the JNDI resources on all server instances in the specified cluster.

instance-name
Lists the JNDI resources on a specified server instance.

Examples

Example 1 Listing JNDI Resources
This example lists the JNDI resources on the default server instance.

asadmin> list-jndi-resources
jndi_resource1
jndi_resource2
jndi_resource3
Command list-jndi-resources executed successfully

Exit Status

0
subcommand executed successfully
error in executing the subcommand

See Also

create-jndi-resource(1), delete-jndi-resource(1)
asadmin(1M)
list-jvm-options

lists options for the Java application launcher

Synopsis

list-jvm-options [--help] [--target target] [--profiler={false|true}]

Description

The list-jvm-options subcommand displays a list of command-line options that are passed to the Java application launcher when GlassFish Server is started.

The options are managed by using the JVM Options page of the Administration Console or by using the create-jvm-options and delete-jvm-options subcommands.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
Specifies the target for which you are listing Java application launcher options.
Valid values are as follows:

server
Specifies the DAS (default).

instance-name
Specifies a GlassFish Server instance.

cluster-name
Specifies a cluster.

configuration-name
Specifies a named configuration.

--profiler
Specifies whether the Java application launcher options to list are for the profiler. Set this option to true only if a profiler has been configured. If this option is set to true and no profiler is configured, an error occurs. The default is false.

Examples

Example 1 Listing the Java Application Launcher Options

This example lists the options that are used by the Java application launcher.

asadmin> list-jvm-options
-Djava.security.auth.login.config=${com.sun.aas.instanceRoot}/config/login.conf
-XX: LogVMOutput
-XX: UnlockDiagnosticVMOptions
-Dcom.sun.enterprise.config.config_environment_factory_class=com.sun.enterprise.config.serverbeans.AppserverConfigEnvironmentFactory
-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/config/keystore.jks
-XX:NewRatio=2
-DANTLR_USE_DIRECT_CLASS_LOADING=true
-Djava.security.policy=${com.sun.aas.instanceRoot}/config/server.policy
-Djdbc.drivers=org.apache.derby.jdbc.ClientDriver
-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/config/cacerts.jks
-XX:NewRatio=2
-Djava.ext.dirs=${com.sun.aas.javaRoot}/lib/ext$\{path.separator\}${com.sun.aas.javaRoot}/jre/lib/ext$\{path.separator\}${com.sun.aas.instanceRoot}/lib/ext$\{path.separator\}${com.sun.aas.derbyRoot}/lib
-XX:MaxPermSize=192m
-XX:LogFile=${com.sun.aas.instanceRoot}/logs/jvm.log

Command list-jvm-options executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-jvm-options(1), delete-jvm-options(1)

asadmin(1M)

For more information about the Java application launcher, see the reference page for the operating system that you are using:

- Oracle Solaris and Linux: java - the Java application launcher
  (http://java.sun.com/javase/6/docs/technotes/tools/solaris/java.html)

- Windows: java - the Java application launcher
  (http://java.sun.com/javase/6/docs/technotes/tools/windows/java.html)
**list-libraries**

lists library JAR files on GlassFish Server

**Synopsis**

```
list-libraries [--help] [--type={common|ext|app}]
```

**Description**

The `list-libraries` subcommand lists library archive files on GlassFish Server. The `--type` option specifies the library type and the GlassFish Server directory for which libraries are to be listed.

This subcommand is supported in remote mode only.

**Options**

--help

Displays the help text for the subcommand.

--type

Specifies the library type and the GlassFish Server directory for which libraries are listed. Valid values are as follows:

- **common**
  Lists the library files for the Common class loader directory, `domain-dir/lib`. This is the default.

- **ext**
  Lists the library files for the Java optional package directory, `domain-dir/lib/ext`.

- **app**
  Lists the library files for the application-specific class loader directory, `domain-dir/lib/applibs`.

For more information about these directories, see "Class Loaders" in *Oracle GlassFish Server Application Development Guide*.

**Examples**

**Example 1 Listing Libraries**

This example lists the libraries in the application-specific class loader directory on the default server instance.

```
asadmin> list-libraries --type app
mylib.jar
xlib.jar
ylib.jar
zlib.jar
Command list-libraries executed successfully.
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

add-library(1), remove-library(1)
asadmin(1M)
"Class Loaders" in Oracle GlassFish Server Application Development Guide
list-lifecycle-modules

lists the lifecycle modules

Synopsis

list-lifecycle-modules [--help] [target]

Description

The list-lifecycle-modules subcommand lists lifecycle modules. A lifecycle module provides a means of running a short or long duration Java-based task at a specific stage in the server life cycle. This subcommand is supported in remote mode only.

Options

--help
?- Displays the help text for the subcommand.

Operands

target
Indicates the location where lifecycle modules are to be listed. Valid values are

■ server- Specifies the default server instance as the target for listing lifecycle modules. server is the name of the default server instance and is the default value for this operand.

■ cluster_name- Specifies a particular cluster as the target for listing lifecycle modules.

■ instance_name- Specifies a particular server instance as the target for listing lifecycle modules.

Examples

Example 1 Listing Lifecycle Modules

asadmin> list-lifecycle-modules
WSTCPConnectorLCModule
Command list-lifecycle-modules executed successfully

WSTCPConnectorLCModule is the only lifecycle module listed for the default target, server.

Exit Status

0
command executed successfully

1
error in executing the command
See Also

create-lifecycle-module(1), delete-lifecycle-module(1)
asadmin(1M)
list-log-attributes

list-log-attributes lists all logging attributes defined for a specified target in a domain

Synopsis

list-log-attributes [--help] [target]

Description

The list-log-attributes subcommand lists all logging attributes currently defined for the specified GlassFish Server domain or target within a domain. The values listed correspond to the values in the logging.properties file for the domain.

This subcommand is supported in remote mode only.

Options

--help
?-
Displays the help text for the subcommand.

Operands

target
Valid values are:

- server - The default server instance. This is the default value.
- configuration_name - The name of a specific configuration.
- cluster_name - The name of a target cluster.
- instance_name - The name of a target server instance.

Examples

Example 1 Listing the Logger Attributes for a Domain

This example lists all loggers attributes for the default domain.

asadmin> list-log-attributes
com.sun.enterprise.server.logging.GFFileHandler.alarms <false>
com.sun.enterprise.server.logging.GFFileHandler.file \n<$(com.sun.aas.instanceRoot)/logs/server.log>
com.sun.enterprise.server.logging.GFFileHandler.flushFrequency <1>
com.sun.enterprise.server.logging.GFFileHandler.formatter \n<com.sun.enterprise.server.logging.UniformLogFormatter>
com.sun.enterprise.server.logging.GFFileHandler.logtoConsole <false>
com.sun.enterprise.server.logging.GFFileHandler.maxHistoryFiles <0>
com.sun.enterprise.server.logging.GFFileHandler.retainErrorsStasticsForHours <0>
com.sun.enterprise.server.logging.GFFileHandler.rotationLimitInBytes <2000000>
com.sun.enterprise.server.logging.GFFileHandler.rotationTimlimitInMinutes <0>
com.sun.enterprise.server.logging.SyslogHandler.useSystemLogging <false>
handlers <java.util.logging.ConsoleHandler>
java.util.logging.ConsoleHandler.formatter \n
Command list-log-attributes executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

collect-log-files(1), list-log-levels(1), rotate-log(1), set-log-attributes(1),
set-log-levels(1)

asadmin(1M)

"Administering the Logging Service" in Oracle GlassFish Server Administration Guide
list-log-levels

lists the loggers and their log levels

Synopsis

list-log-levels [--help] [--target target]

Description

The list-log-levels subcommand lists the current GlassFish Server loggers and their log levels. This subcommand reports on all the loggers that are listed in the logging.properties file. In some cases, loggers that have not been created by the respective containers will appear in the list.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
The server, cluster or server instance for which log levels will be listed.

Operands

target
Valid values are:
- server_name - Default target is server. If no target is specified then log levels are listed for the server.
- cluster_name - The name of a target cluster.
- instance_name - The name of a target server instance.

Examples

Example 1 Listing the Log Levels
This example lists the existing loggers and indicates how their log levels are set.

asadmin> list-log-levels
java.util.logging.ConsoleHandler <FINEST>
javax.enterprise.resource.corba <INFO>
javax.enterprise.resource.javamail <INFO>
javax.enterprise.resource.jdo <INFO>
javax.enterprise.resource.jms <INFO>
javax.enterprise.resource.jta <INFO>
javax.enterprise.resource.resourceadapter <INFO>
javax.enterprise.resource.sqltrace <FINE>
javax.enterprise.resource.webcontainer.jsf.application <INFO>
javax.enterprise.resource.webcontainer.jsf.config <INFO>
Command list-log-levels executed successfully.

**Exit Status**

0
subcommand executed successfully

1
error in executing the subcommand

**See Also**

collect-log-files(1), list-log-attributes(1), rotate-log(1),
set-log-attributes(1), set-log-levels(1)

asadmin(1M)

"Administering the Logging Service" in Oracle GlassFish Server Administration Guide
**list-message-security-providers**

lists all security message providers for the given message layer

**Synopsis**

```
list-message-security-providers [--help]
--layer message_layer
[target]
```

**Description**

The `list-message-security-providers` subcommand enables administrators to list all security message providers (`provider-config sub-elements`) for the given message layer (`message-security-config element of domain.xml`).

This subcommand is supported in remote mode only.

**Options**

```
--help
-?
```

Displays the help text for the subcommand.

```
--layer
```

The message-layer for which the provider has to be listed. The default value is HttpServlet.

**Operands**

```
target
```

Restricts the listing to message security providers for a specific target. Valid values include:

```
server
```

Lists providers for the default server instance `server` and is the default value.

```
domain
```

Lists providers for the domain.

```
cluster
```

Lists providers for the server instances in the cluster.

```
instance
```

Lists providers for a particular server instance.

**Examples**

**Example 1  Listing message security providers**

The following example shows how to list message security providers for a message layer.

```
asadmin> list-message-security-providers
```
--layer SOAP
XWS_ClientProvider
ClientProvider
XWS_ServerProvider
ServerProvider
Command list-message-security-providers executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also
create-message-security-provider(1), delete-message-security-provider(1)
asadmin(1M)
list-modules

lists GlassFish Server modules

Synopsis

list-modules [--help]

Description

The list-modules subcommand displays a list of modules that are accessible to the GlassFish Server module subsystem. The version of each module is shown.

This subcommand is supported in remote mode only.

Options

--help  
-?  
Displays the help text for the subcommand.

Examples

Example 1 Listing GlassFish Server Modules

This example provides a partial listing of modules that are accessible to the GlassFish Server module subsystem

asadmin> list-modules
List Of Modules

Module : org.glassfish.transaction.jts:3.0.0.b66
  Module Characteristics : List of Jars implementing the module
     Jar : file:/home/gfuser/GlassFish/glassfishv3/glassfish/modules/jts.jar
  Module Characteristics : Provides to following services
  Module Characteristics : List of imported modules
     Imports : org.glassfish.transaction.jts:3.0.0.b66
Module : com.sun.enterprise.tiger-types-osgi:0.3.96
Module : org.glassfish.bean-validator:3.0.0.JBoss-400Beta3A
Module : org.glassfish.core.kernel:3.0.0.b66
  Module Characteristics : Provides to following services
  Module Characteristics : List of imported modules
     Imports : org.glassfish.core.kernel:3.0.0.b66
Module : org.glassfish.common.util:3.0.0.b66
  Module Characteristics : Provides to following services
  Module Characteristics : List of imported modules
     Imports : org.glassfish.common.util:3.0.0.b66
...  
Command list-modules executed successfully
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

list-commands(1), list-components(1), list-containers(1)
asadmin(1M)
list-network-listeners

lists the existing network listeners

Synopsis

list-network-listeners [--help] [target]

Description

The list-network-listeners subcommand lists the existing network listeners. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
Restricts the listing to network listeners for a specified target. Valid values are as follows:

server
Lists the network listeners for the default server instance. This is the default value.

configuration-name
Lists the network listeners for the specified configuration.

cluster-name
Lists the network listeners for all server instances in the specified cluster.

instance-name
Lists the network listeners for the specified server instance.

Examples

Example 1 Listing Network Listeners
The following command lists all the network listeners for the server instance:

```bash
asadmin> list-network-listeners
admin-listener
http-listener-1
https-listener-2
Command list-network-listeners executed successfully.
```
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-network-listener(1), delete-network-listener(1)

asadmin(1M)
list-nodes

lists all GlassFish Server nodes in a domain

Synopsis

list-nodes [--help] [--long={false|true}]

Description

The list-nodes subcommand lists all GlassFish Server nodes in a domain. By default, the subcommand displays the following information for each node that is listed:

- The name of the node
- The type of the node, which is one of the following types:
  
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>The node does not support remote communication.</td>
</tr>
<tr>
<td>DCOM</td>
<td>The node supports communication over the DCOM remote protocol.</td>
</tr>
<tr>
<td>SSH</td>
<td>The node supports communication over SSH.</td>
</tr>
</tbody>
</table>

- The name of the host that the node represents

The --long option of the subcommand specifies whether the nodes are listed in long format. In long format, the following additional information about each node is displayed:

- The path to the parent of the base installation directory of GlassFish Server on the host that the node represents
- A comma-separated list of the names of the GlassFish Server instances that reside on the node

If the --terse option of the asadmin(1M) utility is true and the --long option of the subcommand is false, the subcommand lists only the name of each node.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--long
-1
Specifies whether the nodes are listed in long format.

Possible values are as follows:
true
The nodes are listed in long format.

false
The nodes are listed in short format (default).

Examples

Example 1 Listing GlassFish Server Nodes
This example displays the name, type, and host for all GlassFish Server nodes in the domain domain1.

asadmin> list-nodes
localhost-domain1  CONFIG  localhost
sj02  SSH  sj02.example.com
sj01  SSH  sj01.example.com
devnode  CONFIG  localhost
Command list-nodes executed successfully.

Example 2 Listing Only the Names of GlassFish Server Nodes
This example uses the --terse option of the asadmin utility to list only the names of the GlassFish Server nodes in the domain domain1.

asadmin> list-nodes --terse=true
localhost-domain1
sj02
sj01
devnode

Example 3 Listing GlassFish Server Nodes in Long Format
This example lists the GlassFish Server nodes in the domain domain1 in long format.

asadmin> list-nodes --long=true
NODE NAME           TYPE     NODE HOST          INSTALL DIRECTORY     REFERENCED
BY
localhost-domain1   CONFIG   localhost          /export/glassfish3
sj02                SSH      sj02.example.com   /export/glassfish3    pmd-i2,
yml-i2
sj01                SSH      sj01.example.com   /export/glassfish3    pmd-i1,
yml-i1
devnode             CONFIG   localhost          /export/glassfish3    pmdsa1
Command list-nodes executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command
See Also

create-node-config(1), create-node-dcom(1), create-node-ssh(1),
delete-node-config(1), delete-node-dcom(1), delete-node-ssh(1),
list-nodes-config(1), list-nodes-dcom(1), list-nodes-ssh(1)

asadmin(1M)
list-nodes-config

lists all GlassFish Server nodes that do not support remote communication in a domain

Synopsis

list-nodes-config [--help] [--long={false|true}]

Description

The list-nodes-config subcommand lists all GlassFish Server nodes that do not support remote communication in a domain.

Note: To list all nodes in a domain regardless of the type of the node, run the list-nodes(1) subcommand.

By default, the subcommand displays the following information for each node that is listed:

- The name of the node
- The type of the node, which is CONFIG
- The name of the host that the node represents

The --long option of the subcommand specifies whether the nodes are listed in long format. In long format, the following additional information about each node is displayed:

- The path to the parent of the base installation directory of GlassFish Server on the host that the node represents
- A comma-separated list of the names of the GlassFish Server instances that reside on the node

If the --terse option of the asadmin(1M) utility is true and the --long option of the subcommand is false, the subcommand lists only the name of each node.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--long
-1
Specifies whether the nodes are listed in long format.
Possible values are as follows:

true
The nodes are listed in long format.
false
The nodes are listed in short format (default).

Examples

Example 1 Listing GlassFish Server Nodes
This example displays the name, type, and host for all GlassFish Server nodes that do not support remote communication in the domain domain1.

asadmin> list-nodes-config
localhost-domain1  CONFIG  localhost
devnode  CONFIG  localhost
Command list-nodes-config executed successfully.

Example 2 Listing Only the Names of GlassFish Server Nodes
This example uses the --terse option of the asadmin utility to list only the names of the GlassFish Server nodes that do not support remote communication in the domain domain1.

asadmin> list-nodes-config --terse=true
localhost-domain1
devnode

Example 3 Listing GlassFish Server Nodes in Long Format
This example lists the GlassFish Server nodes that do not support remote communication in the domain domain1 in long format.

asadmin> list-nodes-config --long=true
NODE NAME           TYPE     NODE HOST   INSTALL DIRECTORY    REFERENCED BY
localhost-domain1   CONFIG   localhost   /export/glassfish3
devnode             CONFIG   localhost   /export/glassfish3   pmdsal
Command list-nodes-config executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-node-config(1), create-node-dcom(1), create-node-ssh(1),
delete-node-config(1), delete-node-dcom(1), delete-node-ssh(1), list-nodes(1),
list-nodes-dcom(1), list-nodes-ssh(1)
asadmin(1M)
**list-nodes-dcom**

lists all GlassFish Server nodes that support communication over DCOM in a domain

**Synopsis**

```
list-nodes-dcom [--help] [--long={false|true}]
```

**Description**

The `list-nodes-dcom` subcommand lists all GlassFish Server nodes that support communication over the Distributed Component Object Model (DCOM) remote protocol. The DCOM protocol is available only on Windows systems.

**Note:** To list all nodes in a domain regardless of the type of the node, run the `list-nodes(1)` subcommand.

By default, the subcommand displays the following information for each node that is listed:

- The name of the node
- The type of the node, which is **DCOM**
- The name of the host that the node represents

The `--long` option of the subcommand specifies whether the nodes are listed in long format. In long format, the following additional information about each node is displayed:

- The path to the parent of the base installation directory of GlassFish Server on the host that the node represents
- A comma-separated list of the names of the GlassFish Server instances that reside on the node

If the `--terse` option of the `asadmin(1M)` utility is `true` and the `--long` option of the subcommand is `false`, the subcommand lists only the name of each node.

This subcommand is supported in remote mode only.

**Options**

```
--help
-?
```

Displays the help text for the subcommand.

```
--long
-1
```

Specifies whether the nodes are listed in long format.

Possible values are as follows:

```
true
```

The nodes are listed in long format.
false
The nodes are listed in short format (default).

Examples

Example 1 Listing GlassFish Server Nodes
This example displays the name, type, and host for all GlassFish Server nodes that support communication over DCOM in a domain.

asadmin> list-nodes-dcom
xkyd  DCOM  xkyd.example.com
wpmdl2  DCOM  wpmdl2.example.com
wpmdl1  DCOM  wpmdl1.example.com
Command list-nodes-dcom executed successfully.

Example 2 Listing Only the Names of GlassFish Server Nodes
This example uses the --terse option of the asadmin utility to list only the names of the GlassFish Server nodes that support communication over DCOM in a domain.

asadmin> list-nodes-dcom --terse=true
xkyd
wpmdl2
wpmdl1

Example 3 Listing GlassFish Server Nodes in Long Format
This example lists the GlassFish Server nodes that support communication over DCOM in a domain in long format.

asadmin> list-nodes-dcom --long=true
NODE NAME    TYPE   NODE HOST            INSTALL DIRECTORY   REFERENCED BY
xkyd         DCOM   xkyd.example.com     C:\glassfish3
wpmdl2       DCOM   wpmdl2.example.com   C:\glassfish3       wdi2
wpmdl1       DCOM   wpmdl1.example.com   C:\glassfish3       wdi1
Command list-nodes-dcom executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-node-config(1), create-node-dcom(1), create-node-ssh(1),
delete-node-config(1), delete-node-dcom(1), delete-node-ssh(1), list-nodes(1),
list-nodes-config(1), list-nodes-dcom(1)

asadmin(1M)
list-nodes-ssh

lists all GlassFish Server nodes that support communication over SSH in a domain

Synopsis

list-nodes-ssh [--help] [--long={false|true}]

Description

The list-nodes-ssh subcommand lists all GlassFish Server nodes that support communication over secure shell (SSH) in a domain.

Note: To list all nodes in a domain regardless of the type of the node, run the list-nodes(1) subcommand.

By default, the subcommand displays the following information for each node that is listed:

- The name of the node
- The type of the node, which is SSH
- The name of the host that the node represents

The --long option of the subcommand specifies whether the nodes are listed in long format. In long format, the following additional information about each node is displayed:

- The path to the parent of the base installation directory of GlassFish Server on the host that the node represents
- A comma-separated list of the names of the GlassFish Server instances that reside on the node

If the --terse option of the asadmin(1M) utility is true and the --long option of the subcommand is false, the subcommand lists only the name of each node.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--long
-1
Specifies whether the nodes are listed in long format.

Possible values are as follows:

  true
  The nodes are listed in long format.
false
The nodes are listed in short format (default).

Examples

Example 1 Listing GlassFish Server Nodes
This example displays the name, type, and host for all GlassFish Server nodes that support communication over SSH in a domain.

```
asadmin> list-nodes-ssh
sj02  SSH  sj02.example.com
sj01  SSH  sj01.example.com
Command list-nodes-ssh executed successfully.
```

Example 2 Listing Only the Names of GlassFish Server Nodes
This example uses the --terse option of the asadmin utility to list only the names of the GlassFish Server nodes that support communication over SSH in a domain.

```
asadmin> list-nodes-ssh --terse=true
sj02
sj01
```

Example 3 Listing GlassFish Server Nodes in Long Format
This example lists the GlassFish Server nodes that support communication over SSH in a domain in long format.

```
asadmin> list-nodes-ssh --long=true
NODE NAME   TYPE   NODE HOST          INSTALL DIRECTORY    REFERENCED BY
sj02        SSH    sj02.example.com   /export/glassfish3   pmd-i-sj02, yml-i-sj02
sj01        SSH    sj01.example.com   /export/glassfish3   pmd-i-sj01, yml-i-sj01
Command list-nodes-ssh executed successfully.
```

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-node-dcom(1), create-node-config(1), create-node-ssh(1),
delete-node-config(1), delete-node-dcom(1), delete-node-ssh(1), list-nodes(1),
list-nodes-config(1), list-nodes-dcom(1)
asadmin(1M)
list-password-aliases

lists all password aliases

Synopsis

list-password-aliases [--help]

Description

This subcommand lists all of the password aliases.

Options

--help
-?
Displays the help text for the subcommand.

Examples

Example 1 Listing all password aliases
asadmin> list-password-aliases
jmspassword-alias
Command list-password-aliases executed successfully

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-password-alias(1), update-password-alias(1), create-password-alias(1)
asadmin(1M)
list-persistence-types

lists registered persistence types for HTTP sessions and SFSB instances

Synopsis

list-persistence-types [--help] --type={web|ejb}

Description

The list-persistence-types subcommand lists registered persistence types for HTTP sessions and stateful session bean (SFSB) instances. The built-in persistence types are memory, file, and replicated. The memory type does not apply to SFSB instances.

Other persistence types can be added using the StrategyBuilder class. For more information, see the Oracle GlassFish Server Add-On Component Development Guide.

To set the persistence type for HTTP sessions, use the set subcommand to set the persistence-type attribute. For example:

asadmin> set cl-config.availability-service.web-container-availability.persistence-type=file

To set the persistence type for SFSB instances without availability enabled, use the set subcommand to set the sfsb-persistence-type attribute. For example:

asadmin> set cl-config.availability-service.ejb-container-availability.sfsb-persistence-type=file

To set the persistence type for SFSB instances with availability enabled, use the set subcommand to set the sfsb-ha-persistence-type attribute. For example:

asadmin> set cl-config.availability-service.ejb-container-availability.sfsb-ha-persistence-type=replicated

This subcommand is supported in remote mode only.

Options

--help

-?

Displays the help text for the subcommand.

--type

Specifies the type of sessions for which persistence types are listed. Allowed values are as follows:

- web — Lists persistence types for HTTP sessions.
- ejb — Lists persistence types for SFSB instances.
Examples

Example 1  Listing Persistence Types for HTTP Sessions
This example lists persistence types for HTTP sessions.

```
asadmin> list-persistence-types --type=web
memory
file
replicated
```

Command list-persistence-types executed successfully.

Exit Status

```
0
command executed successfully
1
error in executing the command
```

See Also

```
get(1), set(1), list(1)
asadmin(1M)
```

Oracle GlassFish Server Add-On Component Development Guide
list-probes

lists events for monitoring GlassFish Server

Synopsis

list-probes [--help] [--details={false|true}] [event-id]

Description

The list-probes subcommand lists events for monitoring GlassFish Server.

To provide statistics to GlassFish Server, components define events for the operations that generate these statistics. At runtime, components send these events when performing the operations for which the events are defined. For example, to enable the number of received requests to be monitored, a component sends a "request received" event each time that the component receives a request.

The list-probes subcommand enables you to identify which events provide the statistics that you want to monitor. Use this information in JavaScript programs that you write for monitoring GlassFish Server.

The list-probes subcommand requires that monitoring is enabled for GlassFish Server. If monitoring for GlassFish Server is disabled, no events are listed. For information about how to enable monitoring for GlassFish Server, see the enable-monitoring(1) help page.

By default, the list-probes subcommand lists the signatures of all events that are defined for all installed components of GlassFish Server. The signatures for events that are related to a container are listed only if the container is loaded.

An event signature consists of the event identifier (ID) followed by a list of the event's parameters in the following format:

event-id(param-type param-name[, param-type param-name]...)

The replaceable items in this format are as follows:

**event-id**
The event ID, which uniquely identifies the event.

**param-type**
The type of the event parameter. This type is a Java language primitive, such as int, boolean, or java.lang.String.

**param-name**
The name of the event parameter.

The format of an event ID is as follows:

module-provider:module:submodule:event

The replaceable items in this format are as follows:

**module-provider**
Text that identifies the application that is source of the event. For example, for events from Oracle GlassFish Server, module-provider is glassfish.
module
The name of the module for which the event is defined. A module provides significant functionality of GlassFish Server. Examples of module names are web-container, ejb-container, transaction, and webservices.

submodule
The submodule of module for which the event is defined, for example, web-module.

event-type
The type of the event, for example, webModuleStartedEvent.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--details
Specifies whether detailed information about an event is displayed. Possible values are as follows:

false
Detailed information about the events is not displayed (default). Only the event signature of each event is displayed.

ture
Detailed information about the events is displayed. If the --details option is set to true, an operand is required. Otherwise, an error occurs.

Operands

event-id
The full event ID of the event that is to be listed. If this operand is specified without the --details option, the signature of the specified event is listed. If this operand is omitted, all events are listed.

Examples

Example 1 Listing All Events
This command lists all events for monitoring GlassFish Server. For better readability, some events that would listed by this example are not shown.

asadmin> list-probes
glassfish:jdbc:connection-pool:connectionRequestDequeuedEvent (java.lang.String poolName)
glassfish:jca:connection-pool:connectionsFreedEvent (java.lang.String poolName, int count)
glassfish:transaction:transaction-service:deactivated ()
glassfish:kernel:connections-keep-alive:incrementCountFlushesEvent (java.lang.String listenerName)
glassfish:kernel:file-cache:countInfoMissEvent (java.lang.String fileCacheName)
glassfish:ejb:timers:timerRemovedEvent ()
glassfish:jdbc:connection-pool:decrementNumConnFreeEvent (java.lang.String poolName)
Example 2 Displaying Detailed Information About an Event
This example displays detailed information about the
glassfish:web:web-module:webModuleStartedEvent

asadmin list-probes --details glassfish:web:web-module:webModuleStartedEvent

Information similar to the following is displayed.

Events glassfish:web:web-module:webModuleStartedEvent (5GFP)

NAME
glassfish:web:web-module:webModuleStartedEvent - web module started event

SYNOPSIS
glassfish:web:web-module:webModuleStartedEvent(
java.lang.String appName,
java.lang.String hostName)

DESCRIPTION
This event is sent whenever an application has been started
(for example, as part of its deployment).

PARAMETERS
appName
The name of the web application that has been started.

hostName
The name of the virtual server on which the application
has been deployed.

Java EE 6 Last change: 19 Nov 2009 1

Command list-probes executed successfully.

Exit Status

0 command executed successfully

1 error in executing the command

See Also

run-script(1)
enable-monitoring(1)

asadmin(1M)
list-protocol-filters

lists the existing protocol filters

Synopsis

list-protocol-filters [--help]
[-target server]
protocol-name

Description

The list-protocol-filters subcommand lists the existing protocol filters. This
subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

target
Restricts the listing to protocol filters for a specified target. Valid values are as follows:

server
Lists the protocol filters for the default server instance. This is the default value.

configuration-name
Lists the protocol filters for the specified configuration.

cluster-name
Lists the protocol filters for all server instances in the specified cluster.

instance-name
Lists the protocol filters for the specified server instance.

Operands

protocol-name
The name of the protocol for which to list protocol filters.

Examples

Example 1 Listing Protocol Filters
The following command lists all the protocol filters for the server instance:

asadmin> list-protocol-filters http1
http1-filter
https1-filter
Command list-protocol-filters executed successfully.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-protocol-filter(1), delete-protocol-filter(1)
asadmin(1M)
list-protocol-finders

lists the existing protocol finders

Synopsis

list-protocol-finders [--help]
[--target server]
protocol-name

Description

The list-protocol-finders subcommand lists the existing protocol finders. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

target
Restricts the listing to protocol finders for a specified target. Valid values are as follows:

server
Lists the protocol finders for the default server instance. This is the default value.

configuration-name
Lists the protocol finders for the specified configuration.

cluster-name
Lists the protocol finders for all server instances in the specified cluster.

instance-name
Lists the protocol finders for the specified server instance.

Operands

protocol-name
The name of the protocol for which to list protocol finders.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-protocol-finder(1), delete-protocol-finder(1)
asadmin(1M)
list-protocols

lists the existing protocols

Synopsis

list-protocols [--help] [target]

Description

The list-protocols subcommand lists the existing protocols. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
Restricts the listing to protocols for a specified target. Valid values are as follows:

server
Lists the protocols for the default server instance. This is the default value.

configuration-name
Lists the protocols for the specified configuration.

cluster-name
Lists the protocols for all server instances in the specified cluster.

instance-name
Lists the protocols for the specified server instance.

Examples

Example 1 Listing Protocols
The following command lists all the protocols for the server instance:

asadmin> list-protocols
admin-listener
http-1
http-listener-1
http-listener-2
Command list-protocols executed successfully.
Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-protocol(1), delete-protocol(1)
asadmin(1M)
list-resource-adapter-configs

lists the names of the current resource adapter configurations

Synopsis

list-resource-adapter-configs [--help] [--raname raname] [--verbose {false|true}]

Description

This command lists the configuration information in the domain.xml for the connector module. It lists an entry called resource-adapter-config in the domain.xml file. If the --raname option is specified, only the resource adapter configurations for the specified connector module are listed.

This command is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--raname
Specifies the connector module name.

--verbose
Lists the properties that are configured. Default value is false.

Examples

Example 1 Listing the Resource Adapter Configurations
This example lists the current resource adapter configurations.

asadmin> list-resource-adapter-configs
ra1
ra2
Command list-resource-adapter-configs executed successfully

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-resource-adapter-config(1), delete-resource-adapter-config(1)

asadmin(1M)
list-resource-refs

lists existing resource references

Synopsis

list-resource-refs [--help] [target]

Description

The list-resource-refs subcommand lists all resource references in a cluster or an unclustered server instance. This effectively lists all the resources (for example, JDBC resources) available in the JNDI tree of the specified target.

The target instance or instances in the cluster need not be running or available for this subcommand to succeed.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
The target for which you are listing the resource references. Valid targets are as follows:

server
Lists the resource references for the default server instance and is the default target.

cluster_name
Lists the resource references for every server instance in the cluster.

instance_name
Lists the resource references for the named unclustered server instance.

Examples

Example 1 Listing Resource References for a Cluster
This example lists resource references for the cluster cluster1.

asadmin> list-resource-refs cluster1
jms/Topic
Command list-resource-refs executed successfully.
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-resource-ref(1), delete-resource-ref(1)
asadmin(1M)
list-schedules

lists existing schedules

Synopsis

list-schedules [--help]
[--long={false|true}]
[schedule-name]

Description

The list-schedules subcommand lists existing schedules.
This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--long
-l
Provides detailed information about the listed schedules.
The default value is false.

Operands

schedule-name
Restricts the listing to the named schedule.

Examples

Example 1 Listing Schedule Information
This example uses the --long option to display detailed information about all existing schedules.

asadmin> list-schedules --long
NAME     SECOND  MINUTE  HOUR  DAY OF WEEK  DAY OF MONTH  MONTH  YEAR
daily    0       0       0     *            *             *      *
weekly   0       0       0     Sun          *             *      *
monthly  0       0       0     *            1             *      *
Command list-schedules executed successfully.

Exit Status

0
subcommand executed successfully
1
error in executing the subcommand

See Also

create-schedule(1), delete-schedule(1)
asadmin(1M)
list-secure-admin-internal-users

lists the user names that the GlassFish Server DAS and instances use to authenticate with each other and to authorize admin operations.

Synopsis

list-secure-admin-internal-users
[--help]

Description

The list-secure-admin-internal-users subcommand lists the user names that the GlassFish Server DAS and instances use to authenticate with each other and to authorize admin operations.

Options

--help
-?
Displays the help text for the subcommand.

Examples

Example 1 List the user name for secure admin
The following example lists the user names that the GlassFish Server DAS and instances use to authenticate with each other and to authorize admin operations.

asadmin> list-secure-admin-internal-users

Command list-secure-admin-internal-users executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

enable-secure-admin(1)
enable-secure-admin-internal-user(1)
asadmin(1M)
list-secure-admin-principals

lists the certificates for which GlassFish Server accepts admin requests from clients.

Synopsis

list-secure-admin-principals
[--help]

Description

The list-secure-admin-principals subcommand lists the certificates for which GlassFish Server accepts admin requests from clients.

Options

--help
-?
Displays the help text for the subcommand.

Examples

Example 1 List the certificates
The following example shows how to lists the certificates for which GlassFish Server accepts admin requests from clients.

asadmin> list-secure-admin-principals
Command list-secure-admin-principals executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

enable-secure-admin(1)
enable-secure-admin-principal(1)
asadmin(1M)
list-sub-components

lists EJB or servlet components in a deployed module or module of a deployed application

Synopsis

list-sub-components [--help] [--type type] [--appname appname] [--resources] modulename

Description

The list-sub-components subcommand lists EJB or servlet components in a deployed module or in a module of a deployed application. If a module is not specified, all modules are listed. The --appname option functions only when the specified module is stand-alone. To display a specific module in an application, you must specify the module name with the --appname option.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--type
Specifies the type of component to be listed. The options are ejbs and servlets. If nothing is specified, then all of the components are listed.

--appname
Identifies the name of the application. This option is required when the desired output is the subcomponents of an embedded module of a deployed application.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter or number. It can contain alphanumeric characters plus underscore (_), dash (-), and period (.) characters. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.

--resources
Lists the application-scoped resources for each subcomponent.

Operands

modulename
Specifies the name of the module containing the subcomponent.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter or number. It can contain alphanumeric characters plus underscore (_), dash (-), and period (.) characters. For more information about module and application versions,
see “Module and Application Versions” in Oracle GlassFish Server Application Deployment Guide.

Examples

Example 1 Listing Subcomponents
This example lists the subcomponents of the MEjbApp application within the mejb.jar module.

```sh
asadmin> list-sub-components --appname MEjbApp mejb.jar
MEJBBean <StatelessSessionBean>
Command list-sub-components executed successfully.
```

Exit Status

- 0
  subcommand executed successfully

- 1
  error in executing the subcommand

See Also

enable(1), disable(1), list-components(1)
asadmin(1M)

Oracle GlassFish Server Application Deployment Guide
**list-supported-cipher-suites**

enables administrators to list the cipher suites that are supported and available to a specified GlassFish Server target.

**Synopsis**

```
list-supported-cipher-suites [--help] [--target target]
```

**Description**

The `list-supported-cipher-suites` subcommand enables administrators to list the cipher suites that are supported and available to a specified GlassFish Server target. The cipher suites that may be available in addition to the default SSL/TLS providers that are bundled with GlassFish Server packages will vary depending on the third-party provider.

This subcommand is supported in remote mode only.

**Options**

--help

Displays the help text for the subcommand.

--target

Specifies the target for which you want to list cipher suites. The following values are valid:

server

Lists the cipher suites for the default server instance. This is the default value.

configuration_name

Lists the cipher suites for the specified configuration.

cluster_name

Lists the cipher suites for all server instances in the specified cluster.

instance_name

Lists the cipher suites for a specified server instance.

**Examples**

**Example 1  Listing cipher suites**

The following example shows how to list cipher suites for the default domain.

```
asadmin> list-supported-cipher-suites
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA
```
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
SSL_RSA_WITH_DES_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
SSL_RSA_WITH_NULL_MD5
SSL_RSA_WITH_NULL_SHA
SSL_DH_anon_WITH_RC4_128_MD5
TLS_DH_anon_WITH_AES_128_CBC_SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
SSL_DH_anon_WITH_DES_CBC_SHA
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA

Command list-supported-cipher-suites executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

asadmin(1M)
**list-system-properties**

lists the system properties of the domain, configuration, cluster, or server instance

**Synopsis**

`list-system-properties [--help] [target]`

**Description**

The `list-system-properties` subcommand lists the system properties of a domain, configuration, cluster, or server instance.

This subcommand is supported in remote mode only.

**Options**

`--help`  
`-?`
Displays the help text for the subcommand.

**Operands**

`target`
This restricts the listing to system properties for a specific target. Valid values are:

- `domain`
  Lists the system properties defined for the domain.

- `configuration_name`
  Lists the system properties for the named configuration as well as those the cluster inherits from the domain.

- `cluster_name`
  Lists the system properties defined for the named cluster as well as those the cluster inherits from its configuration and the domain.

- `instance_name`
  Lists the system properties defined for the named server instance as well as those the server inherits from its cluster (if the instance is clustered), its configuration, and the domain.

**Examples**

**Example 1 Listing System Properties**

This example lists the system properties on localhost.

```
asadmin> list-system-properties
http-listener-port=1088
Command list-system-properties executed successfully.
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-system-properties(1), delete-system-property(1)
asadmin(1M)
list-threadpools

lists all the thread pools

Synopsis

list-threadpools [--help] target

Description

The list-threadpools subcommand lists the GlassFish Server thread pools. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target
This operand specifies the target for which you are listing thread pools. This operand is required.

Valid values are as follows:

server
Lists the thread pools for the default GlassFish Server instance server.

configuration-name
Lists the thread pools for the named configuration.

cluster-name
Lists the thread pools for every instance in the cluster.

instance-name
Lists the thread pools for a particular instance.

Examples

Example 1  Listing Thread Pools
This example lists the current thread pools for the default instance server.

asadmin> list-threadpools server
admin-thread-pool
http-thread-pool
thread-pool-1
Command list-threadpools executed successfully.
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-threadpool(1), delete-threadpool(1)
asadmin(1M)
list-timers

lists all of the persistent timers owned by server instance(s)

Synopsis

list-timers [--help] [target]

Description

The list-timers subcommand lists the persistent timers owned by a specific server instance or a cluster of server instances. This command is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

target

The target is either a standalone server instance or a cluster. If the target is the stand-alone instance, then the number of timers owned by the instance is listed. If the target is a cluster, then the number of timers owned by each instance in the cluster is listed. The default target is server, the default server instance.

Examples

Example 1 Listing Current Timers in a Server Instance

This example lists persistent timers in a particular standalone server instance. There is one currently active timer set.

asadmin> list-timers server
1

The list-timers command was executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

migrate-timers(1)

asadmin(1M)
"Using the Timer Service" in *The Java EE 6 Tutorial*

"EJB Timer Service" in *Oracle GlassFish Server Application Development Guide*
list-transports

lists the existing transports

Synopsis

list-transports [--help] [target]

Description

The list-transports subcommand lists the existing transports. This subcommand is supported in remote mode only.

Options

--help
?-?
Displays the help text for the subcommand.

Operands

target
Restricts the listing to transports for a specified target. Valid values are as follows:

server
Lists the transports for the default server instance. This is the default value.

configuration-name
Lists the transports for the specified configuration.

cluster-name
Lists the transports for all server instances in the specified cluster.

instance-name
Lists the transports for the specified server instance.

Examples

Example 1 Listing Transports
The following command lists all the transports for the server instance:

asadmin> list-transports
http1-trans
tcp
Command list-transports executed successfully.

Exit Status

0
command executed successfully
error in executing the command

See Also

create-transport(1), delete-transport(1)
asadmin(1M)
**list-virtual-servers**

lists the existing virtual servers

**Synopsis**

```
list-virtual-servers [--help] [target]
```

**Description**

The `list-virtual-servers` subcommand lists the existing virtual servers. This subcommand is supported in remote mode only.

**Options**

`--help`

Displays the help text for the subcommand.

**Operands**

`target`

Restricts the listing to virtual servers for a specified target. Valid values are as follows:

- **server**
  
  Lists the virtual servers for the default server instance. This is the default value.

- **configuration-name**
  
  Lists the virtual servers for the specified configuration.

- **cluster-name**
  
  Lists the virtual servers for all server instances in the specified cluster.

- **instance-name**
  
  Lists the virtual servers for the specified server instance.

**Examples**

**Example 1  Listing Virtual Servers**

The following command lists all the virtual servers for the server instance:

```
asadmin> list-virtual-servers
server
__asadmin
Command list-virtual-servers executed successfully.
```

**Exit Status**

```
0
command executed successfully
```
1
error in executing the command

See Also

create-virtual-server(1), delete-virtual-server(1)

asadmin(1M)
list-web-context-param

lists servlet context-initialization parameters of a deployed web application or module

Synopsis

list-web-context-param [--help]
[--name=context-param-name] application-name[/module]

Description

The list-web-context-param subcommand lists the servlet context-initialization parameters of one of the following items:

- A deployed web application
- A web module in a deployed Java Platform, Enterprise Edition (Java EE) application

The application must already be deployed. Otherwise, an error occurs.

The list-web-context-param subcommand lists only parameters that have previously been set by using the set-web-context-param(1) subcommand. The subcommand does not list parameters that are set only in the application’s deployment descriptor.

For each parameter, the following information is displayed:

- The name of the parameter
- The value to which the parameter is set
- The value of the --ignoreDescriptorItem option of the set-web-context-param subcommand that was specified when the parameter was set
- The description of the parameter or null if no description was specified when the parameter was set

Options

--help
-?
Displays the help text for the subcommand.

--name

The name of the servlet context-initialization parameter that is to be listed. If this option is omitted, all parameters of the application that have previously been set are listed.

Operands

application-name

The name of the application. This name can be obtained from the Administration Console or by using the list-applications(1) subcommand.

The application must already be deployed. Otherwise, an error occurs.
module
The relative path to the module within the application's enterprise archive (EAR) file. The path to the module is specified in the module element of the application's application.xml file.

module is required only if the servlet context-initialization parameter applies to a web module of a Java EE application. If specified, module must follow application-name, separated by a slash (/).

For example, the application.xml file for the myApp application might specify the following web module:

```xml
<module>
  <web>
    <web-uri>myWebModule.war</web-uri>
  </web>
</module>
```

The module would be specified as the operand of this command as myApp/myWebModule.war.

Examples

Example 1 Listing Servlet Context-Initialization Parameters for a Web Application
This example lists all servlet context-initialization parameters of the web application basic-ezcomp that have been set by using the set-web-context-param subcommand. Because no description was specified when the javax.faces.PROJECT_STAGE parameter was set, null is displayed instead of a description for this parameter.

```
asadmin> list-web-context-param basic-ezcomp
javax.faces.STATE_SAVING_METHOD = client ignoreDescriptorItem=false  //The location where the application's state is preserved
javax.faces.PROJECT_STAGE = null ignoreDescriptorItem=true  //null
```

Command list-web-context-param executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

list-applications(1), set-web-context-param(1), unset-web-context-param(1)
asadmin(1M)
list-web-env-entry

lists environment entries for a deployed web application or module

Synopsis

list-web-env-entry [--help] [--name=env-entry-name] application-name[/module]

Description

The list-web-env-entry subcommand lists the environment entries for one of the following items:

- A deployed web application
- A web module in a deployed Java Platform, Enterprise Edition (Java EE) application

The application must already be deployed. Otherwise, an error occurs.

The list-web-env-entry subcommand lists only entries that have previously been set by using the set-web-env-entry(1) subcommand. The subcommand does not list environment entries that are set only in the application's deployment descriptor.

For each entry, the following information is displayed:

- The name of the entry
- The Java type of the entry
- The value to which the entry is set
- The value of the --ignoreDescriptorItem option of the set-web-env-entry subcommand that was specified when the entry was set
- The description of the entry or null if no description was specified when the entry was set

Options

--help
?-?
Displays the help text for the subcommand.

--name
The name of the environment entry that is to be listed. The name is a JNDI name relative to the java:comp/env context. The name must be unique within a deployment component. If this option is omitted, all environment entries that have previously been set for the application are listed.

Operands

application-name
The name of the application. This name can be obtained from the Administration Console or by using the list-applications(1) subcommand.

The application must already be deployed. Otherwise, an error occurs.
module
The relative path to the module within the application’s enterprise archive (EAR) file. The path to the module is specified in the module element of the application’s application.xml file.

module is required only if the environment entry applies to a web module of a Java EE application. If specified, module must follow application-name, separated by a slash (/). For example, the application.xml file for the myApp application might specify the following web module:

```xml
<module>
  <web>
    <web-uri>myWebModule.war</web-uri>
  </web>
</module>
```

The module would be specified as the operand of this command as myApp/myWebModule.war.

Examples

Example 1 Listing Environment Entries for a Web Application
This example lists all environment entries that have been set for the web application hello by using the set-web-env-entry subcommand. Because no description was specified when the Hello Port environment entry was set, null is displayed instead of a description for this entry.

```bash
asadmin> list-web-env-entry hello
Hello User (java.lang.String) = techscribe ignoreDescriptorItem=false
  //User authentication for Hello application
Hello Port (java.lang.Integer) = null ignoreDescriptorItem=true  //null
```

Command list-web-env-entry executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

list-applications(1), set-web-env-entry(1), unset-web-env-entry(1)
asadmin(1M)
login

logs you into a domain

Synopsis

login [--help]

Description

The purpose of the login subcommand is to ease domain administration by letting you log into a particular domain. If GlassFish Server domains are created on various machines (locally), you can run the asadmin utility from any of these machines and manage domains located elsewhere (remotely). This is especially useful when a particular machine is chosen as an administration client that manages multiple domains and servers.

The login subcommand prompts you for the administrator user name and password. After successful login, the .asadminpass file is created in your home directory. (This is the same file that is modified when you run the create-domain subcommand with the --savelogin option.) The literal host name is stored, and no resolution with the DNS is attempted. If a domain is being administered from other machines, it is sufficient to run the login subcommand once. You do not need to specify the asadmin utility options --user and --passwordfile when you run additional remote subcommands on that domain. After you have logged into a domain, you still need to provide the host and port for any subsequent remote subcommands unless you chose the default values for --host (localhost) and --port (4848) options.

Subsequent use of same subcommand with the same parameters will result in overwriting the contents of the .asadminpass file for the given administration host and port. You can decide to overwrite the file or to reject such a login.

Login information is saved permanently and can be used across multiple domain restarts.

There is no logout subcommand. If you want to log in to another domain, run the login subcommand and specify new values for the asadmin utility options --host and --port.

Options

--help
-?
Displays the help text for the subcommand.

Examples

Example 1 Logging Into a Domain on a Remote Machine

This example logs into a domain located on another machine. Options are specified before the login subcommand.

asadmin --host foo --port 8282 login
Please enter the admin user name>admin
Please enter the admin password>
Trying to authenticate for administration of server at host [foo] and port [8282] ...
Login information relevant to admin user name [admin] for host [foo] and admin port [8282] stored at [/.asadminpass] successfully.
Make sure that this file remains protected. Information stored in this file will be used by asadmin commands to manage associated domain.

Example 2  Logging Into a Domain on the Default Port of Localhost
This example logs into a domain on myhost on the default port. Options are specified before the login subcommand.

```
asadmin --host myhost login
```
Please enter the admin user name>admin
Please enter the admin password>
Trying to authenticate for administration of server at host [myhost] and port [4848] ...
An entry for login exists for host [myhost] and port [4848], probably from an earlier login operation.
Do you want to overwrite this entry (y/n)? y
Login information relevant to admin user name [admin] for host [myhost] and admin port [4848] stored at [./home/joe/.asadminpass] successfully.
Make sure that this file remains protected. Information stored in this file will be used by asadmin commands to manage associated domain.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-domain(1), delete-domain(1)
asadmin(1M)
migrate-timers

moves EJB timers when a clustered instance was stopped or has crashed

Synopsis

migrate-timers [--help] [--target target_server_name]

server_name

Description

The migrate-timers subcommand moves EJB timers to a specified server when a server instance stops or crashes, if automatic timer migration is not enabled in the cluster configuration. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This is the target server instance. If this option is not specified, then DAS will find a server instance or multiple server instances. A migration notification will be sent to the selected server instances.

--destination
This option is deprecated. It works exactly as the --target option does.

Operands

server_name
This is the server instance on which the timers are currently located. This server instance should not be running during the migration process.

Examples

Example 1 Migrating Timers
This example shows how to migrate timers from the server named instance1 to a server named instance2.

asadmin>migrate-timers --target instance2 instance1
This command was successfully executed.

Exit Status
0
command executed successfully

1
error in executing the command
See Also

list-timers(1)
asadmin(1M)
monitor

displays monitoring data for commonly used components and services

Synopsis

monitor [--help]
--type type
[--filename filename]
[--interval interval]
[--filter filter]
instance-name

Description

The monitor subcommand displays statistics for commonly monitored GlassFish Server components and services. The --type option must be used to specify the object for which statistics are to be displayed. Data is displayed continuously in a tabular form, or the data can be displayed at a particular time interval by using the --interval option.

Before a given component or service can be monitored, monitoring must be enabled (set to HIGH or LOW) for the component or service by using the Administration Console, the enable-monitoring subcommand, or the set subcommand.

The monitor subcommand has options for filtering the results and capturing the output in a Comma Separated Values (CSV) file. The output appears in a table format. To view the legend of the table header, type h.

This subcommand is supported in local mode only.

Options

--help
-?
Displays the help text for the subcommand.

--type
The component or service to monitor. This option is required. No default value is defined.

httplistener
For this type, the attribute server.monitoring-service.module-monitoring-levels.http-service must be set to LOW or HIGH.

Displays the following statistics for the HTTP listener service:

ec
The total number errors in the processing of HTTP requests.

mt
The longest response time (in milliseconds) for the processing of a single HTTP request.
The total amount of time (in milliseconds) that the HTTP listener service has spent in processing HTTP requests.

The total number of requests that the HTTP listener service has processed.

**JVM**

For this type, the attribute

```
server.server-config.monitoring-service.module-monitoring-levels.jvm
```

must be set to LOW or HIGH.

Displays the following statistics for the Virtual Machine for the Java platform (Java Virtual Machine or JVM machine):¹

**Uptime**

The number of milliseconds that the JVM machine has been running since it was last started.

**Min**

The initial amount of memory (in bytes) that the JVM machine requests from the operating system for memory management during startup.

**Max**

The maximum amount of memory that can be used for memory management.

**Low**

Retained for compatibility with other releases.

**High**

Retained for compatibility with other releases.

**Count**

The amount of memory (in bytes) that is guaranteed to be available for use by the JVM machine.

**Webmodule**

For this type, the attribute

```
server.server-config.monitoring-service.module-monitoring-levels.web-container
```

must be set to LOW or HIGH.

Displays the following statistics for all deployed web modules:

**Asc**

The number of currently active sessions.

**Ast**

The total number of sessions that are currently active or have been active previously.

**Rst**

The total number of rejected sessions.

**St**

The total number of sessions that have been created.

¹ The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform.
ajlc
The number of currently active JavaServer Pages (JSP) technology pages that
are loaded.

mjlc
The maximum number of JSP technology pages that were active at any time
simultaneously.

tjlc
Total number of JSP technology pages that have been loaded.

aslc
The number of currently active Java servlets that are loaded.

mslc
The maximum number of Java servlets that were active at any time
simultaneously.

tslc
The total number of Java servlets that have been loaded.

--filename
Do not specify this option. This option is retained for compatibility with earlier
releases. If you specify this option, a syntax error does not occur. Instead, the
subcommand runs successfully and displays a warning message that the option is
ignored.

--interval
The interval in seconds before capturing monitoring attributes. The interval must be
greater than 0. The monitoring attributes are displayed on stdout until you type
Control-C or q. The default value is 30.

--filter
Do not specify this option. This option is retained for compatibility with earlier
releases. If you specify this option, a syntax error does not occur. Instead, the
subcommand runs successfully and displays a warning message that the option is
ignored.

Operands

instance-name
The server instance for which to view monitoring data. The default value is server.

Examples

Example 1  Displaying Monitoring Statistics by Interval
This example displays monitoring data for the JVM machine every 2000 seconds.

```
asadmin> monitor --type=jvm --interval 2000 server
JVM Monitoring
UpTime(ms)                          Heap and NonHeap Memory(bytes)
current                   min        max        low        high       count
957843                    29523968   188284928  0          0          60370944
q
Command monitor executed successfully.
```
Example 2  Filtering the Monitoring Data

This example uses the filter option to show http-listener-1 statistics.

```
monitor --type http-listener --filter http-listener-1 server
```

HTTP Listener Monitoring: http-listener-1

<table>
<thead>
<tr>
<th>br</th>
<th>bs</th>
<th>c200</th>
<th>c2xx</th>
<th>c302</th>
<th>c3xx</th>
<th>c400</th>
<th>c401</th>
<th>c403</th>
<th>c404</th>
<th>c4xx</th>
<th>c5xx</th>
<th>coc</th>
<th>co</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>20</td>
<td>20</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To see the legend for the table headings, type h.

<table>
<thead>
<tr>
<th>Legend</th>
</tr>
</thead>
<tbody>
<tr>
<td>* br   = Cumulative value of the Bytes received by each of the Request Processors</td>
</tr>
<tr>
<td>* bs   = Cumulative value of the Bytes sent by each of the Request Processors</td>
</tr>
<tr>
<td>* c200 = Number of responses with a status code equal to 200</td>
</tr>
<tr>
<td>* c2xx = Number of responses with a status code in the 2xx range</td>
</tr>
<tr>
<td>* c302 = Number of responses with a status code equal to 302</td>
</tr>
<tr>
<td>* c304 = Number of responses with a status code equal to 304</td>
</tr>
<tr>
<td>* c3xx = Number of responses with a status code in the 3xx range</td>
</tr>
<tr>
<td>* c400 = Number of responses with a status code equal to 400</td>
</tr>
<tr>
<td>* c401 = Number of responses with a status code equal to 401</td>
</tr>
<tr>
<td>* c403 = Number of responses with a status code equal to 403</td>
</tr>
<tr>
<td>* c404 = Number of responses with a status code equal to 404</td>
</tr>
<tr>
<td>* c4xx = Number of responses with a status code equal to 4xx</td>
</tr>
<tr>
<td>* c504 = Number of responses with a status code equal to 504</td>
</tr>
<tr>
<td>* c5xx = Number of responses with a status code equal to 5xx</td>
</tr>
<tr>
<td>* coc  = Number of open connections</td>
</tr>
<tr>
<td>* co   = Number of responses with a status code outside the 2xx, 3xx, 4xx, and 5xx range</td>
</tr>
<tr>
<td>* ctc  = Number of request processing threads currently in the listener thread pool</td>
</tr>
<tr>
<td>* ctb  = Number of request processing threads currently in use in the listener thread pool serving requests</td>
</tr>
<tr>
<td>* ec   = Number of responses with a status code equal to 400</td>
</tr>
<tr>
<td>* moc  = Maximum number of open connections</td>
</tr>
<tr>
<td>* mst  = Minimum number of request processing threads that will be created at listener startup time and maintained as spare threads above the current thread count</td>
</tr>
</tbody>
</table>
* mt  = Maximum number of request processing threads that are created by the listener *
* mtm = Provides the longest response time for a request - not a cumulative value, but *
       *  the largest response time from among the response times *
* pt  = Cumulative value of the times taken to process each request. The processing *
       *  time is the average of request processing times over the request count *
* rc  = Cumulative number of the requests processed so far *
**********************************************************************************
********

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

  enable-monitoring(1), disable-monitoring(1), set(1)
  monitoring(5ASC)
  asadmin(1M)

"Administering the Monitoring Service" in Oracle GlassFish Server Administration Guide
multimode

allows multiple subcommands to be run while preserving environment settings and remaining in the asadmin utility

Synopsis

multimode [--help] [--file filename] [--printprompt={true|false}] [--encoding encode]

Description

The multimode subcommand processes asadmin subcommands sequentially in a single session. The command-line interface prompts for a subcommand, runs that subcommand, displays the results of that subcommand, and then prompts for the next subcommand. All the asadmin options set in multimode apply to subsequent commands until the multimode session is exited. You exit multimode by typing exit, quit, or Ctrl-D.

You can use the export subcommand to set your environment, or use the unset subcommand to remove environment variables from the multimode environment. You can also provide subcommands by passing a previously prepared list of subcommands from a file or standard input (pipe).

You can invoke multimode from within a multimode session. When you exit the second multimode environment, you return to your original multimode environment.

All the remote asadmin utility options can be supplied when invoking the multimode subcommand. The settings will apply as defaults for all subcommands that are run within the multimode session. For a list of the asadmin utility options, see the asadmin(1M) help page.

Options

--help
-?
Displays the help text for the subcommand.

--file
-f
 Reads the subcommands in the specified file.

--printprompt
Controls printing of the asadmin prompt. By default, this option is set to the same value as the --interactive asadmin utility option. Normally you will not need to specify this option. Default is true.

--encoding
Specifies the character set for the file to be decoded. By default, the system character set is used.
Examples

Example 1 Starting a Multimode Session
This example starts a multimode session where: % is the system prompt.

% asadmin multimode
asadmin>

You can also start a multimode session by typing asadmin without options or subcommands at the system prompt.

Example 2 Running Multiple Commands From a File
This example runs a sequence of subcommands from the commands_file.txt file.

% asadmin multimode --file commands_file.txt

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

export(1), unset(1)
asadmin(1M)
ping-connection-pool

tests if a connection pool is usable

Synopsis

ping-connection-pool [--help] pool_name
[--appname application [--modulename module]]

Description

The ping-connection-pool subcommand tests if an existing JDBC or connector connection pool is usable. For example, if you create a new JDBC connection pool for an application that is expected to be deployed later, the JDBC pool is tested with this subcommand before deploying the application.

Before testing availability of a connection pool, you must create the connection pool with authentication and ensure that the server or database is started.

This subcommand is supported in remote mode only.

Application Scoped Resources

The ping-connection-pool subcommand can target resources that are scoped to a specific application or module, as defined in the glassfish-resources.xml for the GlassFish domain.

- To reference the jndi-name for an application scoped resource, perform the lookup using the java:app prefix.
- To reference the jndi-name for a module scoped resource, perform the lookup using the java:module prefix.

The jndi-name for application-scoped-resources or module-scoped-resources are specified using the format java:app/jdbc/myDataSource or java:module/jdbc/myModuleLevelDataSource. This naming scope is defined in the Java EE 6 Specification (http://download.oracle.com/javaee/6/api/).

Options

--help
-?
Displays the help text for the subcommand.

--appname
Name of the application in which the application scoped resource is defined.

--modulename
Name of the module in which the module scoped resource is defined.

Operands

pool_name
Name of the connection pool to be reinitialized.
Examples

Example 1 Contacting a Connection Pool
This example tests to see if the connection pool named DerbyPool is usable.

asadmin> ping-connection-pool DerbyPool
Command ping-connection-pool executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-connector-connection-pool(1), delete-connector-connection-pool(1),
list-connector-connection-pools(1), create-jdbc-connection-pool(1),
delete-jdbc-connection-pool(1), list-jdbc-connection-pools(1)
asadmin(1M)
ping-node-dcom

tests if a node that is enabled for communication over DCOM is usable

Synopsis

ping-node-dcom [--help]
[--validate={false|true}] node-name

Description

The ping-node-dcom subcommand tests if a node that is enabled for communication over the Distributed Component Object Model (DCOM) remote protocol is usable. The DCOM protocol is available only on Windows systems. This subcommand requires DCOM to be configured on the machine where the domain administration server (DAS) is running and on the machine where the node resides. You may run this command from any machine that can contact the DAS.

If the node is usable, the subcommand displays a confirmation that the subcommand could connect to the node through DCOM. This confirmation includes the name of the host that the node represents.

Optionally, the subcommand can also validate the node to determine if the asadmin(1M) utility can run on the host that the node represents. To validate a node, the subcommand runs the version(1) subcommand. If the node is valid, the subcommand displays the version that the version subcommand returns.

The node that is specified as the operand of this subcommand must be enabled for communication over DCOM. If the node is not enabled for communication over DCOM, an error occurs. To determine whether a node is enabled for communication over DCOM, use the list-nodes(1) subcommand.

Note: To test whether a node that is enabled for communication over secure shell (SSH) is usable, use the ping-node-ssh(1) subcommand.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--validate
Specifies whether the subcommand validates the node.

Possible values are as follows:

true
The node is validated.

false
The node is not validated (default).
Operands

node-name
The name of the node to test. The node must be enabled for communication over DCOM. Otherwise, an error occurs.

Examples

Example 1 Testing if a DCOM-Enabled Node Is Usable
This example tests if the DCOM-enabled node wpmd2 is usable.

asadmin> ping-node-dcom wpmd2
Successfully made DCOM connection to node wpmd2 (wpmd2.example.com)
Command ping-node-dcom executed successfully.

Example 2 Validating a DCOM-Enabled Node
This example validates the DCOM-enabled node adc-node.

asadmin> ping-node-dcom --validate=true adc-node
Successfully made DCOM connection to node adc-node (adc-node.example.com)
GlassFish version found at C:\glassfish3:
Using locally retrieved version string from version class.
Version = GlassFish Server Open Source Edition 3.1.2 (build 18)
Command version executed successfully.
Command ping-node-dcom executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-node-dcom(1), delete-node-dcom(1), list-nodes(1), ping-node-ssh(1),
update-node-dcom(1), validate-dcom(1), version(1)

asadmin(1M)
ping-node-ssh

tests if a node that is enabled for communication over SSH is usable

Synopsis

ping-node-ssh [--help]
[--validate={false|true}] node-name

Description

The ping-node-ssh subcommand tests if a node that is enabled for communication over secure shell (SSH) is usable. This subcommand requires secure shell (SSH) to be configured on the machine where the domain administration server (DAS) is running and on the machine where the node resides. You may run this command from any machine that can contact the DAS.

If the node is usable, the subcommand displays a confirmation that the subcommand could connect to the node through SSH. This confirmation includes the name of the host that the node represents.

Optionally, the subcommand can also validate the node to determine if the asadmin(1M) utility can run on the host that the node represents. To validate a node, the subcommand runs the version(1) subcommand. If the node is valid, the subcommand displays the version that the version subcommand returns.

The node that is specified as the operand of this subcommand must be enabled for communication over SSH. If the node is not enabled for communication over SSH, an error occurs. To determine whether a node is enabled for communication over SSH, use the list-nodes(1) subcommand.

Note: To test whether a node that is enabled for communication over the Distributed Component Object Model (DCOM) remote protocol is usable, use the ping-node-dcom(1) subcommand.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--validate
Specifies whether the subcommand validates the node.
Possible values are as follows:

ture
The node is validated.

false
The node is not validated (default).
Operands

node-name
The name of the node to test. The node must be enabled for communication over SSH. Otherwise, an error occurs.

Examples

Example 1 Testing if an SSH-Enabled Node Is Usable
This example tests if the SSH-enabled node sj03-node is usable.

asadmin> ping-node-ssh sj03-node
Successfully made SSH connection to node sj03-node (sj03.example.com)
Command ping-node-ssh executed successfully.

Example 2 Validating an SSH-Enabled Node
This example validates the SSH-enabled node adc-node.

asadmin> ping-node-ssh --validate=true adc-node
Successfully made SSH connection to node adc-node (adc.example.com)
GlassFish version found at /export/glassfish3:
Using locally retrieved version string from version class.
Version = GlassFish Server Open Source Edition 3.1 (build 40)
Command version executed successfully.
Command ping-node-ssh executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-node-ssh(1), delete-node-ssh(1), list-nodes(1), ping-node-dcom(1),
setup-ssh(1), update-node-ssh(1), version(1)
asadmin(1M)
**recover-transactions**

manually recovers pending transactions

**Synopsis**

```
recover-transactions [--help] [--transactionlogdir transaction_log_dir]
[-target target_server_name] server_name
```

**Description**

The `recover-transactions` subcommand manually recovers pending transactions. For an installation of multiple server instances, you can run the `recover-transactions` subcommand from a surviving server instance to recover transactions after a server failure. To use this subcommand in this way, the following conditions must be met:

- Delegated transaction recovery is disabled.
- Transaction logs are stored on a shared file system or in a database that is accessible to all server instances.

For a stand-alone server, do not use this subcommand to recover transactions after a server failure. For a stand-alone server, the `recover-transactions` subcommand can recover transactions only when a resource fails, but the server is still running. If a stand-alone server fails, only the full startup recovery process can recover transactions that were pending when the server failed.

This subcommand is supported in remote mode only.

**Options**

- **--help**
  -?
  Displays the help text for the subcommand.

- **--transactionlogdir**
  The location of the transaction logs for a server for which transaction recovery is requested. This option applies only if transaction logs are stored on a shared file system.

- **--target**
  The target server that performs the recovery for the server that is specified by the `server_name` operand. The target server should be running.

- **--destination**
  This option is deprecated. It works exactly as the `--target` option does.

**Operands**

- **server_name**
  For a stand-alone server, the value of this operand is typically `server`. Transactions are recovered only if a resource fails, but the server is still running.
For an installation of multiple server instances, the value of this operand is the name of the server for which the recovery is required. The in-flight transactions on this server will be recovered. If this server is running, recovery is performed by the same server. In this situation, the --transactionlogdir and --target options should be omitted. If the server is not running, the --target option is required, and the --transactionlogdir option is also required if transaction logs are stored on a shared file system.

Examples

Example 1  Recovering transactions on a running server
% asadmin recover-transactions server1
Transaction recovered.

Example 2  Recovering transactions for a server that is not running
% asadmin recover-transactions --transactionlogdir /logs/tx --target server1
server2
Transaction recovered.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

freeze-transaction-service(1), unfreeze-transaction-service(1),
rollback-transaction(1)
asadmin(1M)

"Administering Transactions" in Oracle GlassFish Server Administration Guide

"Transactions" in The Java EE 6 Tutorial
redeploy

redeploys the specified component

Synopsis

redeploy [--help]
--name component_name
[--upload={true|false}]
[--retrieve local_dirpath]
[--dbvendorname dbvendorname]
[--createtables={true|false}|--dropandcreatetables={true|false}]
[--uniquetablenames={true|false}]
[--deploymentplan deployment_plan]
[--enabled={true|false}]
[--generatermistubs={false|true}]
[--contextroot context_root]
[--precompilejsp={true|false}]
[--verify={false|true}]
[--virtualservers virtual_servers]
[--availabilityenabled={false|true}]
[--asyncreplication={true|false}]
[--lbenabled={true|false}]
[--keepstate={false|true}]
[--libraries jar_file[,jar_file]*]
[--target target]
[--type pkg-type]
[--properties(name=value)[:name=value]*]
[file_archive|filepath]

Description

The redeploy subcommand redeploys an enterprise application, web application, module based on the Enterprise JavaBeans (EJB) specification (EJB module), connector module, or application client module that is already deployed or already exists. The redeploy subcommand preserves the settings and other options with which the application was originally deployed. The application must already be deployed. Otherwise, an error occurs.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--virtualservers
One or more virtual server IDs. Multiple IDs are separated by commas.

--contextroot
Valid only if the archive is a web module. It is ignored for other archive types; defaults to filename without extension.
--precompilejsp
By default this option does not allow the JSP to be precompiled during deployment. Instead, JSPs are compiled during runtime. Default is false.

--verify
If set to true and the required verifier packages are installed from the Update Tool, the syntax and semantics of the deployment descriptor is verified. Default is false.

--name
Name of the deployable component.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter or number. It can contain alphanumerics characters plus underscore (_), dash (-), and period (.) characters. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.

--upload
Specifies whether the subcommand uploads the file to the DAS. In most situations, this option can be omitted.

Valid values are as follows:

false
The subcommand does not upload the file and attempts to access the file through the specified file name. If the DAS cannot access the file, the subcommand fails.

For example, the DAS might be running as a different user than the administration user and does not have read access to the file. In this situation, the subcommand fails if the --upload option is false.

true
The subcommand uploads the file to the DAS over the network connection.

The default value depends on whether the DAS is on the host where the subcommand is run or is on a remote host.

- If the DAS is on the host where the subcommand is run, the default is false.
- If the DAS is on a remote host, the default is true.

If a directory filepath is specified, this option is ignored.

--retrieve
Retrieves the client stub JAR file from the server machine to the local directory.

--dbvendorname
Specifies the name of the database vendor for which tables are created. Supported values include db2, mssql, oracle, derby, javadb, postgresql, and sybase, case-insensitive. If not specified, the value of the database-vendor-name attribute in glassfish-ejb-jar.xml is used. If no value is specified, a connection is made to the resource specified by the jndi-name subelement of the cmp-resource element in the glassfish-ejb-jar.xml file, and the database vendor name is read. If the connection cannot be established, or if the value is not recognized, SQL-92 compliance is presumed.

--createtables
If specified as true, creates tables at deployment of an application with unmapped CMP beans. If specified as false, tables are not created. If not specified, the value of the
create-tables-at-deploy entry in the cmp-resource element of the
glassfish-ejb-jar.xml file determines whether or not tables are created. No unique
constraints are created for the tables.

--dropandcreatetables
If specified as true when the component is redeployed, the tables created by the
previous deployment are dropped before creating the new tables. Applies to deployed
applications with unmapped CMP beans. If specified as false, tables are neither
dropped nor created. If not specified, the tables are dropped if the
drop-tables-at-undeploy entry in the cmp-resource element of the
glassfish-ejb-jar.xml file is set to true, and the new tables are created if the
create-tables-at-deploy entry in the cmp-resource element of the
glassfish-ejb-jar.xml file is set to true.

--uniquetablenames
Guarantees unique table names for all the beans and results in a hash code added to
the table names. This is useful if you have an application with case-sensitive bean
names. Applies to applications with unmapped CMP beans.

--deploymentplan
Deploys the deployment plan, which is a JAR file that contains GlassFish Server
descriptors. Specify this option when deploying a pure EAR file. A pure EAR file is an
EAR without GlassFish Server descriptors.

--enabled
Allows users to access the application. If set to false, users will not be able to access
the application. This option enables the application on the specified target instance or
cluster. If you deploy to the target domain, this option is ignored, since deploying to
the domain doesn’t deploy to a specific instance or cluster. The default is true.

--generatermistubs
If set to true, static RMI-IIOP stubs are generated and put into the client.jar. If set to
false, the stubs are not generated. Default is false.

--availabilityenabled
This option controls whether high-availability is enabled for web sessions and for
stateful session bean (SFSB) checkpointing and potentially passivation. If set to false
(default) all web session saving and SFSB checkpointing is disabled for the specified
application, web application, or EJB module. If set to true, the specified application or
module is enabled for high-availability. Set this option to true only if high availability
is configured and enabled at higher levels, such as the server and container levels.

--asyncreplication
This option controls whether web session and SFSB states for which high availability is
enabled are first buffered and then replicated using a separate asynchronous thread. If set
to true (default), performance is improved but availability is reduced. If the
instance where states are buffered but not yet replicated fails, the states are lost. If set
to false, performance is reduced but availability is guaranteed. States are not buffered
but immediately transmitted to other instances in the cluster.

--lbenabled
This option controls whether the deployed application is available for load balancing.
The default is true.

--keepstate
This option controls whether web sessions, SFSB instances, and persistently created
EJB timers are retained between redeployments.
The default is false. This option is supported only on the default server instance, named server. It is not supported and ignored for any other target.

Some changes to an application between redeployments prevent this feature from working properly. For example, do not change the set of instance variables in the SFSB bean class.

For web applications, this feature is applicable only if in the glassfish-web-app.xml file the persistence-type attribute of the session-manager element is file.

For stateful session bean instances, the persistence type without high availability is set in the server (the sfsb-persistence-type attribute) and must be set to file, which is the default and recommended value.

If any active web session, SFSB instance, or EJB timer fails to be preserved or restored, none of these will be available when the redeployment is complete. However, the redeployment continues and a warning is logged.

To preserve active state data, GlassFish Server serializes the data and saves it in memory. To restore the data, the class loader of the newly redeployed application deserializes the data that was previously saved.

--libraries
A comma-separated list of library JAR files. Specify the library JAR files by their relative or absolute paths. Specify relative paths relative to domain-dir/lib/applibs. The libraries are made available to the application in the order specified.

--target
Specifies the target to which you are deploying. Valid values are:

server
Deploys the component to the default server instance server and is the default value.

domain
Deploys the component to the domain. If domain is the target for an initial deployment, the application is deployed to the domain, but no server instances or clusters reference the application. If domain is the target for a redeployment, and dynamic reconfiguration is enabled for the clusters or server instances that reference the application, the referencing clusters or server instances automatically get the new version of the application. If redeploying, and dynamic configuration is disabled, the referencing clusters or server instances do not get the new version of the application until the clustered or standalone server instances are restarted.

cluster_name
Deploys the component to every server instance in the cluster.

instance_name
Deploys the component to a particular stand-alone server instance.

--type
The packaging archive type of the component that is being deployed. Possible values are as follows:

osgi
The component is packaged as an OSGi Alliance bundle.

The --type option is optional. If the component is packaged as a regular archive, omit this option.
Optional keyword-value pairs that specify additional properties for the deployment. The available properties are determined by the implementation of the component that is being deployed or redeployed. The `--properties` option and the `--property` option are equivalent. You can use either option regardless of the number of properties that you specify.

You can specify the following properties for a deployment:

**jar-signing-alias**
Specifies the alias for the security certificate with which the application client container JAR file is signed. Java Web Start will not run code that requires elevated permissions unless it resides in a JAR file signed with a certificate that the user’s system trusts. For your convenience, GlassFish Server signs the JAR file automatically using the certificate with this alias from the domain’s keystore. Java Web Start then asks the user whether to trust the code and displays the GlassFish Server certificate information. To sign this JAR file with a different certificate, add the certificate to the domain’s keystore, then use this property. For example, you can use a certificate from a trusted authority, which avoids the Java Web Start prompt, or from your own company, which users know they can trust. Default is `s1as`, the alias for the self-signed certificate created for every domain.

**java-web-start-enabled**
Specifies whether Java Web Start access is permitted for an application client module. Default is true.

**compatibility**
Specifies the GlassFish Server release with which to be backward compatible in terms of JAR visibility requirements for applications. The only allowed value is `v2`, which refers to Sun Java System Application Server version 2 or Sun Java System Application Server version 9.1 or 9.1.1. The Java EE 6 platform specification imposes stricter requirements than Java EE 5 did on which JAR files can be visible to various modules within an EAR file. In particular, application clients must not have access to EJB JAR files or other JAR files in the EAR file unless references use the standard Java SE mechanisms (extensions, for example) or the Java EE library-directory mechanism. Setting this property to `v2` removes these Java EE 6 restrictions.

**keepSessions={false|true}**
Superseded by the `--keepstate` option.

This property can be used to specify whether active sessions of the application that is being redeployed are preserved and then restored when the redeployment is complete. Applies to HTTP sessions in a web container. Default is `false`.

- **false**
  Active sessions of the application are not preserved and restored (default).

- **true**
  Active sessions of the application are preserved and restored.

  If any active session of the application fails to be preserved or restored, none of the sessions will be available when the redeployment is complete. However, the redeployment continues and a warning is logged.

  To preserve active sessions, GlassFish Server serializes the sessions and saves them in memory. To restore the sessions, the class loader of the newly redeployed application deserializes any sessions that were previously saved.
**PreserveAppScopedResources**

If set to `true`, preserves any application-scoped resources and restores them during redeployment. Default is `false`.

Other available properties are determined by the implementation of the component that is being redeployed.

**Operands**

**file_archive|filepath**

The path to the archive that contains the application that is being redeployed. This path can be a relative path or an absolute path.

The archive can be in either of the following formats:

- An archive file, for example, `/export/JEE_apps/hello.war`
  
  If the `--upload` option is set to `true`, this is the path to the deployable file on the local client machine. If the `--upload` option is set to `false`, this is the absolute path to the file on the server machine.

- A directory that contains the exploded format of the deployable archive. This is the absolute path to the directory on the server machine.

  If you specify a directory, the `--upload` option is ignored.

Whether this operand is required depends on how the application was originally deployed:

- If the application was originally deployed from a file, the `archive-path` operand is required. The operand must specify an archive file.

- If the application was originally deployed from a directory, the `archive-path` operand is optional.

  The operand can specify a directory or an archive file.

**Examples**

**Example 1 Redeploying a Web Application From a File**

This example redeploy the web application `hello` from the `hello.war` file in the current working directory. The application was originally deployed from a file. Active sessions of the application are to be preserved and then restored when the redeployment is complete.

```
asadmin> redeploy --name hello --properties keepSessions=true hello.war
Application deployed successfully with name hello.
Command redeploy executed successfully
```

**Example 2 Redeploying a Web Application From a Directory**

This example redeploy the web application `helloworld`. The application was originally deployed from a directory.

```
asadmin> redeploy --name helloworld
Application deployed successfully with name helloworld.
Command redeploy executed successfully
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

deploy(1), undeploy(1), list-components(1)
asadmin(1M)

Oracle GlassFish Server Application Deployment Guide
remove-library

removes one or more library JAR files from GlassFish Server

Synopsis

remove-library [--help] [--type={common|ext|app}] library-name [library-name ... ]

Description

The `remove-library` subcommand removes one or more library JAR files from GlassFish Server.

The `--type` option specifies the library type and the GlassFish Server directory from which the library is removed.

The `library-name` operand is the name of the JAR file that contains the library to be removed. To specify multiple libraries, specify multiple names separated by spaces.

---

**Note:** The library archive file is removed from the DAS. For common and extension libraries, you must restart the DAS so the library removals are picked up by the server runtime. To remove the libraries from other server instances, synchronize the instances with the DAS by restarting them.

This command is not supported on the Windows operating system.

---

This subcommand is supported in remote mode only.

Options

--help

Displays the help text for the subcommand.

--type

Specifies the library type and the GlassFish Server directory from which the library is removed. Valid values are as follows:

- **common**
  
  Removes the library files from the Common class loader directory, `domain-dir/lib`. This is the default.

- **ext**
  
  Removes the library files from the Java optional package directory, `domain-dir/lib/ext`.

- **app**
  
  Removes the library files from the application-specific class loader directory, `domain-dir/lib/applibs`. 
For more information about these directories, see "Class Loaders" in Oracle GlassFish Server Application Development Guide.

Operands

library-name
The names of the JAR files that contain the libraries that are to be removed.

Examples

Example 1 Removing Libraries
This example removes the library in the archive file mylib.jar from the application-specific class loader directory on the default server instance.

asadmin> remove-library --type app mylib.jar
Command remove-library executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

add-library(1), list-libraries(1)
asadmin(1M)
"Class Loaders" in Oracle GlassFish Server Application Development Guide
restart-domain

restarts the DAS of the specified domain

Synopsis

`restart-domain [-help] [--debug ={true|false}] [--domaindir domaindir] [--force={true|false}] [--kill={false|true}] [domain-name]`

Description

The `restart-domain` subcommand stops and then restarts the domain administration server (DAS) of the specified domain. If a domain is not specified, the default domain is assumed. If the domains directory contains two or more domains, the `domain-name` operand must be specified. If the DAS is not already running, the subcommand attempts to start it.

The `restart-domain` subcommand does not exit until the subcommand has verified that the domain has been stopped and restarted.

This subcommand is supported in local or remote mode. If you specify a host name, the subcommand assumes you are operating in remote mode, which means you must correctly authenticate to the remote server. In local mode, you normally do not need to authenticate to the server as long as you are running the subcommand as the same user who started the server.

Options

`--help`  
`-?`  
Displays the help text for the subcommand.

`--debug`  

Possible values are as follows:

`true`  
The domain is restarted with JPDA debugging enabled and the port number for JPDA debugging is displayed.

`false`  
The domain is restarted with JPDA debugging disabled (default).

The default is the current setting of this option for the domain that is being restarted.

`--domaindir`  
The domain root directory, which contains the directory of the domain that is to be restarted. If specified, the path must be accessible in the file system. The default location of the domain root directory is `as-install/domains`.  

Part 3
--force
Specifies whether the domain is forcibly stopped immediately before it is restarted.
Possible values are as follows:

true
The domain is forcibly stopped immediately (default).

false
The subcommand waits until all threads that are associated with the domain are
exited before stopping the domain.

--kill
Specifies whether the domain is killed before it is restarted by using functionality of
the operating system to terminate the domain process.
Possible values are as follows:

false
The domain is not killed. The subcommand uses functionality of the Java platform
to terminate the domain process (default).

true
The domain is killed. The subcommand uses functionality of the operating system
to terminate the domain process.

Operands

domain-name
The name of the domain you want to restart. Default is the name specified during
installation, usually domain1.

Examples

Example 1  Restarting a Domain
This example restarts mydomain4 in the default domains directory.
asadmin> restart-domain mydomain4
Successfully restarted the domain
Command restart-domain executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-domain(1), list-domains(1), start-domain(1), stop-domain(1)

asadmin(1M)

Java Platform Debugger Architecture (JPDA)
(http://java.sun.com/javase/technologies/core/toolsapis/jpda/)
**restart-instance**

restarts a running GlassFish Server instance

**Synopsis**

restart-instance [--help]
[-debug={false|true}] instance-name

**Description**

The `restart-instance` subcommand restarts a running GlassFish Server instance. This subcommand requires secure shell (SSH) to be configured on the machine where the domain administration server (DAS) is running and on the machine where the instance resides.

You may run this subcommand from any machine that can contact the DAS.

The subcommand can restart any GlassFish Server instance, regardless of how the instance was created. For example, this subcommand can restart an instance that was created by using the `create-local-instance(1)` subcommand.

When this subcommand restarts an instance, the DAS synchronizes the instance with changes since the last synchronization as follows:

- For the `config` directory, the DAS synchronizes the instance with all changes.
- For the `applications` directory and `docroot` directory, only a change to a top-level subdirectory causes the DAS to synchronize all files under that subdirectory.

If a file below a top level subdirectory is changed without a change to a file in the top level subdirectory, full synchronization is required. In normal operation, files below the top level subdirectories of these directories are not changed. If an application is deployed and undeployed, full synchronization is not necessary to update the instance with the change.

If different synchronization behavior is required, the instance must be stopped and restarted by using following sequence of subcommands:

1. `stop-instance(1)`
2. `start-instance(1)`

This subcommand is supported in remote mode only.

**Options**

`--help`
`-?`
Displays the help text for the subcommand.

**Note:** SSH is not required if the instance resides on a node of type `CONFIG` that represents the local host. A node of type `CONFIG` is not enabled for communication over SSH.
--debug
Specifies whether the instance is restarted with Java Platform Debugger Architecture (JPDA) (http://java.sun.com/javase/technologies/core/toolsapis/jpda/) debugging enabled.

Possible values are as follows:

  true
  The instance is restarted with JPDA debugging enabled and the port number for JPDA debugging is displayed.

  false
  The instance is restarted with JPDA debugging disabled.

The default is the current setting of this option for the instance that is being restarted.

Operands

  instance-name
  The name of the GlassFish Server instance to restart. If the instance is not running, the subcommand displays a warning message and attempts to start the instance.

Examples

Example 1  Restarting a GlassFish Server Instance

This example restarts the GlassFish Server instance pmdsal.

  asadmin> restart-instance pmdsal
  Instance pmdsal was restarted.

  Command restart-instance executed successfully.

Exit Status

  0
  command executed successfully

  1
  error in executing the command

See Also

  create-instance(1), create-local-instance(1), delete-instance(1), delete-local-instance(1), restart-local-instance(1), setup-ssh(1), start-instance(1), start-local-instance(1), stop-instance(1), stop-local-instance(1)
  asadmin(1M)

  Java Platform Debugger Architecture (JPDA) (http://java.sun.com/javase/technologies/core/toolsapis/jpda/)
**restart-local-instance**

restarts a running GlassFish Server instance on the host where the subcommand is run

**Synopsis**

```
restart-local-instance [--help]
[--nodir nodedir] [--node node]
[--debug={false|true}]
[--force={true|false}] [--kill={false|true}]
[instance-name]
```

**Description**

The `restart-local-instance` subcommand restarts a GlassFish Server instance on the host where the subcommand is run. This subcommand does not require secure shell (SSH) to be configured. You must run this command from the host where the instance resides.

The subcommand can restart any GlassFish Server instance, regardless of how the instance was created. For example, this subcommand can restart an instance that was created by using the `create-instance(1)` subcommand.

The `restart-local-instance` subcommand does not contact the domain administration server (DAS) to determine the node on which the instance resides. To determine the node on which the instance resides, the subcommand searches the directory that contains the node directories. If multiple node directories exist, the node must be specified as an option of the subcommand.

When this subcommand restarts an instance, the DAS synchronizes the instance with changes since the last synchronization as follows:

- For the `config` directory, the DAS synchronizes the instance with all changes.
- For the `applications` directory and `docroot` directory, only a change to a top-level subdirectory causes the DAS to synchronize all files under that subdirectory.

If a file below a top level subdirectory is changed without a change to a file in the top level subdirectory, full synchronization is required. In normal operation, files below the top level subdirectories of these directories are not changed. If an application is deployed and undeployed, full synchronization is not necessary to update the instance with the change.

If different synchronization behavior is required, the instance must be stopped and restarted by using following sequence of subcommands:

1. `stop-local-instance(1)`
2. `start-local-instance(1)`

This subcommand is supported in local mode. However, to synchronize the instance with the DAS, this subcommand must be run in remote mode.
Restart a local instance

Options

--help
-?
Displays the help text for the subcommand.

--nodedir
Specifies the directory that contains the instance's node directory. The instance's files are stored in the instance's node directory. The default is `as-install/nodes`.

--node
Specifies the node on which the instance resides. This option may be omitted only if the directory that the `--nodedir` option specifies contains only one node directory. Otherwise, this option is required.

--debug

Possible values are as follows:

true
The instance is restarted with JPDA debugging enabled and the port number for JPDA debugging is displayed.

false
The instance is restarted with JPDA debugging disabled (default).

The default is the current setting of this option for the instance that is being restarted.

--force
Specifies whether the instance is forcibly stopped immediately before it is restarted.

Possible values are as follows:

true
The instance is forcibly stopped immediately (default).

false
The subcommand waits until all threads that are associated with the instance are exited before stopping the instance.

--kill
Specifies whether the instance is killed before it is restarted by using functionality of the operating system to terminate the instance process.

Possible values are as follows:

false
The instance is not killed. The subcommand uses functionality of the Java platform to terminate the instance process (default).

ture
The instance is killed. The subcommand uses functionality of the operating system to terminate the instance process.
Operands

instance-name
The name of the GlassFish Server instance to restart. If the instance is not running, the subcommand displays a warning message and attempts to start the instance.

Examples

Example 1  Restarting an Instance Locally
This example restarts the instance ymlsa1 in the domain domain1 on the host where the subcommand is run.

asadmin> restart-local-instance --node localhost-domain1 ymlsa1
Command restart-local-instance executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-instance(1), create-local-instance(1), delete-instance(1),
delete-local-instance(1), restart-instance(1), start-instance(1),
start-local-instance(1), stop-instance(1), stop-local-instance(1)
asadmin(1M)

Java Platform Debugger Architecture (JPDA)
(http://java.sun.com/javase/technologies/core/toolsapis/jpda/)
**restore-domain**

restores files from backup

**Synopsis**

```
restore-domain [--help]
[--long={false|true}]  
[--filename backup-filename]
[--domaindir domain-root-dir]
[--backupdir backup-directory]
[--backupconfig backup-config-name]
[--force={false|true}]
[domain-name]
```

**Description**

This command restores files under the domain from a backup directory.

The `restore-domain` command is supported in local mode only.

**Options**

--help

Displays the help text for the subcommand.

--long

-1

Displays detailed information about the restore operation.

The default value is false.

--filename

Specifies the name of the backup file to use as the source.

--domaindir

Specifies the domain root directory, the parent directory of the domain to restore.

The default value is `as-install/domains`.

--backupdir

 Specifies the directory under which the backup file is stored.

The default value is `as-install/domains/domain-dir/backups`. If the domain is not in the default location, the location is `domain-dir/backups`.

--backupconfig

(Supported only in Oracle GlassFish Server.) The name of the domain backup configuration in the backup directory under which the backup file is stored.

--force

Causes the restore operation to continue even when the name of the domain to restore does not match the name of the domain stored in the backup file.

The default value is false.
Operands

**domain-name**
Specifies the name of the domain to restore.

This operand is optional if only one domain exists in the GlassFish Server installation.
If the specified domain name does not match the domain name stored in the backup file, an error occurs unless the --force option is specified.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

backup-domain(1), list-backups(1)
resume-domain

resumes a suspended domain

Synopsis

resume-domain [--help]

Description

The `resume-domain` subcommand resumes a suspended domain. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

suspend-domain(1)
asadmin(1M)
rollback-transaction

rolls back the named transaction

Synopsis

rollback-transaction [--help] [--target target] transaction_id

Description

The rollback-transaction subcommand rolls back the named transaction. Before you can roll back a transaction, you must do the following:

1. Enable monitoring using the set subcommand. For example:
   
   asadmin> set clstr1-config.monitoring-service.module-monitoring-levels.transaction-service=HIGH

2. Use the freeze-transaction-service subcommand to halt in-process transactions.

3. Look up the active transactions using the get subcommand with the --monitor option. For example:
   
   asadmin> get --monitor inst1.server.transaction-service.activeids-current

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target on which you are rolling back the transactions. Valid values are server or any other clustered or stand-alone instance.

Operands

transaction_id
Identifier for the transaction to be rolled back.

Examples

Example 1 Using rollback-transaction command

% asadmin rollback-transaction 0000000000000001_00
Command rollback-transaction executed successfully
Exit Status

0
command executed successfully

1
error in executing the command

See Also

freeze-transaction-service(1), unfreeze-transaction-service(1),
recover-transactions(1)
asadmin(1M)

"Administering Transactions" in Oracle GlassFish Server Administration Guide

"Transactions" in The Java EE 6 Tutorial
**rotate-log**

rotates the log file

**Synopsis**

`rotate-log [--help]`

**Description**

The `rotate-log` subcommand rotates the server log by renaming the file with a timestamp name in the format `server.log_date-and-time`, and creating a new log file. Changes take effect dynamically, that is, server restart is not required.

The size of the log queue is configurable through the `logging.properties` file. Log rotation is based on file size or elapsed time since the last log rotation. In some circumstances, the queue might fill up, especially if the log level is set to `FINEST` and there is heavy activity on the server. In this case, the `rotate-log` subcommand can be used to rotate the server log immediately. This subcommand is also useful in creating scripts for rotating the log at convenient times.

This subcommand is supported in remote mode only.

**Options**

--help
-?
Displays the help text for the subcommand.

--target
The server, cluster, or server instance for which logs will be rotated. If this option is omitted, logs are rotated for the default server.

**Operands**

**target**

Valid values are:

- `server_name` - Default target is `server`. If no target is specified then logs are rotated for the server.

- `cluster_name` - The name of a target cluster.

- `instance_name` - The name of a target server instance.

**Examples**

**Example 1  Rotating the Server Log**

This example rotates the server log.

```
asadmin> rotate-log
Command rotate-log executed successfully.
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

collect-log-files(1), list-log-attributes(1), list-log-levels(1),
set-log-attributes(1), set-log-levels(1)
asadmin(1M)
"Administering the Logging Service" in Oracle GlassFish Server Administration Guide
run-script

runs a script for monitoring GlassFish Server events

Synopsis

run-script [--help] [--upload={true|false}] [--timeout timeout] [--target target] script-file-name

Description

The run-script subcommand runs a script for monitoring GlassFish Server events. To stop the script, type Ctrl-C.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--upload
Specifies whether the subcommand uploads the file to the DAS. In most situations, this option can be omitted.

Valid values are as follows:

false
The subcommand does not upload the file and attempts to access the file through the specified file name. If the DAS cannot access the file, the subcommand fails.

For example, the DAS might be running as a different user than the administration user and does not have read access to the file. In this situation, the subcommand fails if the --upload option is false.

ture
The subcommand uploads the file to the DAS over the network connection.

The default value depends on whether the DAS is on the host where the subcommand is run or is on a remote host.

- If the DAS is on the host where the subcommand is run, the default is false.
- If the DAS is on a remote host, the default is true.

--timeout
Specifies the time in seconds to run the script before stopping it automatically. The default is 0 (zero), which means the script is never stopped automatically.

--target
Specifies the target on which you are running the script. Valid values are:

server
Runs the script on the domain administration server, also called the default server instance, which is named server. This is the default value.
run-script

**cluster-name**
Runs the script on every running server instance in the cluster.

**instance-name**
Runs the script on a particular standalone sever instance.

**Operands**

**script-file-name**
The name, including the absolute or relative path, of the file that contains the script to run. A relative path is relative to the directory from which the run-script subcommand is run.

**Examples**

**Example 1  Running a Script for Monitoring GlassFish Server Events**
This example runs the script `/tools/mon/modulestarted.js` for ten minutes (600 seconds).

```
asadmin> run-script --timeout 600 /tools/mon/modulestarted.js
```

**Example 2  Counting the Number of Started and Stopped Web Modules**
This example is a script that counts the number of standalone web modules and web modules in an enterprise archive (EAR) file that are started and the number of web modules that are stopped.

```javascript
var nModuleStarted=0;
var nModuleStopped=0;

function webModuleStarted(appName, hostName) {
    nModuleStarted = nModuleStarted + 1;
    client.print( `\n js> web-module started event called on ' +
        'virtual_server_id = ' + hostName + ',',
        for application = ' + appName + '
        and count = ' + nModuleStarted);
}

function webModuleStopped(appName, hostName) {
    nModuleStopped = nModuleStopped + 1;
    client.print( `\n js> web-module stopped event called on ' +
        'virtual_server_id= ' + hostName + ',
        for application = ' + appName + '
        and count = ' + nModuleStopped);
}

params = java.lang.reflect.Array.newInstance(java.lang.String, 2);
params[0]="appName";
params[1]="hostName";

scriptContainer.registerListener(
scriptContainer.registerListener(
```
Exit Status

0
command executed successfully

1
error in executing the command

See Also

list-probes(1)
asadmin(1M)
set

sets the values of configurable attributes

Synopsis

set [--help] attribute-name=value

Description

The set subcommand uses dotted names to modify the values of one or more configurable attributes.

Attributes from the monitoring hierarchy are read-only, but configuration attributes can be modified. You can use the list(1) subcommand to display the dotted names that represent individual server components and subsystems. For example, a dotted name might be server.applications.web-module. After you discover the particular component or subsystem, you can then use the get subcommand to access the attributes. For more detailed information on dotted names, see the dotted-names(5ASC) help page.

Note: Characters that have special meaning to the shell or command interpreter, such as * (asterisk), should be quoted or escaped as appropriate to the shell, for example, by enclosing the argument in quotes. In multimode, quotes are needed only for arguments that include spaces, quotes, or backslash.

By modifying attributes, you can enable and disable services, and customize how an existing element functions. An asadmin subcommand is provided to update some elements. For example, update-password-alias. However, to update other elements, you must use the set command. For example, you create a JDBC connection pool by using the create-jdbc-connection-pool subcommand. To change attribute settings later, you use the set command.

Any change made by using the asadmin utility subcommands or the Administration Console are automatically applied to the associated GlassFish Server configuration file.

Options

--help
-?
Displays the help text for the subcommand.

Operands

attribute-name=value
Identifies the full dotted name of the attribute name and its value.
Examples

Example 1 Setting a JDBC Connection Pool Attribute
This example changes the steady pool size of the DerbyPool connection pool to 9.

```
asadmin> set resources.jdbc-connection-pool.DerbyPool.steady-pool-size=9
Command set executed successfully.
```

Example 2 Enabling the Monitoring Service for a Monitorable Object
This example enables monitoring for the JVM.

```
asadmin> set server.monitoring-service.module-monitoring-levels.jvm=HIGH
Command set executed successfully.
```

Example 3 Turning on Automatic Recovery for the Transaction Service
This example turns on automatic recovery for the transaction service.

```
asadmin> set server.transaction-service.automatic-recovery=true
Command set executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

get(1), list(1)
dotted-names(5ASC)
asadmin(1M)

Oracle GlassFish Server Administration Guide
set-log-attributes

sets the logging attributes for one or more loggers

Synopsis

set-log-attributes [--help] [--target=target] attribute-name=attribute-value[.attribute-name=attribute-value]*

Description

The set-log-attributes subcommand sets logging attributes for one or more loggers. The attributes you can set correspond to the attributes that are available in the logging.properties file for the domain. Depending on the attributes you set, a server restart may be necessary.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
The server domain, instance, or cluster for which logger attributes will be set. If this option is omitted, attributes are set for the default server.

Operands

Target
Valid values are:

- server_name - Default target is server. If no target is specified then log attributes are set for the server.
- cluster_name - The name of a target cluster.
- instance_name - The name of a target server instance.

attribute-name
The fully scoped name of the logging attribute. The list-log-attributes subcommand can be used to list the names of all currently defined attributes.

attribute-value
The value to apply to the specified attribute.

The attribute values that can be specified include the following. Refer to the online help and "Administering the Logging Service" in Oracle GlassFish Server Administration Guide for complete explanations of each of these values.

com.sun.enterprise.server.logging.GFFileHandler.alarms
Default is false.
com.sun.enterprise.server.logging.GFFileHandler.file
Default is ${com.sun.aas.instanceRoot}/logs/server.log.

com.sun.enterprise.server.logging.GFFileHandler.flushFrequency
Default is 1.

com.sun.enterprise.server.logging.GFFileHandler.formatter
Default is com.sun.enterprise.server.logging.UniformLogFormatter.

com.sun.enterprise.server.logging.GFFileHandler.logtoConsole
Default is false.

com.sun.enterprise.server.logging.GFFileHandler.maxHistoryFiles
Default is 0.

com.sun.enterprise.server.logging.GFFileHandler.retainErrorsStasticsForHou rs
Default is 0.

com.sun.enterprise.server.logging.GFFileHandler.rotationLimitInBytes
Default is 2000000.

com.sun.enterprise.server.logging.GFFileHandler.rotationTimelimitInMinutes
Default is 0.

com.sun.enterprise.server.logging.SyslogHandler.useSystemLogging
Default is false.

handlers
Default is java.util.logging.ConsoleHandler.

java.util.logging.ConsoleHandler.formatter
Default is com.sun.enterprise.server.logging.UniformLogFormatter.

java.util.logging.FileHandler.count
Default is 1.

java.util.logging.FileHandler.formatter
Default is java.util.logging.XMLFormatter.

java.util.logging.FileHandler.limit
Default is 50000.

java.util.logging.FileHandler.pattern
Default is %h/java%u.log.

log4j.logger.org.hibernate.validator.util.Version
Default is warn.

Examples

Example 1 Setting a the Maximum of Log History Files to Maintain
This example sets to 8 the maximum number of log history files for the server as a whole.

asadmin> set-log-attributes --target=server \ncom.sun.enterprise.server.logging.GFFileHandler.maxHistoryFiles=8 \ncom.sun.enterprise.server.logging.GFFileHandler.maxHistoryFiles logging
attribute set with value 8.
These logging attributes are set for server.
Command set-log-attributes executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

collect-log-files(1), list-log-attributes(1), list-log-levels(1), rotate-log(1),
set-log-levels(1)
asadmin(1M)

"Administering the Logging Service" in Oracle GlassFish Server Administration Guide
set-log-levels

sets the log level for one or more loggers

Synopsis

set-log-levels [--help] logger-name=logger-level[::logger-name=logger-level]* [--target=target]

Description

The set-log-levels subcommand sets the log level for one or more loggers. Changes take effect dynamically. Depending on the log levels you set, a server restart may be necessary.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
The server instance or cluster for which log levels will be set.

Operands

target
The server instance or cluster for which log levels will be set. Valid values are:

- server_name - Default target is server. If no target is specified then log levels are set for the server.
- cluster_name - The name of a target cluster.
- instance_name - The name of a target server instance.

logger-name
The name of the logger. The list-log-levels subcommand can be used to list the names of the current loggers.

logger-level
The level to set for the logger. Log level values are SEVERE, WARNING, INFO, CONFIG, FINE, FINER, and FINEST. The default setting is INFO.

Examples

Example 1 Setting a Log Level for a Logger
This example sets the log level of the web container logger to WARNING.

asadmin> set-log-levels javax.enterprise.system.container.web=WARNING
Command set-log-level executed successfully.
Example 2 Setting the Log Level for Multiple Loggers

This example sets the log level of the web container logger to **FINE** and the log level of the EJB container logger to **SEVERE**:

```bash
asadmin set-log-levels javax.enterprise.system.container.web=FINE:
javax.enterprise.system.container.ejb=SEVERE
```

Command set-log-level executed successfully.

**Exit Status**

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>subcommand executed successfully</td>
</tr>
<tr>
<td>1</td>
<td>error in executing the subcommand</td>
</tr>
</tbody>
</table>

**See Also**

collect-log-files(1), list-log-attributes(1), list-log-levels(1), rotate-log(1), set-log-attributes(1)  
asadmin(1M)  
"Administering the Logging Service" in Oracle GlassFish Server Administration Guide
**setup-ssh**

sets up an SSH key on specified hosts

**Synopsis**

```bash
setup-ssh [--help]
[<--sshport ssh-port] [<--sshuser ssh-user] [<--sshkeyfile ssh-keyfile] [<--sshpublickeyfile ssh-public-keyfile] [<--generatekey={false|true}]
host-list
```

**Description**

The `setup-ssh` subcommand sets up a secure shell (SSH) key on the hosts that are specified as the operand of the subcommand. This key enables GlassFish Server to use public-key authentication for authentication of the user's SSH login on remote hosts.

SSH ensures that GlassFish Server clusters that span multiple hosts can be administered centrally. When a user runs a subcommand for cluster administration that acts on multiple hosts, the subcommand is propagated from the domain administration server (DAS) host to remote hosts. To propagate subcommands that act on a GlassFish Server instance that is not running, or on a node where no instances are running, GlassFish Server uses SSH. SSH provides confidentiality and security for data that is exchanged between the DAS and remote hosts.

Public-key authentication uses an SSH key pair that comprises the following keys:

- A private key, which is stored in a secure location on the DAS host and which may be protected with a passphrase
- The public key, which is stored on all the remote hosts with which the DAS communicates

The subcommand does not require any configuration information from the DAS and does not modify the configuration of the DAS.

This subcommand is supported in local mode only.

**Prerequisites for Using the setup-ssh Subcommand**

To use the `setup-ssh` subcommand, the SSH user must be able to use SSH to log in to remote hosts where SSH is to be set up. Specifically, the following prerequisites must be met:

- The `ssh(1)` client is installed on the DAS host and is accessible through the DAS user's path.
- The `sshd(1M)` daemon is installed and running on all hosts where an SSH key is to be set up.
- The user that the `--sshuser` option specifies has an SSH login on all hosts where an SSH key is to be set up.
- The `ssh-keygen(1)` utility is installed on the DAS host either at the default location or in a location that is defined in the DAS user's path.
On Windows systems, the SSH package for Cygwin (http://www.cygwin.com/) or an MKS Software (http://www.mkssoftware.com/) toolkit that provides SSH is installed.

Behavior of the `setup-ssh` Subcommand

The subcommand sets up SSH connectivity between the DAS host and remote hosts by automating the following tasks:

- **Generating an SSH key pair.** If no SSH key pair exists, the default behavior of the subcommand is to prompt the user to generate an SSH key pair. The SSH key pair is generated without an encryption passphrase. If a passphrase-protected key pair is required, the key pair must be generated manually by using the SSH command `ssh-keygen(1)`.

- **Distributing the public key.** The subcommand appends the content of the public key file to the `user-home/.ssh/authorized_keys` file on each remote host. By default, the subcommand locates the public key file in the `user-home/.ssh` directory on the host where the subcommand is run. If the `user-home/.ssh/authorized_keys` file does not exist on a host, the subcommand creates the file. `user-home` is the user's home directory on a host.

To distribute the public key, authentication of the user's SSH login is required. If the private key is protected by a passphrase, the passphrase is also required. By default, the subcommand prompts the user for the password and, if necessary, the passphrase. To distribute the public key without being prompted, run the subcommand as follows:

- Set the `--interactive` option of the `asadmin(1M)` utility to false.
- Set the `--passwordfile` option of the `asadmin` utility to a file in which the `AS_ADMIN_SSHPASSWORD` entry specifies the SSH user's password for logging in to the specified hosts.
- If a passphrase is required, ensure that the file that `--passwordfile` option of the `asadmin` utility specifies also contains an entry for `AS_ADMIN_SSHKEYPASSPHRASE`.

If public key authentication is already set up on a host, the subcommand informs the user that public key authentication is already set up and does not distribute the key to the host.

Options

`--help`

Displays the help text for the subcommand.

`--sshport`

The port to use for SSH connections to the host where SSH is being set up. The default is 22.

`--sshuser`

The SSH user on the remote host that is to run the process for setting up SSH on that host. The default is the user that is running this subcommand. To ensure that the DAS can read this user's SSH private key file, specify the user that is running the DAS process.
--sshkeyfile
The absolute path to the SSH private key file for user that the --sshuser option specifies. This file is used for authentication to the sshd daemon on the host.

The user that is running this subcommand must be able to reach the path to the key file and read the key file.

The default is a key file in the user’s .ssh directory on the host where the subcommand is run. If multiple key files are found, the subcommand uses the following order of preference:

1. id_rsa
2. id_dsa
3. identity

--sshpublickeyfile
The absolute path to the SSH public key file for user that the --sshuser option specifies. The content of the public key file is appended to the user’s .ssh/authorized_keys file on each host where SSH is being set up. If the .ssh/authorized_keys file does not exist on a host, the subcommand creates the file.

The user that is running this subcommand must be able to reach the path to the key file and read the key file.

The default is a key file in the user’s .ssh directory on the host where the subcommand is run. If multiple key files are found, the subcommand uses the following order of preference:

1. id_rsa.pub
2. id_dsa.pub
3. identity.pub

--generatekey
Specifies whether the subcommand generates the SSH key files without prompting the user.

Possible values are as follows:

true
The subcommand generates the SSH key files without prompting the user.

false
The behavior of the subcommand depends on whether the SSH key files exist:

■ If the SSH key files exist, the subcommand does not generate the files.
■ If the SSH key files do not exist, the behavior of the subcommand depends on the value of the --interactive option of the asadmin utility:
  ■ If the --interactive option is true, the subcommand prompts the user to create the files.
  ■ If the --interactive option is false, the subcommand fails.

This value is the default.

Operands

host-list
A space-separated list of the names of the hosts where an SSH key is to be set up.
Examples

Example 1 Setting Up an SSH Key
This example sets up an SSH key for the user gfuser on the hosts sj03 and sj04. The key file is not generated but is copied from the user's .ssh directory on the host where the subcommand is running.

asadmin> setup-ssh sj03 sj04
Enter SSH password for gfuser@sj03>
Copied keyfile /home/gfuser/.ssh/id_rsa.pub to gfuser@sj03
Successfully connected to gfuser@sj03 using keyfile /home/gfuser/.ssh/id_rsa
Copied keyfile /home/gfuser/.ssh/id_rsa.pub to gfuser@sj04
Successfully connected to gfuser@sj04 using keyfile /home/gfuser/.ssh/id_rsa
Command setup-ssh executed successfully.

Example 2 Generating and Setting Up an SSH Key
This example generates and sets up an SSH key for the user gfuser on the hosts sua01 and sua02.

asadmin> setup-ssh --generatekey=true sua01 sua02
Enter SSH password for gfuser@sua01>
Created directory /home/gfuser/.ssh
/usr/bin/ssh-keygen successfully generated the identification
/home/gfuser/.ssh/id_rsa
Copied keyfile /home/gfuser/.ssh/id_rsa.pub to gfuser@sua01
Successfully connected to gfuser@sua01 using keyfile /home/gfuser/.ssh/id_rsa
Copied keyfile /home/gfuser/.ssh/id_rsa.pub to gfuser@sua02
Successfully connected to gfuser@sua02 using keyfile /home/gfuser/.ssh/id_rsa
Command setup-ssh executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

ssh(1), ssh-keygen(1)
asadmin(1M)
sshd(1M)

Cygwin Information and Installation (http://www.cygwin.com/), MKS Software (http://www.mkssoftware.com/)
set-web-context-param

sets a servlet context-initialization parameter of a deployed web application or module

Synopsis

set-web-context-param [--help] --name=context-param-name
  (--value=value|--ignoredescriptoritem={false|true})
  [--description=description] application-name[/module]

Description

The set-web-context-param subcommand sets a servlet context-initialization parameter of one of the following items:

- A deployed web application
- A web module in a deployed Java Platform, Enterprise Edition (Java EE) application

The application must already be deployed. Otherwise, an error occurs.

This subcommand enables you to change the configuration of a deployed application without the need to modify the application's deployment descriptors and repackage and redeploy the application.

This subcommand is supported in remote mode only.

Options

--help
  -?
  Displays the help text for the subcommand.

--name
  The name of the servlet context-initialization parameter that is to be set.

--value
  The value to which the servlet context-initialization parameter is to be set.

Either the --value option or the --ignoredescriptoritem option must be set.

--ignoredescriptoritem
  Specifies whether the servlet context-initialization parameter is ignored if it is set in the application's deployment descriptor. When a parameter is ignored, the application behaves as if the parameter had never been set in the application's deployment descriptor. The behavior of an application in this situation depends on the application.

The possible values are as follows:

  false
  The value is not ignored (default).

  true
  The value is ignored.

Either the --value option or the --ignoredescriptoritem option must be set.
Note: Do not use the --ignoredescriptoritem option to unset a servlet context-initialization parameter that has previously been set by using the set-web-context-param subcommand. Instead, use the unset-web-context-param(1) subcommand for this purpose.

--description
An optional textual description of the context parameter that is being set.

Operands

application-name
The name of the application. This name can be obtained from the Administration Console or by using the list-applications(1) subcommand.

The application must already be deployed. Otherwise, an error occurs.

module
The relative path to the module within the application's enterprise archive (EAR) file. The path to the module is specified in the module element of the application's application.xml file.

module is required only if the servlet context-initialization parameter applies to a web module of a Java EE application. If specified, module must follow application-name, separated by a slash (/).

For example, the application.xml file for the myApp application might specify the following web module:

    <module>
        <web>
            <web-uri>myWebModule.war</web-uri>
        </web>
    </module>

The module would be specified as the operand of this command as myApp/myWebModule.war.

Examples

Example 1  Setting a Servlet Context-Initialization Parameter for a Web Application
This example sets the servlet context-initialization parameter javax.faces.STATE_SAVING_METHOD of the web application basic-ezcomp to client. The description The location where the application's state is preserved is provided for this parameter.

asadmin> set-web-context-param --name=javax.faces.STATE_SAVING_METHOD --description="The location where the application's state is preserved" --value=client basic-ezcomp

Command set-web-context-param executed successfully.
**Example 2  Ignoring a Servlet Context-Initialization Parameter That Is Defined in a Deployment Descriptor**

This example ignores the servlet context-initialization parameter `javax.faces.PROJECT_STAGE` of the web application `basic-ezcomp`.

```
asadmin> set-web-context-param --name=javax.faces.PROJECT_STAGE --ignoredescriptoritem=true basic-ezcomp
```

Command set-web-context-param executed successfully.

**Exit Status**

0  
command executed successfully

1  
error in executing the command

**See Also**

`list-web-context-param(1), unset-web-context-param(1)`
`asadmin(1M)`
set-web-env-entry

sets an environment entry for a deployed web application or module

Synopsis

set-web-env-entry [--help]
--name=env-entry-name --type=env-entry-type
{--value=value|--ignoredescriptoritem={true|false}}
[--description=description] application-name[/module]

Description

The set-web-env-entry subcommand sets an environment entry for one of the following items:

- A deployed web application
- A web module in a deployed Java Platform, Enterprise Edition (Java EE) application

The application must already be deployed. Otherwise, an error occurs.

An application uses the values of environment entries to customize its behavior or presentation.

This subcommand enables you to change the configuration of a deployed application without the need to modify the application's deployment descriptors and repackage and redeploy the application.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--name
The name of the environment entry that is to be set. The name is a JNDI name relative to the java:comp/env context. The name must be unique within a deployment component.

--type
The fully-qualified Java type of the environment entry value that is expected by the application’s code. This type must be one of the following Java types:

- java.lang.Boolean
- java.lang.Byte
- java.lang.Character
- java.lang.Double
- java.lang.Float
- java.lang.Integer
- java.lang.Long
- java.lang.Short
- java.lang.String

**--value**
The value to which the environment entry is to be set. If the --type is java.lang.Character, the value must be a single character. Otherwise, the value must be a string that is valid for the constructor of the specified type.

Either the --value option or the --ignoredescriptoritem option must be set.

**--ignoredescriptoritem**
Specifies whether the environment entry is ignored if it is set in the application’s deployment descriptor. When an environment entry is ignored, the application behaves as if the entry had never been set in the application's deployment descriptor. The behavior of an application in this situation depends on the application.

The possible values are as follows:

- **false**
  The value is **not** ignored (default).

- **true**
  The value is ignored.

Either the --value option or the --ignoredescriptoritem option must be set.

---

**Note:** Do not use the --ignoredescriptoritem option to unset an environment entry that has previously been set by using the set-web-env-entry subcommand. Instead, use the unset-web-env-entry(1) subcommand for this purpose.

---

**--description**
An optional textual description of the environment entry that is being set.

**Operands**

**application-name**
The name of the application. This name can be obtained from the Administration Console or by using the list-applications(1) subcommand.

The application must already be deployed. Otherwise, an error occurs.

**module**
The relative path to the module within the application’s enterprise archive (EAR) file. The path to the module is specified in the module element of the application’s application.xml file.

**module** is required only if the environment entry applies to a web module of a Java EE application. If specified, **module** must follow **application-name**, separated by a slash (/).

For example, the application.xml file for the myApp application might specify the following web module:

```xml
<module>
  <web>
    <web-uri>myWebModule.war</web-uri>
  </web>
</module>
```
The module would be specified as the operand of this command as `myApp/myWebModule.war`.

**Examples**

**Example 1  Setting an Environment Entry for a Web Application**

This example sets the environment entry `Hello User` of the application `hello` to `techscribe`. The Java type of this entry is `java.lang.String`.

```
asadmin> set-web-env-entry --name="Hello User" --type=java.lang.String --value=techscribe --description="User authentication for Hello application" hello
```

Command `set-web-env-entry` executed successfully.

**Example 2  Ignoring an Environment Entry That Is Defined in a Deployment Descriptor**

This example ignores the environment entry `Hello Port` of the web application `hello`.

```
asadmin> set-web-env-entry --name="Hello Port" --type=java.lang.Integer --ignoredescriptoritem=true hello
```

Command `set-web-env-entry` executed successfully.

**Exit Status**

0
command executed successfully

1
error in executing the command

**See Also**

`list-applications(1), list-web-env-entry(1), unset-web-env-entry(1)`

`asadmin(1M)`
**show-component-status**

displays the status of the deployed component

**Synopsis**

```
show-component-status [--help] [--target target] component-name
```

**Description**

The `show-component-status` subcommand gets the status (either enabled or disabled) of the deployed component.

This subcommand is supported in remote mode only.

**Options**

`--help`

Displays the help text for the subcommand.

`--target`

This option specifies the target on which you are showing the component status. Valid values are:

- `server`
  Shows the component status for the default server instance server and is the default value.

- `domain`
  Shows the component status for the domain.

- `cluster_name`
  Shows the component status for the cluster.

- `instance_name`
  Shows the component status for a clustered or stand-alone server instance.

**Operands**

`component-name`

The name of the component whose status is to be listed.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter or number. It can contain alphanumeric characters plus underscore (_), dash (-), and period (.) characters. To list multiple versions, you can use an asterisk (*) as a wildcard character. For more information about module and application versions, see "Module and Application Versions" in *Oracle GlassFish Server Application Deployment Guide*. 
Examples

**Example 1  Showing the Status of a Component**
This example gets the status of the MEjbApp component.

```
asadmin> show-component-status MEjbApp
Status of MEjbApp is enabled
Command show-component-status executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

list-applications(1), list-sub-components(1)
asadmin(1M)

*Oracle GlassFish Server Application Deployment Guide*
start-cluster

starts a cluster

Synopsis

start-cluster [--help] [--autohadboverride={true|false}] [--verbose={false|true}] cluster-name

Description

The start-cluster subcommand starts a cluster by starting all GlassFish Server instances in the cluster that are not already running. This subcommand requires the Distributed Component Object Model (DCOM) remote protocol or secure shell (SSH) to be configured on the host where the domain administration server (DAS) is running and on all hosts where instances in the cluster reside.

---

**Note:** If all instances reside on the same host as the DAS, DCOM or SSH is not required. You might require to start a cluster in which instances reside on hosts where DCOM or SSH is not configured that are remote from the DAS. In this situation, run the start-local-instance subcommand for each instance from the host where the instance resides.

---

You may run this subcommand from any host that can contact the DAS.

This subcommand is supported in remote mode only.

Options

--help

Displays the help text for the subcommand.

--autohadboverride

Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and displays a warning message that the option is ignored.

--verbose

Specifies whether additional status information is displayed when the cluster is started.

Valid values are as follows:

true
Displays the command to start each instance in the cluster and whether the attempt to start each instance succeeded.

false
Displays no additional status information (default).
Operands

`cluster-name`

The name of the cluster to start.

Examples

**Example 1 Staring a Cluster**

This example starts the cluster `ymlcluster`. Additional status information is displayed when the cluster is started.

```
asadmin> start-cluster --verbose ymlcluster
start-instance yml-i-srl-usca-02
start-instance yml-i-srl-usca-01
```

The command `start-instance` executed successfully for:

```
yml-i-srl-usca-02  yml-i-srl-usca-01
```

Command `start-cluster` executed successfully.

Exit Status

0

command executed successfully

1

error in executing the command

See Also

`create-cluster(1), delete-cluster(1), list-clusters(1), setup-ssh(1),
start-local-instance(1), stop-cluster(1), validate-dcom(1)`

`asadmin(1M)`
Part 3

start-database

starts the Java DB

Synopsis

start-database  [--help]  [--jvmoptions  jvm-options]
[ --dbhost  host]  [ --dbport  port-no]
[ --dbhome  db-file-path]

Description

The start-database subcommand starts the Java DB server that is available for use with GlassFish Server. Java DB is based upon Apache Derby. Use this subcommand only for working with applications deployed to the server.

When you start Java DB server by using the start-database subcommand, the database server is started in Network Server mode. Clients connecting to it must use the Java DB ClientDriver. For details on connecting to the database, refer to the Apache Derby documentation.

When the database server starts, or a client connects to it successfully, the following files are created:

- The derby.log file that contains the database server process log along with its standard output and standard error information
- The database files that contain your schema (for example, database tables)

These files are created at the location that is specified by the --dbhome option. To create the database files at a particular location, you must set the --dbhome option. If the --dbhome option is not specified, the start-database subcommand determines where to create these files as follows:

- If the current working directory contains a file that is named derby.log, the start-database subcommand creates the files in the current working directory.
- Otherwise, the start-database subcommand creates the files in the as-install/databases directory.

The start-database subcommand starts the database process, even if it cannot write to the log file.

This subcommand is supported in local mode only.

Options

--help
-?
Displays the help text for the subcommand.

--jvmoptions
A space-separated list of command-line options that are passed to the Java application launcher when the database is started. By default, no options are passed.

The format of an option depends on whether the option has a name and a value or only a name:
If the option has a name and a value, the format is *option-name=value*.

If the option has only a name, the format is *option-name*. For example, `-Xmx512m`.

---

**--dbhost**
The host name or IP address of the Java DB server process. The default is the IP address 0.0.0.0, which denotes all network interfaces on the host where you run the `start-database` subcommand.

---

**--dbport**
The port number where the Java DB server listens for client connections. This port must be available for the listen socket, otherwise the database server will not start. The default is 1527.

---

**--dbhome**
The absolute path to the directory where the database files and the `derby.log` file are created. If the `--dbhome` option is not specified, the `start-database` subcommand determines where to create these files as follows:

- If the current working directory contains a file that is named `derby.log`, the `start-database` subcommand creates the files in the current working directory.
- Otherwise, the `start-database` subcommand creates the files in the `as-install/databases` directory.

To create the database files at a particular location, you must set the `--dbhome` option.

### Examples

#### Example 1  Starting Java DB

This example starts Java DB on the host `host1` and port 5001.

```
asadmin> start-database --dbhost host1 --dbport 5001 --terse=true
Starting database in the background. Log redirected to /opt/SUNWappserver/databases/derby.log.
```

#### Example 2  Starting Java DB With Options for the Java Application Launcher

This example starts Java DB with the options for setting the minimum heap memory size to 128 megabytes and the maximum heap memory size to 512 megabytes.

```
asadmin> start-database --jvmoptions=-Xms128m -Xmx512m --terse=true
Starting database in the background.
Log redirected to /export/glassfish3/glassfish/databases/derby.log.
```

### Exit Status

The exit status applies to errors in executing the `asadmin` utility. For information on database errors, see the `derby.log` file. This file is located in the directory you specify by using the `--dbhome` option when you run the `start-database` subcommand. If you did not specify `--dbhome`, the value of DERBY_INSTALL defaults to `as-install/javadb`.

```
0
subcommand executed successfully

1
error in executing the subcommand
```
See Also

stop-database(1)
asadmin(1M)

"Administering Database Connectivity" in Oracle GlassFish Server Administration Guide

For more information about the Java application launcher, see the reference page for the operating system that you are using:

■ Oracle Solaris and Linux: java - the Java application launcher
  (http://java.sun.com/javase/6/docs/technotes/tools/solaries/java.html)

■ Windows: java - the Java application launcher
  (http://java.sun.com/javase/6/docs/technotes/tools/windows/java.html)
**start-domain**

starts the DAS of the specified domain

**Synopsis**

```bash
start-domain [--help] [--debug={true|false}] [--domaindir domain-dir] [--upgrade={true|false}] [--verbose={true|false}] [domain-name]
```

**Description**

The `start-domain` subcommand starts the domain administration server (DAS) of the specified domain. If a domain is not specified, the default domain is assumed. If the domains directory contains two or more domains, the `domain-name` operand must be specified.

**Note:** On the Windows platform, processes can bind to the same port. To avoid this problem, do not start multiple domains with the same port number at the same time.

This subcommand is supported in local mode only.

**Note:** In Oracle GlassFish Server, the `start-domain` subcommand prompts for a new admin user password if no password has been set for the admin user. Additionally, the admin user password must not be blank if secure administration is enabled; otherwise, the `start-domain` subcommand fails.

To provide the new admin user password, you can use the `--passwordfile` utility option of the `asadmin(1M)` command after adding the entry `AS_ADMIN_NEWPASSWORD` to the password file.

**Options**

`--help`
-?
Displays the help text for the subcommand.

`--debug`

Possible values are as follows:

`true`
- The instance is started with JPDA debugging enabled and the port number for JPDA debugging is displayed.

`false`
- The instance is started with JPDA debugging disabled (default).
--domainroot
The domain root directory, which contains the directory of the domain that is to be restarted. If specified, the path must be accessible in the file system. The default location of the domain root directory is /etc/other/other

--upgrade
Specifies whether the configuration of the domain administration server (DAS) is upgraded to the current release. Normally, if the subcommand detects that the configuration is from an older release of GlassFish Server, the configuration is upgraded automatically before being started. You should not need to use this option explicitly.

Possible values are as follows:

true
When the domain is started, the configuration is modified to be compatible with this release of GlassFish Server, and the GlassFish Server process stops.

false
The configuration of the DAS is not updated (default).

--verbose
-v
Specifies whether detailed information about the domain is displayed in the console window where the subcommand is run.

Possible values are as follows:

true
Detailed startup messages and log messages about the domain are displayed in the console window where the subcommand is run. If the domain is later restarted by running the restart-domain subcommand from a different console window, messages continue to be displayed in the original console window.

You can kill the GlassFish Server process by typing CTRL-C in the console window.

You can kill the GlassFish Server process and obtain a thread dump for the server by typing one of the following key combinations in the console window:

CTRL-
CTRL-Break on Windows systems

false
Detailed information is not displayed (default).

Operands

domain-name
The unique name of the domain you want to start.

This operand is optional if only one domain exists in the GlassFish Server installation.

Examples

Example 1 Starting a Domain
This example starts mydomain4 in the default domains directory.

asadmin> start-domain mydomain4
Waiting for DAS to start.  ............
Started domain: mydomain4
Domain location: /myhome/glassfishv3/glassfish/domains/mydomain4
Admin port for the domain: 4848
Command start-domain executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-domain(1), delete-domain(1), list-domains(1), restart-domain(1), stop-domain(1)
asadmin(1M)
Java Platform Debugger Architecture (JPDA)
(http://java.sun.com/javase/technologies/core/toolsapis/jpda/)
start-instance

starts a GlassFish Server instance

Synopsis

start-instance [--help]
[--debug={false|true}] [--sync={normal|full|none}]
instance-name

Description

The start-instance subcommand starts a GlassFish Server instance. This subcommand requires the Distributed Component Object Model (DCOM) remote protocol or secure shell (SSH) to be configured on the machine where the domain administration server (DAS) is running and on the machine where the instance resides. You may run this subcommand from any machine that can contact the DAS.

The subcommand can start any GlassFish Server instance, regardless of how the instance was created. For example, this subcommand can start an instance that was created by using the create-local-instance(1) subcommand.

This command is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--debug
Specifies whether the instance is started with Java Platform Debugger Architecture (JPDA) (http://java.sun.com/javase/technologies/core/toolsapis/jpda/) debugging enabled.

Possible values are as follows:

  true
  The instance is started with JPDA debugging enabled and the port number for JPDA debugging is displayed.

  false
  The instance is started with JPDA debugging disabled (default).

--sync
The type of synchronization between the DAS and the instance's files when the instance is started.

Note: DCOM or SSH is not required if the instance resides on a node of type CONFIG that represents the local host. A node of type CONFIG is not enabled for remote communication over DCOM or SSH.
Possible values are as follows:

**none**
The DAS does not synchronize the instance's files with any changes. This type of synchronization minimizes the time that is required to start the instance.

**normal**
The DAS synchronizes the instance with changes since the last synchronization as follows:

- For the `config` directory, the DAS synchronizes the instance with all changes.
- For the `applications` directory and `docroot` directory, only a change to a top-level subdirectory causes the DAS to synchronize all files under that subdirectory.

If a file below a top level subdirectory is changed without a change to a file in the top level subdirectory, full synchronization is required. In normal operation, files below the top level subdirectories of these directories are not changed. If an application is deployed and undeployed, full synchronization is not necessary to update the instance with the change.

This value is the default.

**full**
The DAS synchronizes the instance with all of the instance's files, regardless of whether the files have changed since the last synchronization. This type of synchronization might delay the startup of the instance while the DAS updates all files in the instance's directories.

### Operands

**instance-name**
The name of the GlassFish Server instance to start.

### Examples

**Example 1  Starting a GlassFish Server Instance**
This example starts the GlassFish Server instance `pmdsal`.

```bash
asadmin> start-instance pmdsal
Waiting for the server to start ..........................................
Successfully started the instance: pmdsal
instance Location: /export/glassfish3/glassfish/nodes/localhost/pmdsal
Admin Port: 24848
Command start-local-instance executed successfully.
```

The instance, `pmdsal`, was started on host `localhost`

Command `start-instance executed successfully`.

**Example 2  Starting a GlassFish Server Instance With JPDA Debugging Enabled**
This example starts the GlassFish Server instance `ymlsal` with JPDA debugging enabled.

```bash
asadmin> start-instance --debug=true ymlsal
Waiting for the server to start ..........................................
Successfully started the instance: ymlsal
```
instance Location: /export/glassfish3/glassfish/nodes/localhost/ymlsa1
Admin Port: 24849
Debugging is enabled. The debugging port is: 29010
Command start-local-instance executed successfully.
The instance, ymlsa1, was started on host localhost

Command start-instance executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-instance(1), create-local-instance(1), delete-instance(1),
delete-local-instance(1), setup-ssh(1), start-domain(1), start-local-instance(1),
stop-domain(1), stop-instance(1), stop-local-instance(1), validate-dcom(1)
asadmin(1M)

Java Platform Debugger Architecture (JPDA)
(http://java.sun.com/javase/technologies/core/toolsapis/jpda/)
start-local-instance

starts a GlassFish Server instance on the host where the subcommand is run

Synopsis

start-local-instance [--help]
[--nodedir node-dir] [--node node]
[--debug={false|true}] [--sync={normal|full|none}]
[--verbose={false|true}] [instance-name]

Description

The start-local-instance subcommand starts a GlassFish Server instance on the host where the subcommand is run. This subcommand does not require the Distributed Component Object Model (DCOM) remote protocol or secure shell (SSH) to be configured. You must run this command from the host where the instance resides.

The subcommand can start any GlassFish Server instance, regardless of how the instance was created. For example, this subcommand can start an instance that was created by using the create-instance(1) subcommand.

The start-local-instance subcommand does not contact the domain administration server (DAS) to determine the node on which the instance resides. To determine the node on which the instance resides, the subcommand searches the directory that contains the node directories. If multiple node directories exist, the node must be specified as an option of the subcommand.

This subcommand is supported in local mode. However, to synchronize the instance with the DAS, this subcommand must be run in remote mode.

Options

--help
-? Displays the help text for the subcommand.

--nodedir
Specifies the directory that contains the instance’s node directory. The instance's files are stored in the instance's node directory. The default is as-install/nodes.

--node
Specifies the node on which the instance resides. This option may be omitted only if the directory that the --nodedir option specifies contains only one node directory. Otherwise, this option is required.

--debug
Specifies whether the instance is started with Java Platform Debugger Architecture (JPDA) (http://java.sun.com/javase/technologies/core/toolsapis/jpda/) debugging enabled.

Possible values are as follows:
true
The instance is started with JPDA debugging enabled and the port number for
JPDA debugging is displayed.

false
The instance is started with JPDA debugging disabled (default).

--sync
The type of synchronization between the DAS and the instance's files when the
instance is started.
Possible values are as follows:

none
The DAS does not synchronize the instance's files with any changes. This type of
synchronization minimizes the time that is required to start the instance.

normal
The DAS synchronizes the instance with changes since the last synchronization as
follows:
- For the config directory, the DAS synchronizes the instance with all changes.
- For the applications directory and docroot directory, only a change to a
top-level subdirectory causes the DAS to synchronize all files under that
subdirectory.
  If a file below a top level subdirectory is changed without a change to a file in
the top level subdirectory, full synchronization is required. In normal
operation, files below the top level subdirectories of these directories are not
changed. If an application is deployed and undeployed, full synchronization is
not necessary to update the instance with the change.
This value is the default.

full
The DAS synchronizes the instance with all of the instance's files, regardless of
whether the files have changed since the last synchronization. This type of
synchronization might delay the startup of the instance while the DAS updates all
files in the instance's directories.

Caution: If the DAS is not running or is unreachable from the host
where you are running this subcommand, do not set the --sync
option to full. To perform a full synchronization, the subcommand
removes the instance's cache. If the DAS cannot be contacted to
replace the cache, the subcommand fails and the instance cannot be
restarted until it is resynchronized with the DAS.

--verbose
-v
Specifies whether detailed information about the instance is displayed in the console
window where the subcommand is run.
Possible values are as follows:

true
Detailed startup messages and log messages about the instance are displayed in
the console window where the subcommand is run. If the instance is later
restarted by running the `restart-local-instance(1)` subcommand from a different console window, messages continue to be displayed in the original console window.

You can kill the GlassFish Server process by typing `CTRL-C` in the console window.

You can kill the GlassFish Server process and obtain a thread dump for the server by typing one of the following key combinations in the console window:

- `CTRL-\` on UNIX systems
- `CTRL-Break` on Windows systems

`false`
Detailed information is not displayed (default).

Operands

instance-name
The name of the instance to start.

Examples

Example 1 Starting an Instance Locally
This example starts the instance `yml-i-sj01` on the host where the subcommand is run.

```bash
asadmin> start-local-instance --node sj01 yml-i-sj01
Waiting for the server to start ..................................
Successfully started the instance: yml-i-sj01
instance Location: /export/glassfish3/glassfish/nodes/sj01/yml-i-sj01
Log File: /export/glassfish3/glassfish/nodes/sj01/yml-i-sj01/logs/server.log
Admin Port: 24849
Command start-local-instance executed successfully.
```

Exit Status

0  
command executed successfully

1  
error in executing the command

See Also

create-instance(1), create-local-instance(1), delete-instance(1),
delete-local-instance(1), restart-instance(1), restart-local-instance(1),
start-domain(1), start-instance(1), stop-domain(1), stop-instance(1),
stop-local-instance(1)

asadmin(1M)
Java Platform Debugger Architecture (JPDA)
(http://java.sun.com/javase/technologies/core/toolsapids/jpda/)
**stop-cluster**

stops a GlassFish Server cluster

**Synopsis**

```
stop-cluster [--help]
[--verbose={false|true}]
[--kill={false|true}]
[--autohadboverride={true|false}]
cluster-name
```

**Description**

The `stop-cluster` subcommand stops a GlassFish Server cluster by stopping all running GlassFish Server instances in the cluster.

This subcommand is supported in remote mode only.

**Options**

--help
-?
Displays the help text for the subcommand.

--verbose
Specifies whether additional progress messages about the status of instances in the cluster are displayed while the cluster is being stopped.

Possible values are as follows:

true
Additional progress messages about the status of instances in the cluster are displayed.

false
No messages about the status of instances in the cluster are displayed.

--kill
Specifies whether each instance in the cluster is killed by using functionality of the operating system to terminate the instance process.

Possible values are as follows:

false
No instances are killed. The subcommand uses functionality of the Java platform to terminate each instance process (default).

true
Each instance is killed. The subcommand uses functionality of the operating system to terminate each instance process.

--autohadboverride
Do not specify this option. This option is retained for compatibility with earlier releases. If you specify this option, a syntax error does not occur. Instead, the
subcommand runs successfully and displays a warning message that the option is ignored.

**Operands**

*cluster-name*

The name of the cluster to stop.

**Examples**

**Example 1 Stopping a Cluster**

This example stops the cluster `pmdcluster`. Additional progress messages about the status of instances in the cluster are displayed while the cluster is being stopped.

```bash
asadmin> stop-cluster --verbose pmdcluster
stop-instance pmd-i-sj01
stop-instance pmd-i-sj02
```

The command `stop-instance` executed successfully for: `pmd-i-sj01` `pmd-i-sj02`

Command `stop-cluster` executed successfully.

**Exit Status**

0  
command executed successfully

1  
error in executing the command

**See Also**

`create-cluster(1), delete-cluster(1), list-clusters(1), start-cluster(1)`

`asadmin(1M)`
stop-database

stops the Java DB

Synopsis

    stop-database [--help] [--dbuser db-user]
    [--dbhost host] [--dbport port-no]

Description

The stop-database subcommand stops a process of the Java DB server. Java DB server is available for use with GlassFish Server and is based upon Apache Derby. The database is typically started with the start-database(1) subcommand. A single host can have multiple database server processes running on different ports. The stop-database subcommand stops the database server process for the specified port only.

This subcommand is supported in local mode only.

Options

    --help
    -?
    Displays the help text for the subcommand.

    --dbuser
    The user name of the Java DB user that is to stop the server process. This option is required only when Java DB user authentication is enabled.

    If this option is omitted, no user is specified. By default, Java DB user authentication is disabled, so no user or password is required.

    --dbhost
    The host name or IP address of the Java DB server process. The default is the IP address 0.0.0.0, which denotes all network interfaces on the host where you run the stop-database subcommand.

    --dbport
    The port number where the Java DB server listens for client connections. The default is 1527.

Examples

Example 1 Stopping Java DB

This example stops Java DB on host host1 and port 5001.

    asadmin> stop-database --dbhost host1 --dbport 5001
    Connection obtained for host: host1, port number 5001.
    Shutdown successful.
    Command stop-database executed successfully.
Exit Status

The exit status applies to errors in executing the asadmin utility. For information on database errors, see the derby.log file. This file is located in the directory you specify by using the --dbhome option when you run the start-database subcommand. If you did not specify --dbhome, the value of DERBY_INSTALL defaults to as-install/javadb.

0
command executed successfully

1
error in executing the command

See Also

start-database(1)

asadmin(1M)

"Administering Database Connectivity" in Oracle GlassFish Server Administration Guide
stop-domain

stops the Domain Administration Server of the specified domain

Synopsis

stop-domain [--help] [--domaindir domaindir]
[-force={true|false}] [-kill={false|true}]
[domain-name]

Description

The stop-domain subcommand stops the Domain Administration Server (DAS) of the specified domain. If the domain directory is not specified, the domain in the default domains directory is stopped. If there are two or more domains in the domains directory, the domain-name operand must be specified.

This subcommand is supported in local or remote mode. If you specify a host name, the subcommand assumes you are operating in remote mode, which means you must correctly authenticate to the remote server. In local mode, you normally do not need to authenticate to the server as long as you are running the subcommand as the same user who started the server.

Options

--help
-?
Displays the help text for the subcommand.

--domaindir
Specifies the directory of the domain that is to be stopped. If specified, the path must be accessible in the file system. If not specified, the domain in the default as-install/domains directory is stopped.

--force
Specifies whether the domain is forcibly stopped immediately.
Possible values are as follows:

true
The domain is forcibly stopped immediately (default).

false
The subcommand waits until all threads that are associated with the domain are exited before stopping the domain.

--kill
Specifies whether the domain is killed by using functionality of the operating system to terminate the domain process.
Possible values are as follows:
false
The domain is not killed. The subcommand uses functionality of the Java platform to terminate the domain process (default).

true
The domain is killed. The subcommand uses functionality of the operating system to terminate the domain process.

Operands

domain-name
The name of the domain you want to stop. Default is the name specified during installation, usually domain1.

Examples

Example 1 Stopping a Domain
This example stops the domain named sampleDomain in the default domains directory.

asadmin> stop-domain sampleDomain
Waiting for the domain to stop ..................
Command stop-domain executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-domain(1), list-domains(1), restart-domain(1), start-domain(1)
asadmin(1M)
stop-instance

stops a running GlassFish Server instance

Synopsis

stop-instance [--help]
[--force={false|true}] [--kill={false|true}]
instance-name

Description

The stop-instance subcommand stops a running GlassFish Server instance.

The subcommand can stop any GlassFish Server instance, regardless of how the instance was created. For example, this subcommand can stop an instance that was created by using the create-local-instance(1) subcommand.

This command is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--force
Specifies whether the instance is forcibly stopped immediately.
Possible values are as follows:

true
The instance is forcibly stopped immediately (default).

false
The subcommand waits until all threads that are associated with the instance are exited before stopping the instance.

--kill
Specifies whether the instance is killed by using functionality of the operating system to terminate the instance process.
Possible values are as follows:

false
The instance is not killed. The subcommand uses functionality of the Java platform to terminate the instance process (default).

true
The instance is killed. The subcommand uses functionality of the operating system to terminate the instance process.
Operands

instance-name
This is the name of the GlassFish Server instance to stop.

Examples

This example stops the GlassFish Server instance yml-i-sj01.

Example 1 Stopping a GlassFish Server Instance

asadmin> stop-instance yml-i-sj01
The instance, yml-i-sj01, was stopped.

Command stop-instance executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-instance(1), create-local-instance(1), delete-instance(1), delete-local-instance(1), start-domain(1), start-instance(1), start-local-instance(1), stop-domain(1), stop-local-instance(1)

asadmin(1M)
stop-local-instance

stops a GlassFish Server instance on the machine where the subcommand is run

Synopsis

stop-local-instance [--help]
[--nodedir node-dir] [--node node]
[--force={true|false}] [--kill={false|true}]
[instance-name]

Description

The stop-local-instance subcommand stops a GlassFish Server instance on the machine where the subcommand is run. This subcommand does not require secure shell (SSH) to be configured. You must run this command from the machine where the instance resides.

The subcommand can stop any GlassFish Server instance, regardless of how the instance was created. For example, this subcommand can stop an instance that was created by using the create-instance(1) subcommand.

The stop-local-instance subcommand does not contact the DAS to determine the node on which the instance resides. To determine the node on which the instance resides, the subcommand searches the directory that contains the node directories. If multiple node directories exist, the node must be specified as an option of the subcommand.

This subcommand is supported in local mode.

Options

--help
-?
Displays the help text for the subcommand.

--nodedir
Specifies the directory that contains the instance’s node directory. The instance’s files are stored in the instance’s node directory. The default is as-install/nodes.

--node
Specifies the node on which the instance resides. This option may be omitted only if the directory that the --nodedir option specifies contains only one node directory. Otherwise, this option is required.

--force
Specifies whether the instance is forcibly stopped immediately.
Possible values are as follows:

true
The instance is forcibly stopped immediately (default).
The subcommand waits until all threads that are associated with the instance are exited before stopping the instance.

--kill
Specifies whether the instance is killed by using functionality of the operating system to terminate the instance process.

Possible values are as follows:

false
The instance is not killed. The subcommand uses functionality of the Java platform to terminate the instance process (default).

true
The instance is killed. The subcommand uses functionality of the operating system to terminate the instance process.

Operands

instance-name
The name of the instance to stop.

Examples

Example 1 Stopping an Instance Locally
This example stops the instance yml-i-sj01 on the machine where the subcommand is run.

asadmin> stop-local-instance --node sj01 yml-i-sj01
Waiting for the instance to stop ...
Command stop-local-instance executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-instance(1), create-local-instance(1), delete-instance(1), delete-local-instance(1), start-instance(1), start-local-instance(1), stop-instance(1)
asadmin(1M)
suspending a running domain

Synopsis

suspend-domain [--help]
  [--message reason-message]
  [--timeout timeout-in-seconds]

Description

The suspend-domain subcommand suspends a running domain, bringing the domain to a state where GlassFish Server will not accept any asadmin, Administration Console, or REST API commands that might alter the files in the domain directory.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--message

Specifies an explanatory message to be displayed to administrators who try to perform operations that would be blocked because the domain is suspended.

This option has no default value. When the option is not used, administrators receive a message of the form:

The domain was suspended by a user on date-and-time.

When the option is used, administrators receive a message of the form:

The domain was suspended by a user on date-and-time. Reason: reason-message

--timeout

Specifies a time in seconds to wait for running commands to complete. If the running commands do not complete, the suspend-domain command fails.

The default value is 30.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

resume-domain(1)
asadmin(1M)
undeploy

removes a deployed component

Synopsis

undeploy [--help] [--target target] [--droptables={true|false}] [--cascade={false|true}] name

Description

The undeploy subcommand uninstalls a deployed application or module and removes it from the repository.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--cascade
If set to true, deletes all the connection pools and connector resources associated with the resource adapter being undeployed. If set to false, the undeploy fails if any pools and resources are still associated with the resource adapter. Then, either those pools and resources must be deleted explicitly, or the option must be set to true. If the option is set to false, and if there are no pools and resources still associated with the resource adapter, the resource adapter is undeployed. This option is applicable to connectors (resource adapters) and applications. Default value is false.

--droptables
If set to true, drops the tables that the application created by using CMP beans during deployment. If set to false, tables are not dropped. If not specified, the value of the drop-tables-at-deploy entry in the cmp-resource element of the glassfish-ejb-jar.xml file determines whether or not tables are dropped. Default value is true.

--target
Specifies the target from which you are undeploying. Valid values are:

server
Undeploys the component from the default server instance server and is the default value.

domain
Undeploys the component from the domain.

cluster_name
Undeploys the component from every server instance in the cluster.

instance_name
Undeploys the component from a particular stand-alone server instance.
Operands

**name**
Name of the deployed component.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (:). The version identifier must begin with a letter or number. It can contain alphanumeric characters plus underscore (_), dash (-), and period (.) characters. To delete multiple versions, you can use an asterisk (*) as a wildcard character. For more information about module and application versions, see "Module and Application Versions" in Oracle GlassFish Server Application Deployment Guide.

Examples

**Example 1 Undeploying an Enterprise Application**
This example undeploys an enterprise application named Cart.ear.

```
asadmin> undeploy Cart
Command undeploy executed successfully.
```

**Example 2 Undeploying an Enterprise Bean With Container-Managed Persistence (CMP)**
This example undeploys a CMP bean named myejb and drops the corresponding database tables.

```
asadmin> undeploy --droptables=true myejb
Command undeploy executed successfully.
```

**Example 3 Undeploying a Connector (Resource Adapter)**
This example undeploys the connector module named jdbcra and performs a cascading delete to remove the associated resources and connection pools.

```
asadmin> undeploy --cascade=true jdbcra
Command undeploy executed successfully.
```

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

deploy(1), redeploy(1), list-components(1)

asadmin(1M)

Oracle GlassFish Server Application Deployment Guide
unfreeze-transaction-service

resumes all suspended transactions

Synopsis

unfreeze-transaction-service [--help] [--target target]

Description

The `unfreeze-transaction-service` subcommand restarts the transaction subsystem and resumes all in-flight transactions. Invoke this subcommand on an already frozen transaction subsystem. This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--target
This option specifies the target on which you are unfreezing the transaction service. Valid values are:

server
Unfreezes the transaction service for the default server instance `server` and is the default value.

configuration_name
Unfreezes the transaction service for all server instances that use the named configuration.

cluster_name
Unfreezes the transaction service for every server instance in the cluster.

instance_name
Unfreezes the transaction service for a particular server instance.

Examples

Example 1 Using `unfreeze-transaction-service`

```
% asadmin unfreeze-transaction-service
Command unfreeze-transaction-service executed successfully
```

Exit Status

0
command executed successfully

1
error in executing the command
See Also

freeze-transaction-service(1), rollback-transaction(1), recover-transactions(1)
asadmin(1M)

"Administering Transactions" in Oracle GlassFish Server Administration Guide

"Transactions" in The Java EE 6 Tutorial
uninstall-node

uninstalls GlassFish Server software from specified hosts

Synopsis

uninstall-node [--help]
[--installdir as-install-parent]
[--sshport ssh-port] [--sshuser ssh-user]
[--sshkeyfile ssh-keyfile]
[--force={false|true}]
host-list

Description

The `uninstall-node` subcommand uninstalls GlassFish Server software from the hosts that are specified as the operand of the subcommand. This subcommand requires secure shell (SSH) to be configured on the host where the subcommand is run and on each host where the GlassFish Server software is being uninstalled.

By default, if any node except the predefined node `localhost-domain` resides on any host from which GlassFish Server software is being uninstalled, the subcommand fails. To uninstall the GlassFish Server software from a host on which user-defined nodes reside, set the `--force` option to true. If the `--force` option is true, the subcommand removes the entire content of the parent of the base installation directory.

If a file under the parent of the base installation directory is open, the subcommand fails.

If multiple hosts are specified, the configuration of the following items must be the same on all hosts:

- Parent of the base installation directory for the GlassFish Server software
- SSH port
- SSH user
- SSH key file

The subcommand does not modify the configuration of the domain administration server (DAS).

This subcommand is supported in local mode only.

Options

`--help`
-?
Displays the help text for the subcommand.
--installdir
The absolute path to the parent of the base installation directory where the GlassFish Server software is installed on each host, for example, /export/glassfish3/.

The user that is running this subcommand must have write access to the specified directory. Otherwise, an error occurs.

The specified directory must contain the installation of the GlassFish Server software on the host. Otherwise, an error occurs.

The default is the parent of the base installation directory of the GlassFish Server software on the host where this subcommand is run.

--sshport
The port to use for SSH connections to the host where the GlassFish Server software is to be uninstalled. The default is 22.

--sshuser
The user on the host where the GlassFish Server software is to be uninstalled that is to run the process for connecting through SSH to the host. The default is the user that is running this subcommand. To ensure that the DAS can read this user's SSH private key file, specify the user that is running the DAS process.

--sshkeyfile
The absolute path to the SSH private key file for user that the --sshuser option specifies. This file is used for authentication to the sshd daemon on the host.

The user that is running this subcommand must be able to reach the path to the key file and read the key file.

The default is a key file in the user's .ssh directory. If multiple key files are found, the subcommand uses the following order of preference:

1. id_rsa
2. id_dsa
3. identity

--force
Specifies whether the subcommand uninstalls the GlassFish Server software from a host even if a user-defined node resides on the host. Possible values are as follows:

false
If a user-defined node resides on a host, the software is not uninstalled and the subcommand fails (default).

If the --force option is false, the subcommand removes only the GlassFish Server software files. Other content if the parent of the base installation directory, such as configuration files, are not removed.

true
The subcommand uninstalls the GlassFish Server software from the host even if a user-defined node resides on the host.

If the --force option is true, the subcommand removes the entire content of the parent of the base installation directory.
Operands

**host-list**
A space-separated list of the names of the hosts from which the GlassFish Server software is to be uninstalled.

Examples

**Example 1 Uninstalling GlassFish Server Software From the Default Location**
This example uninstalls GlassFish Server software on the hosts sj03.example.com and sj04.example.com from the default location.

```
asadmin> uninstall-node sj03 sj04
Successfully connected to gfuser@sj03.example.com using keyfile /home/gfuser/.ssh/id_rsa
Successfully connected to gfuser@sj04.example.com using keyfile /home/gfuser/.ssh/id_rsa
Command uninstall-node executed successfully.
```

**Example 2 Forcibly Uninstalling GlassFish Server Software**
This example uninstalls GlassFish Server software on the host sj02.example.com. The software is uninstalled even if a user-defined node resides on the host. The entire content of the /export/glassfish3 directory is removed.

Some lines of output are omitted from this example for readability.

```
asadmin> uninstall-node --force --installdir /export/glassfish3 sj02.example.com
Successfully connected to gfuser@sj02.example.com using keyfile /home/gfuser/.ssh/id_rsa
...
Force removing directory /export/glassfish3/glassfish/lib/appclient
Force removing directory /export/glassfish3/glassfish/lib
Force removing directory /export/glassfish3/glassfish
Command uninstall-node executed successfully.
```

Exit Status

```
0
command executed successfully
1
error in executing the command
```

See Also

install-node(1), install-node-dcom(1), install-node-ssh(1),
uninstall-node-dcom(1), uninstall-node-ssh(1)
asadmin(1M)
uninstall-node-dcom

uninstalls GlassFish Server software from specified DCOM-enabled hosts

Synopsis

uninstall-node-dcom [--help]
[--installdir as-install-parent]
[--windowsuser windows-user] [--windowsdomain windows-domain]
[--force={false|true}]
host-list

Description

The `uninstall-node-dcom` subcommand uninstalls GlassFish Server software from the hosts that are specified as the operand of the subcommand. This subcommand requires the Distributed Component Object Model (DCOM) remote protocol to be configured on the host where the subcommand is run and on each host where the GlassFish Server software is being uninstalled.

By default, if any node except the predefined node `localhost-domain` resides on any host from which GlassFish Server software is being uninstalled, the subcommand fails. To uninstall the GlassFish Server software from a host on which user-defined nodes reside, set the `--force` option to true. If the `--force` option is true, the subcommand removes the entire content of the parent of the base installation directory.

If a file under the parent of the base installation directory is open, the subcommand fails.

If multiple hosts are specified, the configuration of the following items must be the same on all hosts:

- Parent of the base installation directory for the GlassFish Server software
- Windows user
- Windows domain

The subcommand does not modify the configuration of the domain administration server (DAS).

This subcommand is supported in local mode only.

Options

`--help`
-?
Displays the help text for the subcommand.
--installdir
The absolute path to the parent of the base installation directory where the GlassFish Server software is installed on each host, for example, C:\glassfish3.

The user that is running this subcommand must have write access to the specified directory. Otherwise, an error occurs.

The specified directory must contain the installation of the GlassFish Server software on the host. Otherwise, an error occurs.

The default is the parent of the base installation directory of the GlassFish Server software on the host where this subcommand is run.

--windowsuser
-w
The user on this node’s host that is to run the process for connecting to the host through DCOM. The default is the user that is running the DAS process.

--windowsdomain
-d
The name of the Windows domain that contains the user that the --windowsuser option specifies. The default is the name of the host on which the subcommand is run.

--force
Specifies whether the subcommand uninstalls the GlassFish Server software from a host even if a user-defined node resides on the host. Possible values are as follows:

false
If a user-defined node resides on a host, the software is not uninstalled and the subcommand fails (default).

If the --force option is false, the subcommand removes only the GlassFish Server software files. Other content if the parent of the base installation directory, such as configuration files, are not removed.

ture
The subcommand uninstalls the GlassFish Server software from the host even if a user-defined node resides on the host.

If the --force option is true, the subcommand removes the entire content of the parent of the base installation directory.

Operands

host-list
A space-separated list of the names of the hosts from which the GlassFish Server software is to be uninstalled.

Examples

Example 1 Uninstalling GlassFish Server Software From the Default Location
This example uninstalls GlassFish Server software on the hosts wpmdl1.example.com and wpmdl2.example.com from the default location.

asadmin> uninstall-node-dcom wpmdl1 wpmdl2
Command uninstall-node-dcom executed successfully.
Example 2  Forcibly Uninstalling GlassFish Server Software

This example uninstalls GlassFish Server software on the host xkyd.example.com.

The software is uninstalled even if a user-defined node resides on the host. The entire content of the C:\glassfish3 directory is removed.

asadmin>  uninstall-node-dcom --force --installdir C:\glassfish3 xkyd.example.com
Command uninstall-node-dcom executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

install-node-dcom(1), uninstall-node-ssh(1)
asadmin(1M)
uninstall-node-ssh

uninstalls GlassFish Server software from specified SSH-enabled hosts

Synopsis

uninstall-node-ssh [--help]
[--installdir as-install-parent]
[--sshport ssh-port] [--sshuser ssh-user]
[--sshkeyfile ssh-keyfile]
[--force={false|true}]
host-list

Description

The `uninstall-node-ssh` subcommand uninstalls GlassFish Server software from the hosts that are specified as the operand of the subcommand. This subcommand requires secure shell (SSH) to be configured on the host where the subcommand is run and on each host where the GlassFish Server software is being uninstalled.

By default, if any node except the predefined node `localhost-domain` resides on any host from which GlassFish Server software is being uninstalled, the subcommand fails. To uninstall the GlassFish Server software from a host on which user-defined nodes reside, set the `--force` option to true. If the `--force` option is true, the subcommand removes the entire content of the parent of the base installation directory.

If a file under the parent of the base installation directory is open, the subcommand fails.

If multiple hosts are specified, the configuration of the following items must be the same on all hosts:

- Parent of the base installation directory for the GlassFish Server software
- SSH port
- SSH user
- SSH key file

The subcommand does not modify the configuration of the domain administration server (DAS).

This subcommand is supported in local mode only.

Options

`--help`

Displays the help text for the subcommand.
--installdir
The absolute path to the parent of the base installation directory where the GlassFish Server software is installed on each host, for example, /export/glassfish3/.

The user that is running this subcommand must have write access to the specified directory. Otherwise, an error occurs.

The specified directory must contain the installation of the GlassFish Server software on the host. Otherwise, an error occurs.

The default is the parent of the base installation directory of the GlassFish Server software on the host where this subcommand is run.

--sshport
The port to use for SSH connections to the host where the GlassFish Server software is to be uninstalled. The default is 22.

--sshuser
The user on the host where the GlassFish Server software is to be uninstalled that is to run the process for connecting through SSH to the host. The default is the user that is running this subcommand. To ensure that the DAS can read this user's SSH private key file, specify the user that is running the DAS process.

--sshkeyfile
The absolute path to the SSH private key file for user that the --sshuser option specifies. This file is used for authentication to the sshd daemon on the host.

The user that is running this subcommand must be able to reach the path to the key file and read the key file.

The default is a key file in the user's .ssh directory. If multiple key files are found, the subcommand uses the following order of preference:

1. id_rsa
2. id_dsa
3. identity

--force
Specifies whether the subcommand uninstalls the GlassFish Server software from a host even if a user-defined node resides on the host. Possible values are as follows:

false
If a user-defined node resides on a host, the software is not uninstalled and the subcommand fails (default).

If the --force option is false, the subcommand removes only the GlassFish Server software files. Other content if the parent of the base installation directory, such as configuration files, are not removed.

true
The subcommand uninstalls the GlassFish Server software from the host even if a user-defined node resides on the host.

If the --force option is true, the subcommand removes the entire content of the parent of the base installation directory.
Operands

`host-list`  
A space-separated list of the names of the hosts from which the GlassFish Server software is to be uninstalled.

Examples

Example 1 Uninstalling GlassFish Server Software From the Default Location  
This example uninstalls GlassFish Server software on the hosts sj03.example.com and sj04.example.com from the default location.

```
asadmin> uninstall-node-ssh sj03 sj04
Successfully connected to gfuser@sj03.example.com using keyfile /home/gfuser/.ssh/id_rsa
Successfully connected to gfuser@sj04.example.com using keyfile /home/gfuser/.ssh/id_rsa
Command uninstall-node-ssh executed successfully.
```

Example 2 Forcibly Uninstalling GlassFish Server Software  
This example uninstalls GlassFish Server software on the host sj02.example.com.  
The software is uninstalled even if a user-defined node resides on the host. The entire content of the `/export/glassfish3` directory is removed.  
Some lines of output are omitted from this example for readability.

```
asadmin> uninstall-node-ssh --force --installdir /export/glassfish3 sj02.example.com
Successfully connected to gfuser@sj02.example.com using keyfile /home/gfuser/.ssh/id_rsa
...
Force removing directory /export/glassfish3/glassfish/lib/appclient
Force removing directory /export/glassfish3/glassfish/lib
Force removing directory /export/glassfish3/glassfish
Command uninstall-node-ssh executed successfully.
```

Exit Status

0  
command executed successfully

1  
error in executing the command

See Also

install-node(1), install-node-ssh(1), uninstall-node(1), uninstall-node-dcom(1)  
asadmin(1M)
unset

removes one or more variables from the multimode environment

Synopsis

unset [--help] variable-list

Description

The `unset` subcommand removes one or more environment variables that are set for the multimode environment. After removal, the variables and their associated values no longer apply to the multimode environment.

To list the environment variables that are set, use the `export` subcommand without options. If the `export` subcommand lists no environment variables, no environment variables are set.

This subcommand is supported in local mode only.

Options

`--help`

Displays the help text for the subcommand.

Operands

`variable-list`

A space-separated list of the environment variables to remove.

Examples

**Example 1  Listing the Environment Variables That Are Set**

This example uses the `export` subcommand to list the environment variables that have been set.

```bash
asadmin> export
AS_ADMIN_USER = admin
AS_ADMIN_HOST = bluestar
AS_ADMIN_PREFIX = server1.jms-service
AS_ADMIN_PORT = 8000
Command export executed successfully
```

**Example 2  Removing an Environment Variable**

This example removes the `AS_ADMIN_PREFIX` environment variable.

```bash
asadmin> unset AS_ADMIN_PREFIX
Command unset executed successfully
```
Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

export(1), multimode(1)

asadmin(1M)
**unset-web-context-param**

unsets a servlet context-initialization parameter of a deployed web application or module

**Synopsis**

```
unset-web-context-param [--help]
--name=context-param-name application-name[/module]
```

**Description**

The `unset-web-context-param` subcommand unsets a servlet context-initialization parameter of one of the following items:

- A deployed web application
- A web module in a deployed Java Platform, Enterprise Edition (Java EE) application

When a parameter is unset, its value reverts to the value, if any, that is set in the application's deployment descriptor.

The application must already be deployed. Otherwise, an error occurs.

The parameter must have previously been set by using the `set-web-context-param` subcommand. Otherwise, an error occurs.

---

**Note:** Do not use the `unset-web-context-param` subcommand to change the value of a servlet context-initialization parameter that is set in an application's deployment descriptor. Instead, use the `set-web-context-param(1)` subcommand for this purpose.

---

This subcommand enables you to change the configuration of a deployed application without the need to modify the application's deployment descriptors and repackage and redeploy the application.

This subcommand is supported in remote mode only.

**Options**

```
--help
-?
```

Displays the help text for the subcommand.

```
--name
```

The name of the servlet context-initialization parameter that is to be unset. This parameter must have previously been set by using the `set-web-context-param` subcommand. Otherwise, an error occurs.
Operands

(application-name)
The name of the application. This name can be obtained from the Administration Console or by using the list-applications(1) subcommand.

The application must already be deployed. Otherwise, an error occurs.

(module)
The relative path to the module within the application's enterprise archive (EAR) file. The path to the module is specified in the module element of the application's application.xml file.

(module) is required only if the servlet context-initialization parameter applies to a web module of a Java EE application. If specified, module must follow application-name, separated by a slash (/).

For example, the application.xml file for the myApp application might specify the following web module:

```xml
<module>
  <web>
    <web-uri>myWebModule.war</web-uri>
  </web>
</module>
```

The module would be specified as the operand of this command as myApp/myWebModule.war.

Examples

Example 1 Unsetting a Servlet Context-Initialization Parameter for a Web Application

This example unsets the servlet context-initialization parameter javax.faces.STATE_SAVING_METHOD of the web application basic-ezcomp. The parameter reverts to the value, if any, that is defined in the application's deployment descriptor.

asadmin> unset-web-context-param
--name=javax.faces.STATE_SAVING_METHOD basic-ezcomp

Command unset-web-context-param executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

list-applications(1), list-web-context-param(1), set-web-context-param(1)
asadmin(1M)
**unset-web-env-entry**

unsets an environment entry for a deployed web application or module

**Synopsis**

```
unset-web-env-entry [--help] --name=env-entry-name application-name[/module]
```

**Description**

The `unset-web-env-entry` subcommand unsets an environment entry for one of the following items:

- A deployed web application
- A web module in a deployed Java Platform, Enterprise Edition (Java EE) application

When an entry is unset, its value reverts to the value, if any, that is set in the application’s deployment descriptor.

The application must already be deployed. Otherwise, an error occurs.

The entry must have previously been set by using the `set-web-env-entry(1)` subcommand. Otherwise, an error occurs.

---

**Note:** Do not use the `unset-web-env-entry` subcommand to change the value of an environment entry that is set in an application's deployment descriptor. Instead, use the `set-web-env-entry` subcommand for this purpose.

---

This subcommand enables you to change the configuration of a deployed application without the need to modify the application's deployment descriptors and repackaging and redeploying the application.

This subcommand is supported in remote mode only.

**Options**

```
--help
-?
```

Displays the help text for the subcommand.

```
--name
```

The name of the environment entry that is to be unset. The name is a JNDI name relative to the `java:comp/env` context. The name must be unique within a deployment component. This entry must have previously been set by using the `set-web-env-entry` subcommand. Otherwise, an error occurs.
Operands

application-name
The name of the application. This name can be obtained from the Administration Console or by using the list-applications(1) subcommand.

The application must already be deployed. Otherwise, an error occurs.

module
The relative path to the module within the application's enterprise archive (EAR) file. The path to the module is specified in the module element of the application's application.xml file.

module is required only if the environment entry applies to a web module of a Java EE application. If specified, module must follow application-name, separated by a slash (/).

For example, the application.xml file for the myApp application might specify the following web module:

```xml
<module>
  <web>
    <web-uri>myWebModule.war</web-uri>
  </web>
</module>
```

The module would be specified as the operand of this command as myApp/myWebModule.war.

Examples

Example 1 Unsetting an Environment Entry for a Web Application
This example unsets the environment entry Hello User of the web application hello. The entry reverts to the value, if any, that is defined in the application's deployment descriptor.

```
asadmin> unset-web-env-entry --name="Hello User" hello
```

Command unset-web-env-entry executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

list-applications(1), list-web-env-entry(1), set-web-env-entry(1)

asadmin(1M)
update-admin-server-coordinates

updates administration server host and port information on domain nodes

Synopsis

update-admin-server-coordinates [--help]
[--adminhost admin-host]
[--adminport admin-port]
[--long={false|true}]
[node-name]

Description

The `update-admin-server-coordinates` subcommand updates administration server host and port information on domain nodes.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--adminhost
Specifies the host name of the administration server, qualified sufficiently for the domain nodes to contact it.

The default value is the actual host name of the current administration server.

--adminport
Specifies the port number that domain nodes are to use to connect to the administration server.

The default value is the port number of the current administration server.

--long
-1
Provides detailed information about the update operation.

The default value is false.

Operands

node-name

Restricts the update operation to the named domain node.

Exit Status

0

subcommand executed successfully
error in executing the subcommand

See Also

asadmin(1M)
update-admin-server-local-coordinates

updates administration server host and port information on a node

Synopsis

update-admin-server-local-coordinates [--help]
--adminhost admin-host
--adminport admin-port
[--nodedir node-directory]
node-name

Description

The update-admin-server-local-coordinates subcommand updates administration server host and port information on a local node.

This subcommand is supported in local mode only.

Options

--help
-?
Displays the help text for the subcommand.

--adminhost
Specifies the host name of the administration server.
The default value is the actual host name of the current administration server.

--adminport
Specifies the port number that the node is to use to connect to the administration server.
The default value is the port number of the current administration server.

--nodedir
Specifies the directory where node information is stored.
The default value is as-install/nodes.

Operands

node-name
Specifies the name of the node to update.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand
See Also

asadmin(1M)
update-connector-security-map

modifies a security map for the specified connector connection pool

Synopsis

update-connector-security-map [--help]
--poolname connector_connection_pool_name
[--addprincipals principal_name1[,principal_name2]...]
[--addusergroups user_group1[,user_group2]...]
[--removeprincipals principal_name1[,principal_name2]...]
[--removeusergroups user_group1[,user_group2]...]
[--mappedusername username]
mapname

Description

The update-connector-security-map subcommand modifies a security map for the
specified connector connection pool.

For this subcommand to succeed, you must have first created a connector connection
pool using the create-connector-connection-pool subcommand.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--poolname
Specifies the name of the connector connection pool to which the security map that is
to be updated belongs.

--addprincipals
Specifies a comma-separated list of EIS-specific principals to be added. Use either the
--addprincipals or --addusergroups options, but not both in the same command.

--addusergroups
Specifies a comma-separated list of EIS user groups to be added. Use either the
--addprincipals or --addusergroups options, but not both in the same command.

--removeprincipals
Specifies a comma-separated list of EIS-specific principals to be removed.

--removeusergroups
Specifies a comma-separated list of EIS user groups to be removed.

--mappedusername
Specifies the EIS username.
Operands

mapname
The name of the security map to be updated.

Examples

Example 1  Updating a Connector Security Map
This example adds principals to the existing security map named securityMap1.

asadmin> update-connector-security-map --poolname connector-pool1
--addprincipals principal1,principal2 securityMap1
Command update-connector-security-map executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-connector-security-map(1), delete-connector-security-map(1),
list-connector-security-maps(1)

asadmin(1M)
update-connector-work-security-map

modifies a work security map for the specified resource adapter

Synopsis

update-connector-work-security-map [--help] --raname raname
[--addprincipals eis-principal1=server-principal1, eis-principal2=server-principal2]*
[--addgroups eis-group1=server-group1, eis-group2=server-group2]*
[--removeprincipals eis-principal1, eis-principal2]*
[--removegroups eis-group1, eis-group2]*
mapname

Description

The update-connector-work-security-map subcommand modifies a security map for the specified resource adapter.
This subcommand is supported in remote mode only.

Options

--help
?-?
Displays the help text for the subcommand.

--addgroups
Specifies a comma-separated list of EIS groups to be added. Use either the
--addprincipals option or the --addgroups option, but not both.

--addprincipals
Specifies a comma-separated list of EIS-specific principals to be added. Use either the
--addprincipals option or the --addgroups option, but not both.

--removegroups
Specifies a comma-separated list of EIS groups to be removed.

--removeprincipals
Specifies a comma-separated list of EIS-specific principals to be removed.

--raname
Indicates the connector module name with which the work security map is associated.

Operands

mapname
The name of the work security map to be updated.

Examples

Example 1 Update a Connector Work Security Map
This example updates workSecurityMap2 by removing eis-group-2.
asadmin> update-connector-work-security-map
   --rename my-resource-adapter --removegroups eis-group-2 workSecurityMap2
Command update-connector-work-security-map executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

create-connector-work-security-map(1), delete-connector-work-security-map(1),
list-connector-work-security-maps(1)

asadmin(1M)
**update-file-user**

updates a current file user as specified

**Synopsis**

```bash
update-file-user [--help] [--groups user_groups[:user_groups]*] [--target target] [--authrealmname authrealm_name] username
```

**Description**

The `update-file-user` subcommand updates an existing entry in the keyfile using the specified user name, password and groups. Multiple groups can be entered by separating them, with a colon (:).

If a new password is not provided, this subcommand fails if secure administration is enabled and the user being updated is an administrative user.

**Options**

- **--help**
- `-?`
  Displays the help text for the subcommand.

- **--groups**
  This is the name of the group to which the file user belongs.

- **--authrealmname**
  Name of the authentication realm where the user to be updated can be found.

- **--target**
  This option helps specify the target on which you are updating a file user. Valid values are:

  - **server**
    Updates the file user in the default server instance. This is the default value.

  - **cluster_name**
    Updates the file user on every server instance in the cluster.

  - **instance_name**
    Updates the file user on a specified sever instance.

**Operands**

- **username**
  This is the name of the file user to be updated.
Examples

Example 1 Updating a User's Information in a File Realm
The following example updates information for a file realm user named sample_user.

asadmin> update-file-user
--groups staff:manager:engineer sample_user
Command update-file-user executed successfully

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

delete-file-user(1), list-file-users(1), create-file-user(1),
list-file-groups(1)
asadmin(1M)
update-node-config

updates the configuration data of a node

Synopsis

update-node-config [--help]
[--nodehost node-host]
[--installdir as-install-parent] [--nodedir node-dir]
node-name

Description

The update-node-config subcommand updates the configuration data of a node.
This subcommand can update any node, regardless of whether the node is enabled for remote communication. If a node that is enabled for remote communication is updated, the node is not enabled for remote communication after the update.
Options of this subcommand specify the new values of the node's configuration data. The default for these options is to leave the existing value unchanged.
This subcommand does not require the Distributed Component Object Model (DCOM) remote protocol or secure shell (SSH) to be configured to update the node. You may run this subcommand from any host that can contact the DAS.
This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--nodehost
The name of the host that the node is to represent after the node is updated.

--installdir
The full path to the parent of the base installation directory of the GlassFish Server software on the host, for example, /export/glassfish3.

--nodedir
The path to the directory that is to contain GlassFish Server instances that are created on the node. If a relative path is specified, the path is relative to the as-install directory, where as-install is the base installation directory of the GlassFish Server software on the host.

Operands

node-name
The name of the node to update. The node must exist. Otherwise, an error occurs.
Examples

Example 1  Updating the Host That a Node Represents
This example updates the host that the node sj04 represents to hsj04.

asadmin> update-node-config --nodehost hsj04 sj04
Command update-node-config executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

create-node-config(1), create-node-dcom(1), create-node-ssh(1),
delete-node-config(1), delete-node-dcom(1), delete-node-ssh(1), install-node(1),
install-node-dcom(1), install-node-ssh(1), list-nodes(1), uninstall-node(1),
uninstall-node-dcom(1), uninstall-node-ssh(1), update-node-dcom(1),
update-node-ssh(1)

asadmin(1M)
update-node-dcom

updates the configuration data of a node

Synopsis

update-node-dcom [--help]
[--nodehost node-host]
[--installdir as-install-parent] [--nodedir node-dir]
[--windowsuser windows-user] [--windowsdomain windows-domain]
[--force={false|true}]
node-name

Description

The update-node-dcom subcommand updates the configuration data of a node. This subcommand requires the Distributed Component Object Model (DCOM) remote protocol to be configured on the host where the domain administration server (DAS) is running and on the host where the node resides. The DCOM protocol is available only on Windows systems. You may run this subcommand from any host that can contact the DAS.

This subcommand can update any node, regardless of whether the node is enabled for remote communication. If the node is not enabled for remote communication, the subcommand enables DCOM communication for the node and updates any other specified configuration data.

Options of this subcommand specify the new values of the node's configuration data. The default for most options is to leave the existing value unchanged. However, if this subcommand is run to enable DCOM communication for a node, default values are applied if any of the following options is omitted:

- --windowsuser
- --windowsdomain

By default, the subcommand fails and the node is not updated if the DAS cannot contact the node's host through DCOM. To force the node to be updated even if the host cannot be contacted through DCOM, set the --force option to true.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--nodehost
The name of the host that the node is to represent after the node is updated.

--installdir
The full path to the parent of the base installation directory of the GlassFish Server software on the host, for example, /export/glassfish3/.
update-node-dcom

**--nodedir**
The path to the directory that is to contain GlassFish Server instances that are created on the node. If a relative path is specified, the path is relative to the *as-install* directory, where *as-install* is the base installation directory of the GlassFish Server software on the host.

**--windowsuser**
-w
The user on this node’s host that is to run the process for connecting to the host through DCOM. The default depends on whether this subcommand is run to enable DCOM communication for the node:

- If the node is already enabled for communication over DCOM, the default is to leave the user unchanged.
- If this subcommand is run to enable DCOM communication for the node, the default is the user that is running the DAS process.

If the **--nodehost** option is set to localhost, the **--windowsuser** option is ignored.

**--windowsdomain**
-d
The name of the Windows domain that contains the user that the **--windowsuser** option specifies. The default depends on whether this subcommand is run to enable DCOM communication for the node:

- If the node is already enabled for communication over DCOM, the default is to leave the domain unchanged.
- If this subcommand is run to enable DCOM communication for the node, the default is the name of the host on which the subcommand is run.

**--force**
Specifies whether the node is updated even if validation of the node’s parameters fails. To validate a node’s parameters, the DAS must be able to contact the node’s host through DCOM. Possible values are as follows:

- **false**
The node is not updated if validation of the node’s parameters fails (default).
- **true**
The node is updated even if validation of the node’s parameters fails.

Operands

**node-name**
The name of the node to update. The node must exist. Otherwise, an error occurs.

Examples

**Example 1  Updating the Host That a Node Represents**
This example updates the host that the node wpmd11 represents to wj01.

asadmin> update-node-dcom --nodehost wj01 wpmd11
Command update-node-dcom executed successfully.
Example 2  Forcing the Update of a Node

This example forces the update of the node xkyd to enable the node to communicate over DCOM.

```
$ asadmin> update-node-dcom --force xkyd
Warning: some parameters appear to be invalid.
Could not find a remote Glassfish installation on host: xkyd.example.com at
C:\glassfish3\glassfish
Continuing with node update due to use of --force.
Command update-node-dcom executed successfully.
```

Exit Status

```
0
command executed successfully

1
error in executing the command
```

See Also

```
create-node-config(1), create-node-dcom(1), create-node-ssh(1),
delete-node-config(1), delete-node-dcom(1), delete-node-ssh(1), install-node(1),
install-node-dcom(1), install-node-ssh(1), list-nodes(1), uninstall-node(1),
uninstall-node-dcom(1), uninstall-node-ssh(1), update-node-config(1),
update-node-ssh(1)
```

```
 huntsman(1M)
```

Oracle GlassFish Server 3.1.2 Reference Manual
update-node-ssh

updates the configuration data of a node

Synopsis

update-node-ssh [--help]
[--nodehost node-host]
[--installdir as-install-parent] [--nodestack node-dir]
[--sshport ssh-port] [--sshuser ssh-user]
[--sshkeyfile ssh-keyfile]
[--force=(false|true)]
node-name

Description

The update-node-ssh subcommand updates the configuration data of a node. This subcommand requires secure shell (SSH) to be configured on the machine where the domain administration server (DAS) is running and on the machine where the node resides. You may run this subcommand from any machine that can contact the DAS.

This subcommand can update any node, regardless of whether the node is enabled for remote communication. If the node is not enabled for remote communication, the subcommand enables SSH communication for the node and updates any other specified configuration data.

Options of this subcommand specify the new values of the node's configuration data. The default for most options is to leave the existing value unchanged. However, if this subcommand is run to enable SSH communication for a node, default values are applied if any of the following options is omitted:

- --sshport
- --sshuser
- --sshkeyfile

By default, the subcommand fails and the node is not updated if the DAS cannot contact the node's host through SSH. To force the node to be updated even if the host cannot be contacted through SSH, set the --force option to true.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

--nodehost
The name of the host that the node is to represent after the node is updated.

--installdir
The full path to the parent of the base installation directory of the GlassFish Server software on the host, for example, /export/glassfish3.
The path to the directory that is to contain GlassFish Server instances that are created on the node. If a relative path is specified, the path is relative to the as-install directory, where as-install is the base installation directory of the GlassFish Server software on the host.

The port to use for SSH connections to this node’s host. The default depends on whether this subcommand is run to enable SSH communication for the node:

- If the node is already enabled for communication over SSH, the default is to leave the port unchanged.
- If this subcommand is run to enable SSH communication for the node, the default port is 22.

If the --nodehost is set to localhost, the --sshport option is ignored.

The user on this node’s host that is to run the process for connecting to the host through SSH. The default depends on whether this subcommand is run to enable SSH communication for the node:

- If the node is already enabled for communication over SSH, the default is to leave the user unchanged.
- If this subcommand is run to enable SSH communication for the node, the default is the user that is running the DAS process.

If the --nodehost option is set to localhost, the --sshuser option is ignored.

The absolute path to the SSH private key file for user that the --sshuser option specifies. This file is used for authentication to the sshd daemon on the node’s host.

Note: GlassFish Server also supports password authentication through the AS_ADMIN_SSHPASSWORD entry in the password file. The password file is specified in the --paswordfile option of the asadmin(1M) utility.

If the SSH private key file is protected by a passphrase, the password file must contain the AS_ADMIN_SSHKEYPASSPHRASE entry.

The path to the key file must be reachable by the DAS and the key file must be readable by the DAS.

The default depends on whether this subcommand is run to enable SSH communication for the node:

- If the node is already enabled for communication over SSH, the default is to leave the key file unchanged.
- If this subcommand is run to enable SSH communication for the node, the default is the key file in the user’s .ssh directory. If multiple key files are found, the subcommand uses the following order of preference:

  1. id_rsa
  2. id_dsa
  3. identity
--force
Specifies whether the node is updated even if validation of the node’s parameters fails. To validate a node’s parameters, the DAS must be able to contact the node’s host through SSH. Possible values are as follows:

false
The node is not updated if validation of the node’s parameters fails (default).

true
The node is updated even if validation of the node’s parameters fails.

Operands

**node-name**
The name of the node to update. The node must exist. Otherwise, an error occurs.

Examples

**Example 1 Updating the Host That a Node Represents**
This example updates the host that the node lssh represents to sj04.

```
asadmin> update-node-ssh --nodehost sj04 lssh
Command update-node-ssh executed successfully.
```

**Example 2 Forcing the Update of a Node**
This example forces the update of the node sj01 to enable the node to communicate over SSH.

```
asadmin> update-node-ssh --force sj01
Warning: some parameters appear to be invalid.
Could not connect to host sj01 using SSH.
Could not authenticate. Tried authenticating with specified key at /home/gfuser/.ssh/id_rsa
Continuing with node update due to use of --force.
Command update-node-ssh executed successfully.
```

Exit Status

**0**
command executed successfully

**1**
error in executing the command

See Also

create-node-dcom(1), create-node-config(1), create-node-ssh(1),
delete-node-config(1), delete-node-dcom(1), delete-node-ssh(1), install-node(1),
install-node-dcom(1), install-node-ssh(1), list-nodes(1), uninstall-node(1),
uninstall-node-dcom(1), uninstall-node-ssh(1), update-node-config(1),
update-node-dcom(1)

asadmin(1M)
update-password-alias

updates a password alias

Synopsis

update-password-alias [--help]
aliasname

Description

This subcommand updates the password alias IDs in the named target. An alias is a
token of the form ${ALIAS=password-alias-password}. The password corresponding
to the alias name is stored in an encrypted form. The update-password-alias
subcommand takes both a secure interactive form (in which the user is prompted for
all information) and a more script-friendly form, in which the password is propagated
on the command line.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Operands

aliasname

This is the name of the password as it appears in domain.xml.

Examples

Example 1  Updating a Password Alias

asadmin> update-password-alias jmspassword-alias
Please enter the alias password>
Please enter the alias password again>
Command update-password-alias executed successfully.

Exit Status

0
command executed successfully

1
error in executing the command

See Also

delete-password-alias(1), list-password-aliases(1), create-password-alias(1)

asadmin(1M)
uptime

returns the length of time that the DAS has been running

Synopsis

uptime [--help]

Description

The uptime subcommand returns the length of time that the domain administration server (DAS) has been running since it was last restarted.

This subcommand is supported in remote mode only.

Options

--help
-?
Displays the help text for the subcommand.

Examples

Example 1  Showing How Long the DAS Has Been Running
This example shows the length of time that the DAS has been running.

asadmin> uptime
Uptime: 2 days, 1 hours, 30 minutes, 18 seconds, Total milliseconds: 178218706
Command uptime executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

list-domains(1), start-domain(1), stop-domain(1)

asadmin(1M)
uptime
validate-dcom
tests the connection over DCOM to a remote host

Synopsis
validate-dcom [--help]
[--windowsuser windows-user] [--windowsdomain windows-domain]
[--remotetestdir remote-test-directory]
[--verbose={false|true}] host-name

Description
The validate-dcom subcommand tests the connection over the Distributed Component Object Model (DCOM) remote protocol to the remote host that is specified as the operand of the subcommand. The DCOM protocol is available only on Windows systems.

The prerequisites for running this subcommand are as follows:

- DCOM must be configured on the host where the domain administration server (DAS) is running and on the remote host that is specified as the operand of the subcommand.
- The following ports must be open on the remote host:
  - DCOM port 135 or 139
  - Windows Shares port 445
- The Server Windows Service is enabled on the remote host.

This subcommand verifies whether the prerequisites for creating a DCOM node or creating a GlassFish Server instance on a DCOM node are met. If this subcommand fails, any attempt to create a DCOM node or create an instance on a DCOM node will also fail.

You may run this command from any machine that can contact the DAS.

The subcommand tests the connection over DCOM to the remote host by performing the following operations on the remote host:

- Resolving the host name
- Connecting to DCOM through port 135 or 139
- Connecting to Windows Shares through port 445
- Copying a script that is named delete_me.bat in the folder that is specified by the --remotetestdir option
- Running the script to obtain a listing of the folder in which the delete_me.bat was written

By default, the subcommand displays no information about the operations that are performed to test the connection to the remote host. To display information about these operations, set the --verbose option to true.

This subcommand is supported in remote mode only.
Options

--help
-?
Displays the help text for the subcommand.

--windowsuser
-w
The user on the remote host that is to run the process for connecting to that host through DCOM. The default is the user that is running the DAS process.

--windowsdomain
-d
The name of the Windows domain that contains the user that the --windowsuser option specifies. The default is the name of the host on which the subcommand is run.

--remotetestdir
Specifies the folder on the remote host in which the subcommand creates the delete_me.bat file. The user that the --windowsuser option specifies must have write access to this folder. Otherwise an error occurs. The default is C:\

--verbose
-v
Specifies whether the subcommand displays information about the operations that are performed to test the connection to the remote host.
Possible values are as follows:

false
No information about the operations that are performed to test the connection is displayed (default).

ture
The subcommand displays information about the operations that are performed to test the connection.

If the --verbose option is true and the AS_DEBUG environment variable is set, the subcommand also displays the command line for each process that is running on the remote host.

Operands

host-name
The name of the host to which to test the connection over DCOM.

Examples

Example 1 Testing the Connection Over DCOM to a Remote Host
This example tests the connection over DCOM to the host wpmd12.

asadmin> validate-dcom --wpmd12
Command validate-dcom executed successfully.
**Example 2 Verbosely Testing the Connection Over DCOM to a Remote Host**

This example tests the connection over DCOM to the host wpmdl2 and displays information about the operations that are performed to test the connection to the remote host.

```bash
asadmin> validate-dcom --verbose wpmdl2
```

Successfully resolved host name to: wpmdl2/192.168.122.106
Successfully connected to DCOM Port at port 135 on host wpmdl2.
Successfully connected to NetBIOS Session Service at port 139 on host wpmdl2.
Successfully connected to Windows Shares at port 445 on host wpmdl2.
Successfully accessed C: on wpmdl2 using DCOM.
Successfully wrote delete_me.bat to C: on wpmdl2 using DCOM.
Successfully accessed WMI (Windows Management Interface) on wpmdl2. There are 40 processes running on wpmdl2.
Successfully ran the test script on wpmdl2 using DCOM.
The script simply ran the DIR command. Here are the first few lines from the output of the dir command on the remote machine:

```
C:\Windows\system32>dir C:\
Volume in drive C has no label.
Volume Serial Number is XXXX-XXX

Directory of C:\

12/05/2011  10:21 AM                63 .asadminpass
02/15/2011  01:08 PM    <DIR>          aroot
12/05/2011  04:43 PM                 8 delete_me.bat
03/03/2011  10:36 AM    <DIR>          export
```

Command validate-dcom executed successfully.

**Exit Status**

0  
command executed successfully

1  
error in executing the command

**See Also**

- `create-node-dcom(1)`, `delete-node-dcom(1)`, `list-nodes(1)`, `ping-node-dcom(1)`, `update-node-dcom(1)`
- `asadmin(1M)`
validate-multicast

validates that multicast transport is available for clusters

Synopsis

validate-multicast [--help] [--multicastport multicastport]
[--multicastaddress multicastaddress]
[--bindaddress bindaddress]
[--sendperiod sendperiod]
[--timeout timeout]
[--timetolive timetolive]
[--verbose={false|true}]

Description

The validate-multicast subcommand validates that multicast transport is available for clusters. You should run this subcommand at the same time on each of the hosts to be validated. This subcommand is available in local mode.

Note: Do not run the validate-multicast subcommand using the DAS and cluster’s multicast address and port values while the DAS and cluster are running. Doing so results in an error.

The validate-multicast subcommand must be run at the same time on two or more machines to validate whether multicast messages are being received between the machines.

As long as all machines see each other, multicast is validated to be working properly across the machines. If the machines are not seeing each other, set the --bindaddress option explicitly to ensure that all machines are using interface on same subnet, or increase the --timetolive option from the default of 4. If these changes fail to resolve the multicast issues, ask the network administrator to verify that the network is configured so the multicast messages can be seen between all the machines used to run the cluster.

Options

--help
-?
Displays the help text for the subcommand.

--multicastport
The port for the multicast socket on which the Group Management Service (GMS) listens for group events. Specify a standard UDP port number in the range 2048-32000. The default is 2048.

--multicastaddress
The address for the multicast socket on which the GMS listens for group events. Specify a class D IP address. Class D IP addresses are in the range 224.0.0.0 to 239.255.255.255, inclusive. The address 224.0.0.0 is reserved and should not be used. The default is 228.9.3.1.
validate-multicast

--bindaddress
The local interface to receive multicast datagram packets for the GMS. The default is to use all available binding interfaces.

On a multi-home machine (possessing two or more network interfaces), this attribute enables you to indicate which network interface is used for the GMS. This value must be a local network interface IP address.

--sendperiod
The number of milliseconds between test messages sent between nodes. The default is 2000.

--timeout
The number of seconds before the subcommand times out and exits. The default is 20. You can also exit this subcommand using Ctrl-C.

--timetolive
The default time-to-live for multicast packets sent out on the multicast socket in order to control the scope of the multicasts. The time-to-live value must be between zero and 255 inclusive. The default is the JDK default or a minimum defined by a constant in the GMS subsystem, whichever is lower. To see the time-to-live value being used, use the --verbose option.

--verbose
If used without a value or set to true, provides additional debugging information. The default is false.

Examples

Example 1 Validating multicast transport
Run from host machine1:

asadmin> validate-multicast
Will use port 2,048
Will use address 228.9.3.1
Will use bind address null
Will use wait period 2,000 (in milliseconds)

Listening for data...
Sending message with content "machine1" every 2,000 milliseconds
Received data from machine1 (loopback)
Received data from machine2
Exiting after 20 seconds. To change this timeout, use the --timeout command line option.
Command validate-multicast executed successfully

Run from host machine2:

asadmin> validate-multicast
Will use port 2,048
Will use address 228.9.3.1
Will use bind address null
Will use wait period 2,000 (in milliseconds)

Listening for data...
Sending message with content "machine2" every 2,000 milliseconds
Received data from machine2 (loopback)
Received data from machine1
Exiting after 20 seconds. To change this timeout, use the --timeout command line
validate-multicast

Option
Command validate-multicast executed successfully

Exit Status

0
command executed successfully

1
error in executing the command

See Also

get-health(1)
asadmin(1M)
verify-domain-xml

verifies the content of the domain.xml file

Synopsis

verify-domain-xml [--help]
[--domindir domain-dir] [domain-name]

Description

Verifies the content of the domain.xml file by checking the following:

■ That the domain.xml file can be parsed
■ That the names for elements that have them are unique

This subcommand is supported in local mode only.

Options

-h --help
Displays the help text for the subcommand.

--domindir
Specifies the domain root directory, where the domains are located. The path must be accessible in the file system. The default is as-install/domains.

Operands

domain_name
Specifies the name of the domain. The default is domain1.

Examples

Example 1 Using verify-domain-xml

asadmin> verify-domain-xml
All Tests Passed.
domain.xml is valid

Exit Status

0
command executed successfully
1
error in executing the command

See Also

asadmin(1M)
version

displays version information for GlassFish Server

Synopsis

version [--help] [--verbose={false|true}] [--local={false|true}]

Description

The version subcommand displays version information for GlassFish Server. By default, if the subcommand cannot contact the domain administration server (DAS), the subcommand retrieves the version information locally and displays a warning message.

This subcommand is supported in remote mode and local mode.

Options

--help
  -?
  Displays the help text for the subcommand.

--verbose
  -v
  If this option is set to true, the subcommand provides the version of the Java Runtime Environment (JRE) that the server is running. The default is false.

--local
  If this option is set to true, the subcommand obtains the version locally from the installation of GlassFish Server on the host where the subcommand is run.
  If this option is set to false (default), the subcommand attempts to contact the DAS to obtain the version. If the attempt to contact the DAS fails, the subcommand retrieves the version locally and displays a warning message.

Examples

Example 1 Obtaining Version Information From a Running DAS

asadmin> version
Version = Oracle GlassFish Server 3.1 (build 34)
Command version executed successfully.

Example 2 Obtaining Version Information When the DAS Cannot be Reached

asadmin> version
Version string could not be obtained from Server [localhost:4848] for some reason. (Turn debugging on e.g. by setting AS_DEBUG=true in your environment, to see the details).
Using locally retrieved version string from version class.
Version = Oracle GlassFish Server 3.1 (build 34)
Command version executed successfully.
Example 3  Obtaining Version Information Locally

asadmin> version --local
Using locally retrieved version string from version class.
Version = Oracle GlassFish Server 3.1 (build 34)
Command version executed successfully.

Exit Status

0
subcommand executed successfully

1
error in executing the subcommand

See Also

list-modules(1)
asadmin(1M)
This section describes GlassFish Server utility commands.
appclient

launches the Application Client Container and invokes the client application typically packaged in the application JAR file

Synopsis

appclient [client_application_classfile | -client client_application_jar]
[-mainclass main_class_name | -name display_name]
[-xml sun-acc.xml file] [-textauth]
[-targetserver host[:port][,host[:port]...]]
[-user username] [-passwordfile password_file]
[application-options]

appclient [jvm-options]
[-mainclass main_class_name | -name display_name]
[-xml client_config_xml_file] [-textauth]
[-targetserver host[:port][,host[:port]...]]
[-user username] [-passwordfile password_file]

class-selector [application-options]

Description

Use the appclient command to launch the Application Client Container and invoke a client application that is typically packaged in an application JAR file. The application client JAR file is specified and created during deployment by the Administration Console or the asadmin deploy command with the --retrieve option. You can also retrieve the client JAR file using the asadmin get-client-stubs command.

The Application Client Container is a set of Java classes, libraries, and other files that are required to execute a first-tier application client program on a Virtual Machine for the Java platform (JVM machine). The Application Client Container communicates with the server using RMI-IIOP.

The client JAR file that is retrieved after deploying an application should be passed with the -client or -jar option when running the appclient utility. The client JAR file name is of the form app-nameClient.jar. For multiple application clients in an EAR file, you must use the -mainclass or -name option to specify which client to invoke.

If the application client is a stand-alone module or the only client in an EAR file, the Application Client Container can find the client without using the -mainclass or -name options. If you provide a -mainclass or -name value that does not match what is in the client, the Application Client Container launches the client anyway but issues a warning that the selection did not match the information in the client. The warning also displays what the actual main class and name values are for the client.

Options

jvm-options
optional; you can set JVM options for the client application. These can be any valid java command options except -client or -jar. JVM options can be intermixed with other appclient command options as long as both types of options appear before the class-selector.
**client_application_classfile**
optional; the file system pathname of the client application .class file. A relative pathname must be relative to the current directory. This class file must contain the `main()` method to be invoked by the Application Client Container.

If you use `client_application_classfile` and the class is dependent on other user classes, you must also set the classpath. You can either use the `-classpath` JVM option in the `appclient` command or set the `CLASSPATH` environment variable. For more information about setting a classpath, see Setting the Class Path, Oracle Solaris Version (http://download.oracle.com/javase/6/docs/technotes/tools/solaris/classpath.html) or Setting the Class Path, Windows Version (http://download.oracle.com/javase/6/docs/technotes/tools/windows/classpath.html).

**-client**
optional; the name and location for the client JAR file.

**-mainclass**
optional; the full classname of the main client application as specified in the Main-Class entry in the MANIFEST.MF file. Used for a multiple client applications. By default, uses the class specified in the client jar. For example, com.example.test.AppClient.

**-name**
optional; the display name for the client application. Used for multiple client applications. By default, the display name is specified in the client jar application-client.xml file which is identified by the display-name attribute.

**-xml**
optional if using the default domain, instance, and name (sun-acc.xml), otherwise it is required; identifies the name and location of the client configuration XML file. If not specified, defaults to the sun-acc.xml file in the domain-dir/config directory.

**-textauth**
optional; used to specify using text format authentication when authentication is needed.

**-targetserver**
optional; a comma-separated list of one or more server specifications for ORB endpoints. Each server specification must include at least the host. Optionally, a server specification can include the port as well. If the port is omitted from a server specification, the default value, 3700, is used for that host.

**-user**
optional; the application user who is authorized to have access to particular guarded components in the application, such as EJB components.

**-passwordfile**
optional; specifies the name, including the full path, of a file that contains the password for application clients in the following format:

```
PASSWORD=appclient-password
```

If this option is omitted, the password is specified interactively at the command prompt.
Caution: Avoid specifying a password interactively at the command prompt. Such a password can be seen by users who can run commands to display running processes and the commands to start them, such as ps.

For security reasons, a password that is specified as an environment variable is not read by the appclient utility.

**class-selector**

required; you must specify the client application class using one of the following class selectors.

- **-jar jar-file**
  
  the name and location of the client JAR file. The application client JAR file is specified and created during deployment by the asadmin deploy command. If specified, the -classpath setting is ignored in deference to the Class-Path setting in the client JAR file's manifest.

- **class-name**
  
  the fully qualified name of the application client's main class. The Application Client Container invokes the main method of this class to start the client. For example, com.example.test.AppClient.

  If you use class-name as the class selector, you must also set the classpath. You can either use the -classpath JVM option in the appclient command or set the CLASSPATH environment variable. For more information about setting a classpath, see Setting the Class Path, Oracle Solaris Version (http://download.oracle.com/javase/6/docs/technotes/tools/solaris/classpath.html) or Setting the Class Path, Windows Version (http://download.oracle.com/javase/6/docs/technotes/tools/windows/classpath.html).

**application-options**

optional; you can set client application arguments.

**Examples**

Example 1 Using the appclient command

```
appclient -xml sun-acc.xml -jar myclientapp.jar scott sample
```

Where: sun-acc.xml is the name of the client configuration XML file, myclientapp.jar is the client application .jar file, and scott and sample are arguments to pass to the application. If sun-acc.xml and myclientapp.jar are not in the current directory, you must give the absolute path locations; otherwise the relative paths are used. The relative path is relative to the directory where the command is being executed.

**Attributes**

See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Stability</td>
<td>Unstable</td>
</tr>
</tbody>
</table>
See Also

get-client-stubs(1)
asadmin(1M), package-appclient(1M)
asadmin

utility for performing administrative tasks for Oracle GlassFish Server

Synopsis

asadmin [--host host]
[--port port]
[--user admin-user]
[--passwordfile filename]
[--terse={true|false}]
[--secure={false|true}]
[--echo={true|false}]
[--interactive={true|false}]
[--help]
[subcommand [options] [operands]]

Description

Use the asadmin utility to perform administrative tasks for Oracle GlassFish Server. You can use this utility instead of the Administration Console interface.

Subcommands of the asadmin Utility

The subcommand identifies the operation or task that you are performing. Subcommands are case-sensitive. Each subcommand is either a local subcommand or a remote subcommand.

- A local subcommand can be run without a running domain administration server (DAS). However, to run the subcommand and have access to the installation directory and the domain directory, the user must be logged in to the machine that hosts the domain.
- A remote subcommand is always run by connecting to a DAS and running the subcommand there. A running DAS is required.

asadmin Utility Options and Subcommand Options

Options control the behavior of the asadmin utility and its subcommands. Options are also case-sensitive.

The asadmin utility has the following types of options:

- **asadmin utility options.** These options control the behavior of the asadmin utility, not the subcommand. The asadmin utility options may precede or follow the subcommand, but asadmin utility options after the subcommand are deprecated. All asadmin utility options must either precede or follow the subcommand. If asadmin utility options are specified both before and after the subcommand, an error occurs. For a description of the asadmin utility options, see the "Options" section of this help information.

- **Subcommand options.** These options control the behavior of the subcommand, not the asadmin utility. Subcommand options must follow the subcommand. For a description of a subcommand’s options, see the help information for the subcommand.
A subcommand option may have the same name as an `asadmin` utility option, but the effects of the two options are different.

The `asadmin` utility options and some subcommand options have a long form and a short form.

- The long form of an option has two dashes (`--`) followed by an option word.
- The short form of an option has a single dash (`-`) followed by a single character.

For example, the long form and the short form of the option for specifying terse output are as follows:

- Long form: `--terse`
- Short form: `-t`

Most options require argument values, except Boolean options, which toggle to enable or disable a feature.

**Operands of `asadmin` Subcommands**

Operands specify the items on which the subcommand is to act. Operands must follow the argument values of subcommand options, and are set off by a space, a tab, or double dashes (`--`). The `asadmin` utility treats anything that follows the subcommand options and their values as an operand.

**Escape Characters in Options for the `asadmin` Utility**

Escape characters are required in options of the `asadmin` utility for the following types of characters:

- **Meta characters in the UNIX operating system.** These characters have special meaning in a shell. Meta characters in the UNIX operating system include: `\/,!$%^&*|{}[]"'`.

  To disable these characters, use the backslash (`\`) escape character or enclose the entire command-line argument in single quote (`'`) characters.

  The following examples illustrate the effect of escape characters on the `*` character. In these examples, the current working directory is the domains directory.

  - The following command, without the escape character, echoes all files in the current directory:

    ```
prompt% echo *
domain1 domain2
    ```

  - The following command, in which the backslash (`\`) escape character precedes the `*` character, echoes the `*` character:

    ```
prompt% echo \*
    ```

  - The following command, in which the `*` character is enclosed in single quote (`'`) characters, echoes the `*` character:

    ```
prompt% echo '*'
    ```

  The escape character is also a special character in the UNIX operating system and in the Java language. Therefore, in the UNIX operating system and in multimode, you must apply an additional escape character to every escape character in the command line. This requirement does not apply to the Windows operating system.
For example, the backslash (\) UNIX operating system meta character in the option argument `Test\Escape\Character` is specified on UNIX and Windows systems as follows:

- On UNIX systems, each backslash must be escaped with a second backslash:
  `Test\\Escape\\Character`

- On Windows systems, no escape character is required:
  `Test\Escape\Character`

---

**Note:** In contexts where meta characters in the UNIX operating system are unambiguous, these characters do not require escape characters. For example, in the `set(1)` subcommand, the value that is to be set is specified as `name=value`. Because `name` can never include an equals sign, no escape character is required to disable the equals sign. Therefore, everything after the equals sign is an uninterpreted string that the `set` subcommand uses unchanged.

---

- **Spaces.** The space is the separator in the command-line interface. To distinguish a space in a command-line argument from the separator in the command-line interface, the space must be escaped as follows:
  - For the UNIX operating system in single mode and multimode, and for all operating systems in multimode, use the backslash (\) escape character or enclose the entire command-line argument in single quote (') characters or double quote (") characters.
  - For the Windows operating system in single mode, enclose the entire command-line argument in double quote (") characters.

- **Option delimiters.** The `asadmin` utility uses the colon character (:) as a delimiter for some options. The backslash (\) escape character is required if the colon is part of any of the following items:
  - A property
  - An option of the Virtual Machine for the Java platform (Java Virtual Machine or JVM machine)

For example, the operand of the subcommand `create-jvm-options(1)` specifies JVM machine options in the following format:

```
(jvm-option-name=jvm-option-value)
[:jvm-option-name=jvm-option-value]*
```

Multiple JVM machine options in the operand of the `create-jvm-options` subcommand are separated by the colon (:) delimiter. If `jvm-option-name` or `jvm-option-value` contains a colon, the backslash (\) escape character is required before the colon.

The backslash (\) escape character is also required before a single quote (‘) character or a double quote (”) character in an option that uses the colon as a delimiter.

---

1 The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform.
When used without single quote (’) characters, the escape character disables the option delimiter in the command-line interface.

For the UNIX operating system in single mode and multimode, and for all operating systems in multimode, the colon character and the backslash character in an option that uses the colon as a delimiter must be specified as follows:

- To pass a literal backslash to a subcommand, two backslashes are required. Therefore, the colon (:) must be escaped by two backslashes (\). 
- To prevent a subcommand from treating the backslash as a special character, the backslash must be escaped. As a result, two literal backslashes (\) must be passed to the subcommand. To prevent the shell from interpreting these backslashes as special characters, each backslash must be escaped. Therefore, the backslash must be specified by a total of four backslashes (\\).

For the Windows operating system in single mode, a backslash (\) is required to escape the colon (:) and the backslash (\) in an option that uses the colon as a delimiter.

Instead of using the backslash (\) escape character, you can use the double quote (") character or single quote (’) character. The effects of the different types of quote characters on the backslash (\) character are as follows:

- Between double quote (") characters, the backslash (\) character is a special character.
- Between single quote (’) characters, the backslash (\) character is not a special character.

Requirements for Using the --secure Option
The requirements for using the --secure option are as follows:

- The domain that you are administering must be configured for security.
- The security-enabled attribute of the http-listener element in the DAS configuration must be set to true.

To set this attribute, use the set subcommand.

Server Restart After Creation or Deletion
When you use the asadmin subcommands to create or delete a configuration item, you must restart the DAS for the change to take effect. To restart the DAS, use the restart-domain(1) subcommand.

Help Information for Subcommands and the asadmin Utility
To obtain help information for an asadmin utility subcommand, specify the subcommand of interest as the operand of the help subcommand. For example, to obtain help information for the start-domain(1) subcommand, type:

```
asadmin help start-domain
```

If you run the help subcommand without an operand, this help information for the asadmin utility is displayed.

To obtain a listing of available asadmin subcommands, use the list-commands(1) subcommand.
Options

--host
-H
The machine name where the DAS is running. The default value is localhost.

--port
-p
The HTTP port or HTTPS port for administration. This port is the port in the URL that you specify in your web browser to manage the domain. For example, in the URL http://localhost:4949, the port is 4949.

The default port number for administration is 4848.

--user
-u
The user name of the authorized administrative user of the DAS.

If you have authenticated to a domain by using the `asadmin login` command, you need not specify the --user option for subsequent operations on the domain.

--passwordfile
-W
Specifies the name, including the full path, of a file that contains password entries in a specific format.

The entry for a password must have the AS_ADMIN_ prefix followed by the password name in uppercase letters, an equals sign, and the password.

The entries in the file that are read by the `asadmin` utility are as follows:

- AS_ADMIN_PASSWORD=administration-password
- AS_ADMIN_MASTERPASSWORD=master-password

The entries in this file that are read by subcommands are as follows:

- AS_ADMIN_NEWPASSWORD=new-administration-password (read by the `start-domain(1)` subcommand)
- AS_ADMIN_USERPASSWORD=user-password (read by the `create-file-user(1)` subcommand)
- AS_ADMIN_ALIASPASSWORD=alias-password (read by the `create-password-alias(1)` subcommand)
- AS_ADMIN_MAPPEDPASSWORD=mapped-password (read by the `create-connector-security-map(1)` subcommand)
- AS_ADMIN_WINDOWSPASSWORD=windows-password (read by the `create-node-dcom(1), install-node-dcom(1), and update-node-dcom(1)` subcommands)
- AS_ADMIN_SSHPASSWORD=sshd-password (read by the `create-node-ssh(1), install-node(1), install-node-ssh(1), and update-node-ssh(1)` subcommands)
- AS_ADMIN_SSHKEYPASSPHRASE=sshd-passphrase (read by the `create-node-ssh(1), install-node(1), install-node-ssh(1), and update-node-ssh(1)` subcommands)
- AS_ADMIN_JMSDBPASSWORD=jdbc-user-password (read by the `configure-jms-cluster(1)` subcommand)

These password entries are stored in clear text in the password file. To provide additional security, the `create-password-alias` subcommand can be used to create aliases for passwords that are used by remote subcommands. The password for which
the alias is created is stored in an encrypted form. If an alias exists for a password, the alias is specified in the entry for the password as follows:

```
AS_ADMIN_password-name=${ALIAS=password-alias-name}
```

For example:

```
AS_ADMIN_SSHPASSWORD=${ALIAS=ssh-password-alias}
AS_ADMIN_SSHKEYPASSPHRASE=${ALIAS=ssh-key-passphrase-alias}
```

In domains that do not allow unauthenticated login, all remote subcommands must specify the administration password to authenticate to the DAS. The password can be specified by one of the following means:

- Through the `--passwordfile` option
- Through the `login(1)` subcommand
- Interactively at the command prompt

The `login` subcommand can be used to specify only the administration password. For other passwords that remote subcommands require, use the `--passwordfile` option or specify them at the command prompt.

After authenticating to a domain by using the `asadmin login` command, you need not specify the administration password through the `--passwordfile` option for subsequent operations on the domain. However, only the `AS_ADMIN_PASSWORD` option is not required. You still must provide the other passwords, for example, `AS_ADMIN_USERPASSWORD`, when required by individual subcommands, such as `update-file-user(1)`.

For security reasons, a password that is specified as an environment variable is not read by the `asadmin` utility.

The master password is not propagated on the command line or an environment variable, but can be specified in the file that the `--passwordfile` option specifies.

The default value for `AS_ADMIN_MASTERPASSWORD` is `changeit`.

```
--terse
-t
```

If true, output data is very concise and in a format that is optimized for use in scripts instead of for reading by humans. Typically, descriptive text and detailed status messages are also omitted from the output data. Default is false.

```
--secure
-s
```

If set to true, uses SSL/TLS to communicate with the DAS.

The default is false.

```
--echo
-e
```

If set to true, the command-line statement is echoed on the standard output. Default is false.

```
--interactive
-I
```

If set to true, only the required options are prompted.

The default depends on how the `asadmin` utility is run:

- If the `asadmin` utility is run from a console window, the default is true.
If the `asadmin` utility is run without a console window, for example, from within a script, the default is `false`.

```bash
--help
-?
```
Displays the help text for the `asadmin` utility.

**Examples**

**Example 1  Running an `asadmin` Utility Subcommand in Single Mode**
This example runs the `list-applications(1)` subcommand in single mode. In this example, the default values for all options are used.

The example shows that the application `hello` is deployed on the local host.

```bash
asadmin list-applications
  hello <web>
```
Command `list-applications` executed successfully.

**Example 2  Specifying an `asadmin` Utility Option With a Subcommand**
This example specifies the `--host` `asadmin` utility option with the `list-applications` subcommand in single mode. In this example, the DAS is running on the host `srvr1.example.com`.

The example shows that the applications `basic-ezcomp`, `scrumtoys`, `ejb31-war`, and `automatic-timer-ejb` are deployed on the host `srvr1.example.com`.

```bash
asadmin --host srvr1.example.com list-applications
  basic-ezcomp <web>
  scrumtoys <web>
  ejb31-war <ejb, web>
  automatic-timer-ejb <ejb>
```
Command `list-applications` executed successfully.

**Example 3  Specifying an `asadmin` Utility Option and a Subcommand Option**
This example specifies the `--host` `asadmin` utility option and the `--type` subcommand option with the `list-applications` subcommand in single mode. In this example, the DAS is running on the host `srvr1.example.com` and applications of type `web` are to be listed.

```bash
asadmin --host srvr1.example.com list-applications --type web
  basic-ezcomp <web>
  scrumtoys <web>
  ejb31-war <ejb, web>
```
Command `list-applications` executed successfully.

**Example 4  Escaping a Command-Line Argument With Single Quote Characters**
The commands in this example specify the backslash (`\`) UNIX operating system meta character and the colon (`:`) option delimiter in the property value `c:\extras\pmdapp`. 
For the UNIX operating system in single mode and multimode, and for all operating systems in multimode, the backslash (\) is required to escape the backslash (\) meta character and the colon (:) option delimiter:

```
asadmin deploy --property extras.home='c:\\extras\\pmdapp' pmdapp.war
```

Application deployed with name pmdapp.
Command deploy executed successfully

For the Windows operating system in single mode, the single quote (') characters eliminate the need for other escape characters:

```
asadmin deploy --property extras.home='c:\extras\pmdapp' pmdapp.war
```

Application deployed with name pmdapp.
Command deploy executed successfully

**Example 5  Specifying a UNIX Operating System Meta Character in an Option**

The commands in this example specify the backslash (\) UNIX operating system meta character in the option argument `Test\Escape\Character`.

For the UNIX operating system in single mode and multimode, and for all operating systems in multimode, the backslash (\) is required to escape the backslash (\) meta character:

```
asadmin --user admin --passwordfile gfpass create-jdbc-connection-pool
--datasourceclassname sampleClassName
--description Test\Escape\Character
sampleJDBCConnectionPool
```

For the Windows operating system in single mode, no escape character is required:

```
asadmin --user admin --passwordfile gfpass create-jdbc-connection-pool
--datasourceclassname sampleClassName
--description Test\Escape\Character
sampleJDBCConnectionPool
```

**Example 6  Specifying a Command-Line Argument That Contains a Space**

The commands in this example specify spaces in the operand `C:\Documents and Settings\gfuser\apps\hello.war`.

For all operating systems in single mode or multimode, the entire operand can be enclosed in double quote (" ) characters:

```
asadmin deploy "C:\Documents and Settings\gfuser\apps\hello.war"
```

Application deployed with name hello.
Command deploy executed successfully.

For the UNIX operating system in single mode and multimode, and for all operating systems in multimode, the entire command-line argument can be enclosed in single quote (') characters:

```
asadmin> deploy 'C:\Documents and Settings\gfuser\apps\hello.war'
```

Application deployed with name hello.
Command deploy executed successfully.

Alternatively, for the UNIX operating system in single mode and multimode, and for all operating systems in multimode, the backslash (\) escape character can be used before each space in the operand. In this situation, the backslash (\) escape character is required before each backslash in the operand:
asadmin> deploy C:Documents and Settingsgfuserappshello.war
Application deployed with name hello.
Command deploy executed successfully.

Example 7  Specifying a Meta Character and an Option Delimiter Character in a Property

The commands in this example specify the backslash (\) UNIX operating system meta character and the colon (:) option delimiter character in the --property option of the create-jdbc-connection-pool subcommand.

The name and value pairs for the --property option are as follows:

user=dbuser
passwordfile=dbpasswordfile
DatabaseName=jdbc:derby
server=http://localhost:9092

For the UNIX operating system in single mode and multimode, and for all operating systems in multimode, a backslash (\) is required to escape the colon (:) and the backslash (\):

```
asadmin --user admin --passwordfile gfpass create-jdbc-connection-pool
--datasourceclassname com.derby.jdbc.jdbcDataSource
--property user=dbuser:passwordfile=dbpasswordfile:
DatabaseName=jdbc:\:derby:server=http\://localhost\:9092 javadb-pool
```

Alternatively, the entire argument to the --property option can be enclosed in single quote (') characters:

```
asadmin --user admin --passwordfile gfpass create-jdbc-connection-pool
--datasourceclassname com.derby.jdbc.jdbcDataSource
--property 'user=dbuser:passwordfile=dbpasswordfile:
DatabaseName="jdbc:derby":server="http://localhost:9092"' javadb-pool
```

For the Windows operating system in single mode, a backslash (\) is required to escape only the colon (:) but not the backslash (\):

```
asadmin --user admin --passwordfile gfpass create-jdbc-connection-pool
--datasourceclassname com.derby.jdbc.jdbcDataSource
--property user-dbuser:passwordfile-dbpasswordfile:
DatabaseName=jdbc:derby:server=http://localhost:9092 javadb-pool
```

For all operating systems, the need to escape the colon (:) in a value can be avoided by enclosing the value in double quote characters or single quote characters:

```
asadmin --user admin --passwordfile gfpass create-jdbc-connection-pool
--datasourceclassname com.derby.jdbc.jdbcDataSource
--property user=dbuser:passwordfile=dbpasswordfile:
DatabaseName="jdbc:derby":server="http://localhost:9092" javadb-pool
```

Example 8  Specifying an Option Delimiter and an Escape Character in a JVM Machine Option

The commands in this example specify the following characters in the -Dlocation=c:sunappserver JVM machine option:

- The colon (:) option delimiter
- The backslash (\) escape character
For the UNIX operating system in single mode and multimode, and for all operating systems in multimode, these characters must be specified as follows:

- To pass a literal backslash to a subcommand, two backslashes are required. Therefore, the colon (:) must be escaped by two backslashes (\ ).
- To prevent the subcommand from treating the backslash as a special character, the backslash must be escaped. As a result, two literal backslashes (\ ) must be passed to the subcommand. To prevent the shell from interpreting these backslashes as special characters, each backslash must be escaped. Therefore, the backslash must be specified by a total of four backslashes (\\ ).

The resulting command is as follows:

```
asadmin create-jvm-options --target test-server
  -e -Dlocation=c:\\\sun\\appserver
```

For the Windows operating system in single mode, a backslash (\ ) is required to escape the colon (:) and the backslash (\ ):

```
asadmin create-jvm-options --target test-server
  -e -Dlocation=c:\\sun\appserver
```

### Example 9  Specifying an Option That Contains an Escape Character

The commands in this example specify the backslash (\ ) character and the double quote (" ) characters in the "Hello\App\" \authentication option argument.

For the UNIX operating system in single mode and multimode, and for all operating systems in multimode, a backslash (\ ) is required to escape the double quote character (" ) and the backslash (\ ):

```
asadmin set-web-env-entry --name="Hello User" --type=java.lang.String
  --value=techscribe --description="Hello\App\"\authentication hello
```

For the Windows operating system in single mode, a backslash (\ ) is required to escape only the double quote ("), but not the backslash (\ ):

```
asadmin set-web-env-entry --name="Hello User" --type=java.lang.String
  --value=techscribe --description="Hello\App\"\authentication hello
```

### Environment Variables

Environment variables modify the default values of asadmin utility options as shown in the following table.

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>asadmin Utility Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS_ADMIN_TERSE</td>
<td>--terse</td>
</tr>
<tr>
<td>AS_ADMIN_ECHO</td>
<td>--echo</td>
</tr>
<tr>
<td>AS_ADMIN_INTERACTIVE</td>
<td>--interactive</td>
</tr>
<tr>
<td>AS_ADMIN_HOST</td>
<td>--host</td>
</tr>
<tr>
<td>AS_ADMIN_PORT</td>
<td>--port</td>
</tr>
<tr>
<td>AS_ADMIN_SECURE</td>
<td>--secure</td>
</tr>
<tr>
<td>AS_ADMIN_USER</td>
<td>--user</td>
</tr>
<tr>
<td>AS_ADMIN_PASSWORDFILE</td>
<td>--passwordfile</td>
</tr>
</tbody>
</table>
Attributes

See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Stability</td>
<td>Unstable</td>
</tr>
</tbody>
</table>

See Also

capture-schema

stores the database metadata (schema) in a file for use in mapping and execution

Synopsis

capture-schema -username name -password password
[-dburl url] [-driver jdbc_driver_classname]
[-schemaname schemaname] [-table tablename]
-out filename

Description

Stores the database metadata (schema) in a file.

Run capture-schema as the same database user that owns the table(s), and use that same username with the -username option (and -schemaname, if required).

When running capture-schema against an Oracle database, you should grant the database user running the capture-schema command the ANALYZE ANY TABLE privilege.

Options

-username
user name for authenticating access to a database.

-password
password for accessing the selected database.

-dburl
JDBC URL required by the driver for accessing a database.

-driver
JDBC driver classname in your CLASSPATH.

-schemaname
name of the user schema being captured. If not specified, the default will capture metadata for all tables from all the schemas accessible to this user.

Specifying this parameter is highly recommended. Without this option, if more than one schema is accessible to this user, more than one table with the same name may be captured, which will cause problems when mapping CMP fields to tables.

The specified schema name must be uppercase.

-table
name of a table; multiple table names can be specified. If no table is specified, all the tables in the database or named schema are captured.

The specified table name or names are case sensitive. Be sure to match the case of the previously created table names.
-out
name of the output file. This option is required. If the specified output file does not contain the .dbschema suffix, it will be appended to the filename.

Examples

Example 1 Using the capture-schema command

capture-schema -username cantiflas -password enigma
-dburl jdbc:oracle:thin:@sadbuttrue:1521:ora817
-driver oracle.jdbc.driver.OracleDriver
-schemaname CANTIFLAS -out cantiflas.dbschema

Where: sun-acc.xml is the name of the client configuration XML file, myclientapp.jar is the client application .jar file, and scott and sample are arguments to pass to the application. If sun-acc.xml and myclientapp.jar are not in the current directory, you must give the absolute path locations; otherwise the relative paths are used. The relative path is relative to the directory where the command is being executed.

See Also

asadmin(1M)
package-appclient

packs the application client container libraries and jar files

Synopsis

package-appclient

Description

Use the package-appclient command to pack the application client container libraries and jar files into an appclient.jar file, which is created in the as-install/lib directory on the machine on which GlassFish Server is installed. The appclient.jar file provides an application client container package targeted at remote hosts that do not contain a server installation.

After copying the appclient.jar file to a remote location, unjar it to get a set of libraries and jar files in the appclient directory under the current directory.

After unjarring on the client machine, modify appclient/glassfish/config/asenv.conf (asenv.bat for Windows) as follows:

- set AS_WEBSERVICES_LIB to path-to-appclient/appclient/lib
- set AS_IMQ_LIB to path-to-appclient/appclient/mq/lib
- set AS_INSTALL to path-to-appclient/appclient
- set AS_JAVA to your JDK 1.6 home directory
- set AS_ACC_CONFIG to path-to-appclient/appclient/glassfish/domains/domain1/config/sun-acc.xml

Modify appclient/glassfish/domains/domain1/config/sun-acc.xml as follows:

- Ensure the DOCTYPE file references
  path-to-appclient/appclient/glassfish/lib/dtds
- Ensure that target-server address attribute references the server machine.
- Ensure that target-server port attribute references the ORB port on the remote machine.
- Ensure that log-service references a log file; if the user wants to put log messages to a log file.

To use the newly installed application client container, you must do the following:

- Obtain the application client files for your target application, including the generated yourAppClient.jar file.
- Execute the appclient utility: appclient -client yourAppClient.jar

Attributes

See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Stability</td>
<td>Unstable</td>
</tr>
</tbody>
</table>
See Also

appclient(1M)
This section describes concepts that are related to GlassFish Server administration.
application

server-side Java applications and web services

Description

The Java EE platform enables applications to access systems that are outside of the application server. Applications connect to these systems through resources. The GlassFish Server infrastructure supports the deployment of many types of distributed applications and is an ideal foundation for building applications based on Service Oriented Architectures (SOA). SOA is a design methodology aimed at maximizing the reuse of application services. These features enable you to run scalable and highly available Java EE applications.

See Also

create-application-ref(1), deploy(1), list-applications(1)
configuration

the data set that determines how GlassFish Server operates

Description

The configuration of GlassFish Server is the data set that determines how it operates. Parts of this configuration determine the operation of specific parts of GlassFish Server, such as the following:

- Services, such as the transaction service
- Resources, such as databases
- Deployed applications or modules, such as web applications
- Clusters and server instances

The term configuration is also used to describe a part of the overall configuration, such as the transaction service configuration or the configuration of a database. In clustered environments, clusters or server instances can share configurations.

Examples of configuration data are port numbers, flags that enable or disable processes, application names, and so on. Most of these data points are name/value pairs, either hard-coded attributes or more flexibly defined properties.

The hierarchical structure of the configuration is explained in the dotted names page.

You can view and change most of the GlassFish Server configuration using either the Administration Console or the asadmin utility and its subcommands. To list the structure of all or part of the configuration, use the list subcommand. To view the value of one or more attributes or properties, use the get subcommand. To change the value of an attribute or property, use the set subcommand.

See Also

get(1), list(1), set(1)
asadmin(1M)
dotted-names(5ASC)
"Configuration Tasks" in Oracle GlassFish Server Administration Guide
domain

Domains have their own configurations.

Description

A domain provides a common authentication and administration point for a collection of zero or more server instances. The administration domain encompasses several manageable resources, including instances, clusters, and their individual resources. A manageable resource, such as a server instance, may belong to only one domain.

See Also

asadmin(1M)
dotted-names

syntax for using periods to separate name elements

Description

A dotted name is an identifier for a particular GlassFish Server element, such as a configurable or a monitorable object. A dotted name uses the period (.), known as dot, as a delimiter to separate the parts of an element name. The period in a dotted name is similar to the slash (/) character that delimits the levels in the absolute path name of a file in the UNIX file system.

The subcommands of the asadmin utility use dotted names as follows:

- The list subcommand provides the fully qualified dotted names of the management components' attributes.
- The get subcommand provides access to the attributes.
- The set subcommand enables you to modify configurable attributes and set properties.

The configuration hierarchy is loosely based on the domain's schema document, and the attributes are modifiable. The attributes of the monitoring hierarchy are read-only.

The following format is used for configuration dotted names (italic indicates replaceable):

- `config-name.config-element-name.primary-key.attribute-name | instance-name.config-element-name.primary-key.attribute-name`

The following format is used for resource dotted names (italic indicates replaceable):

- `server-name.resource-name.primary-key.attribute-name | domain.resources.resource-name.primary-key.attribute-name`

The following rules apply to forming dotted names:

- The top-level is configuration, server, or domain name. For example, server-config (default configuration), server (default server), or domain1 (default domain).
- A dot (.) always separates two sequential parts of the name.
- A part of the name usually identifies a server subsystem or its specific instance. For example, web-container, log-service, thread-pool-1.
- If any part of the name itself contains a dot (.), then the dot must be escaped with a leading \ (backslash) so that the . (dot) does not act like a delimiter. For further information on escape characters, see the asadmin(1M) help page.
- An * (asterisk) character can be used anywhere in the dotted name and acts like the wildcard character in regular expressions. Additionally, an * can collapse all the parts of the dotted name. For example, a long dotted name such as `this.is.really.long.hierarchy` can be abbreviated to `th*.hierarchy`. The . (dot) always delimits the parts of the dotted name.
The --monitor option of the get and list subcommands selects the monitoring or configuration hierarchy. If the subcommand specifies --monitor=false (the default), the configuration hierarchy is selected. If the subcommand specifies --monitor=true, the monitoring hierarchy is selected.

If you know the complete dotted name and do not need to use a wildcard, the list, get, and set subcommands treat the name differently:

- The list subcommand treats a complete dotted name as the name of a parent node in the abstract hierarchy. When you specify this name to the list subcommand, the names of the immediate children at that level are returned. For example, the following command lists all the web modules deployed to the domain or the default server:

```
asadmin> list server.applications.web-module
```

- The get and set subcommands treat a complete dotted name as the fully qualified name of the attribute of a node (whose dotted name itself is the name that you get when you remove the last part of this dotted name). When you specify this name to the get or set subcommand, the subcommand acts on the value of that attribute, if such an attribute exists. You will never start with this case because in order to find out the names of attributes of a particular node in the hierarchy, you must use the * wildcard character. For example, the following dotted name returns the context root of the web application deployed to the domain or default server:

```
server.applications.web-module.JSPWiki.context-root
```
Example 2 Listing All the Monitorable Objects
The following example lists all the monitorable objects.

```
asadmin> list --monitor *
```

Output similar to the following is displayed:

```
server
server.jvm
server.jvm.class-loading-system
server.jvm.compilation-system
server.jvm.garbage-collectors
server.jvm.garbage-collectors.Copy
server.jvm.garbage-collectors.MarkSweepCompact
server.jvm.memory
server.jvm.operating-system
server.jvm.runtime
server.network
server.network.admin-listener
server.network.admin-listener.connections
server.network.admin-listener.file-cache
server.network.admin-listener.keep-alive
server.network.admin-listener.thread-pool
```
server.network.http-listener-1
server.network.http-listener-1.connections
server.network.http-listener-1.file-cache
server.network.http-listener-1.keep-alive
server.network.http-listener-1.thread-pool
server.transaction-service
Command list executed successfully.

See Also

list(1), get(1), set(1)
asadmin(1M)
an instance in GlassFish Server has its own Java EE configuration, Java EE resources, application deployment areas, and server configuration settings

Description

GlassFish Server creates one server instance, called server at the time of installation. You can delete the server instance and create a new instance with a different name.

For many users, one server instance meets their needs. However, depending upon your environment, you might want to create additional server instances. For example, in a development environment you can use different server instances to test different GlassFish Server configurations, or to compare and test different application deployments. Because you can easily add or delete a server instance, you can use them to create temporary "sandbox" areas to experiment with while developing.
logging

capturing information on GlassFish Server runtime events

Description

Logging is the process by which GlassFish Server captures data about events that occur during GlassFish Server operation. GlassFish Server components and application components generate logging data, which is saved in the server log, typically `domain-dir/logs/server.log`. The server log is the first source of information if GlassFish Server problems occur.

The server log is rotated when the file reaches the specified size in bytes, or the specified time has elapsed. The file can also be rotated manually by using the `rotate-log` subcommand.

In addition to the server log, the `domain-dir/logs` directory contains two other kinds of logs:

- HTTP service access logs, located in the `/access` subdirectory
- Transaction service logs, located in the `/tx` subdirectory

Logging levels can be configured by using the Administration Console or the `set-log-levels` subcommand. Additional properties can be set by using the Administration Console or by editing the `logging.properties` file. The default `logging.properties` file is typically located in `domain-dir/config`.

Although application components can use the Apache Commons Logging Library to record messages, the platform standard JSR 047 API is recommended for better log configuration.

See Also

`list-log-levels(1), rotate-log(1), set-log-levels(1)`, `asadmin(1M)`

"Administering the Logging Service" in *Oracle GlassFish Server Administration Guide"
monitoring

reviewing the runtime state of components and services deployed in GlassFish Server

Description

Monitoring is the process of reviewing the statistics of a system to improve performance or solve problems. By monitoring the state of various components and services deployed in GlassFish Server, performance bottlenecks can be identified, failures can be anticipated, and runtime standards can be established and observed. Data gathered by monitoring can also be useful in performance tuning and capacity planning.

The GlassFish Server monitoring service is enabled by default, that is, the monitoring-enabled attribute of the monitoring-service element is set to true. Once the monitoring service is enabled, a deployed module can then be enabled for monitoring by setting its monitoring level to HIGH or LOW (default is OFF). Monitoring can be configured dynamically by using the Administration Console or the enable-monitoring and the disable-monitoring subcommands. The set subcommand can also be used with dotted names to enable or disable monitoring. However, a server restart is required for changes made by using the set subcommand to take affect.

Monitoring data can be viewed by using the Administration Console or by using the subcommands of the asadmin utility.

- The monitor subcommand displays monitoring data for a given type, similar to the UNIX top command. The data is presented at given intervals.
- The list and get subcommands display comprehensive data. Both use dotted names to specify monitorable objects.

Alternate tools for monitoring GlassFish Server components and services include JConsole and the REST interface.

The Monitoring Scripting Client or DTrace Monitoring can be used to start the available monitoring probes. Using these tools is helpful in identifying performance issues during runtime. Monitoring Scripting Client or DTrace Monitoring are only usable if their modules are present.

See Also

monitor(1), enable-monitoring(1), disable-monitoring(1), list(1), get(1), set(1)
dotted-names(5ASC)
asadmin(1M)

"Administering the Monitoring Service” in Oracle GlassFish Server Administration Guide
passwords

securing and managing GlassFish Server

Description

An administrator of GlassFish Server manages one or more domains, each of which can have distinct administrative credentials. By managing a domain, an administrator effectively manages various resources like server instances, server clusters, libraries etc. that are required by the enterprise Java applications.

See Also

change-admin-password(1), change-master-password(1), create-password-alias(1), list-password-aliases(1), delete-password-alias(1)

asadmin(1M)
resource

Provide connectivity to various types of EIS.

Description

GlassFish Server provides support for JDBC, JMS, and JNDI resources.

See Also

asadmin(1M)
security

secure and administer GlassFish Server applications

Description

Security is about protecting data: how to prevent unauthorized access or damage to it in storage or transit. GlassFish Server has a dynamic, extensible security architecture based on the Java EE standard. Built in security features include cryptography, authentication and authorization, and public key infrastructure. GlassFish Server is built on the Java security model, which uses a sandbox where applications can run safely, without potential risk to systems or users.

See Also

cchange-admin-password(1), change-master-password(1), create-auth-realm(1),
ccreate-file-user(1), create-message-security-provider(1),
ccreate-password-alias(1), create-ssl(1), delete-auth-realm(1),
delete-file-user(1), delete-message-security-provider(1),
delete-password-alias(1), delete-ssl(1), list-auth-realms(1),
list-connector-security-maps(1), list-file-groups(1), list-file-users(1).
asadmin(1M)
This section describes events that provide monitoring data for GlassFish Server.
generic-probe

generic event description

Description

No detailed information is available for this event.
The event signature contains information that you can use in JavaScript programs that you write for monitoring GlassFish Server.
An event signature consists of the event identifier (ID) followed by a list of the event's parameters in the following format:

```
event-id(param-type param-name[,param-type param-name]...)```

The replaceable items in this format are as follows:

- **event-id**
  The event ID, which uniquely identifies the event.

- **param-type**
  The type of the event parameter. This type is a Java language primitive, such as int, boolean, or java.lang.String.

- **param-name**
  The name of the event parameter.

The format of an event ID is as follows:

```
module-provider:module:submodule:event```

The replaceable items in this format are as follows:

- **module-provider**
  Text that identifies the application that is source of the event. For example, for events from Oracle GlassFish Server, **module-provider** is glassfish.

- **module**
  The name of the module for which the event is defined. A module provides significant functionality of GlassFish Server. Examples of module names are web-container, ejb-container, transaction, and webservices.

- **submodule**
  The submodule of **module** for which the event is defined, for example, web-module.

- **event-type**
  The type of the event, for example, webModuleStartedEvent.
**Synopsis**

glassfish:connector-pool:applications:connectionAcquiredEvent(
java.lang.String poolName,
java.lang.String applicationName)

**Description**

This event is sent when an application acquires a connection from a connector connection pool.

Use this event to count the number of logical connections an application has acquired from a pool since the last sampling.

**Parameters**

- **poolName**
  The name of the connector connection pool from which the connection is acquired.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:connector-pool:applications:connectionReleasedEvent**

connection returned by application to connector connection pool event

**Synopsis**

glassfish:connector-pool:applications:connectionReleasedEvent(
java.lang.String poolName,
java.lang.String applicationName)

**Description**

This event is sent when an application releases a connection, returning it to the
connector connection pool.

Use this event to count the number of logical connections an application releases to a
pool.

**Parameters**

- **poolName**
  The name of the connector connection pool to which the connection is returned.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is
  applicable only for application-scoped resources.
**Synopsis**

glassfish:connector-pool:applications:connectionUsedEvent(
java.lang.String poolName,
java.lang.String applicationName)

**Description**

This event is sent whenever an application uses a connection in the connector connection pool.

Use this event to get the total number of connections that are currently being used by an application.

**Parameters**

- **poolName**
  The name of the connector connection pool that contains the connection being used.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
decrease in connections used event

Synopsis

glassfish:connector-pool:applications:decrementConnectionUsedEvent(
java.lang.String poolName,
java.lang.String applicationName)

Description

This event is sent whenever an application destroys a connection or releases a connection back to the connector connection pool. This event indicates that the number of connections in use has decreased by 1.

Use this event to calculate the total number of connections that are currently being used by an application.

Parameters

  poolName
  The name of the connector connection pool that contains the destroyed or released connection.

  applicationName
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

glassfish:deployment:lifecycle:applicationDeployedEvent

```java
glassfish:deployment:lifecycle:applicationDeployedEvent (  
    String appName,  
    String appType,  
    String loadTime)
```

**Description**

This event is sent when an application is deployed. Use this event to count the total number of deployed applications, count the number of currently deployed applications, and provide information about the deployed applications (the application name, the application type, and the time taken for this application to load).

**Parameters**

- **appName**
  The name of the deployed application.

- **appType**
  The type of the application that is deployed, for example, web or ejb.

- **loadTime**
  The time taken for this application to load.
**glassfish:deployment:lifecycle:applicationUndeployedEvent**

application undeployed event

**Synopsis**

glassfish:deployment:lifecycle:applicationUndeployedEvent (String appName, String appType)

**Description**

This event is sent when an application is undeployed. Use this event to count the total number of deployed applications and the number of the currently deployed applications.

**Parameters**

- **appName**
  The name of the undeployed application.

- **appType**
  The type of the application that is undeployed, for example, web or ejb.
**glassfish:ejb:bean:beanCreatedEvent**

EJB create method called event

**Synopsis**

```java
glassfish:ejb:bean:beanCreatedEvent {
    long beanId,
    String appName,
    String modName,
    String ejbName)
```

```java
glassfish:ejb:bean__beanspecific__:beanCreatedEvent {
    long beanId,
    String appName,
    String modName,
    String ejbName)
```

**Description**

These two events are sent when the EJB create method is called.

Use the generic version of this event to track EJB create method invocations across the entire EJB container.

Use the bean-specific version of this event to track EJB create method invocations by the EJB container of a specific bean. In the bean-specific version, `beanspecific` has the form:

```
appName_modName_ejbName
```

In this form, `appName`, `modName`, and `ejbName` are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the `sampleBean` bean in the `sampleModule.jar` module of the `sample-App` application is:

```
glassfish:ejb:bean__sample_App_sampleModule_jar_sampleBean__:beanCreatedEvent
```

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

- **beanId**
  The unique identifier for the EJB container associated with this EJB.

- **appName**
  The name of the application associated with the EJB.

- **modName**
  The name of the module that contains the EJB.

- **ejbName**
  The name of the EJB.
glassfish:ejb:bean:beanDestroyedEvent

EJB remove method called event

Synopsis

glassfish:ejb:bean:beanDestroyedEvent (long beanId, String appName, String modName, String ejbName)

glassfish:ejb:bean__beanspecific_:beanDestroyedEvent (long beanId, String appName, String modName, String ejbName)

Description

These two events are sent when the EJB remove method is called.
Use the generic version of this event to track EJB remove method invocations across the entire EJB container.
Use the bean-specific version of this event to track EJB remove method invocations by the EJB container of a specific bean. In the bean-specific version, beanspecific has the form:

appName_modName_ejbName

In this form, appName, modName, and ejbName are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the sampleBean bean in the sampleModule.jar module of the sample-App application is:

glassfish:ejb:bean__sample_App_sampleModule_jar_sampleBean_:beanDestroyedEvent

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

Parameters

beanId
The unique identifier for the EJB container associated with this EJB.

appName
The name of the application associated with the EJB.

modName
The name of the module that contains the EJB.

ejbName
The name of the EJB.
glassfish:ejb:bean:containerEnteringEvent

EJB container enter event

Synopsis

```java
glassfish:ejb:bean:containerEnteringEvent (  
long beanId,  
String appName,  
String modName,  
String ejbName)
```

```java
glassfish:ejb:bean__beanspecific__:containerEnteringEvent (  
long beanId,  
String appName,  
String modName,  
String ejbName)
```

Description

These two events are sent when the execution stack enters EJB container code. Use the generic version of this event to track time spent in the EJB container (including calls to other internal modules) across the entire EJB container.

Use the bean-specific version of this event to track time spent in the EJB container (including calls to other internal modules) arising from a specific bean. In the bean-specific version, `beanspecific` has the form:

```
appName_modName_ejbName
```

In this form, `appName`, `modName`, and `ejbName` are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the `sampleBean` bean in the `sampleModule.jar` module of the `sample-App` application is:

```
glassfish:ejb:bean__sample_App_sampleModule_jar_sampleBean__:containerEnteringEvent
```

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

Parameters

- **beanId**
  The unique identifier for the EJB container associated with this EJB.

- **appName**
  The name of the application associated with the EJB.

- **modName**
  The name of the module that contains the EJB.

- **ejbName**
  The name of the EJB.
**Synopsis**

glassfish:ejb:bean:containerLeavingEvent (  
long beanId,  
String appName,  
String modName,  
String ejbName)

glassfish:ejb:bean__beanspecific__:containerLeavingEvent (  
long beanId,  
String appName,  
String modName,  
String ejbName)

**Description**

These two events are sent when the execution stack leaves EJB container code. Use the generic version of this event to track time spent in the EJB container (including calls to other internal modules) across the entire EJB container. Use the bean-specific version of this event to track time spent in the EJB container (including calls to other internal modules) arising from a specific bean. In the bean-specific version, beanspecific has the form:

appName_modName_ejbName

In this form, appName, modName, and ejbName are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the sampleBean bean in the sampleModule.jar module of the sample-App application is:

glassfish:ejb:bean__sample_App_sampleModule_jar_sampleBean__:containerLeavingEvent

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

**beanId**  
The unique identifier for the EJB container associated with this EJB.

**appName**  
The name of the application associated with the EJB.

**modName**  
The name of the module that contains the EJB.

**ejbName**  
The name of the EJB.
**Synopsis**

```java
glassfish:ejb:bean:messageDeliveredEvent {
  long beanId,
  String appName,
  String modName,
  String ejbName)

glassfish:ejb:bean__beanspecific_:messageDeliveredEvent {
  long beanId,
  String appName,
  String modName,
  String ejbName)
```

**Description**

These two events are sent when the message had been successfully delivered. Use the generic version of this event to track successful message deliveries across the entire EJB container.

Use the bean-specific version of this event to track successful message deliveries for a specific bean. In the bean-specific version, `beanspecific` has the form:

`appName_modName_ejbName`

In this form, `appName`, `modName`, and `ejbName` are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the `sampleBean` bean in the `sampleModule.jar` module of the `sample-App` application is:

`glassfish:ejb:bean__sample_App_sampleModule_jar_sampleBean_:messageDeliveredEvent`

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

- **beanId**
  The unique identifier for the EJB container associated with this EJB.

- **appName**
  The name of the application associated with the EJB.

- **modName**
  The name of the module that contains the EJB.

- **ejbName**
  The name of the EJB.
**Synopsis**

```java
glassfish:ejb:bean:methodEndEvent {
    long beanId,
    String appName,
    String modName,
    String ejbName,
    Throwable exception,
    Method method
}

glassfish:ejb:bean__beanspecific__:methodEndEvent {
    long beanId,
    String appName,
    String modName,
    String ejbName,
    Throwable exception,
    Method method
}
```

**Description**

These two events are sent when the EJB container code finished executing a bean method.

Use the generic version of this event to track EJB methods executed by the EJB container across the entire EJB container.

Use the bean-specific version of this event to track EJB methods executed by the EJB container for a specific bean. In the bean-specific version, `beanspecific` has the form:

`appName_modName_ejbName`

In this form, `appName`, `modName`, and `ejbName` are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the `sampleBean` bean in the `sampleModule.jar` module of the `sample-App` application is:

`glassfish:ejb:bean__sample_App_sampleModule_jar_sampleBean__:methodEndEvent`

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

- **beanId**
  The unique identifier for the EJB container associated with this EJB.

- **appName**
  The name of the application associated with the EJB.
**modName**
The name of the module that contains the EJB.

**ejbName**
The name of the EJB.

**exception**
The Throwable if the exception had been thrown by this method. Null if the method finished successfully.

**method**
The Method object.
**Synopsis**

glassfish:ejb:bean:methodReadyAddEvent {
  long beanId,
  String appName,
  String modName,
  String ejbName)

glassfish:ejb:bean__beanspecific__:methodReadyAddEvent {
  long beanId,
  String appName,
  String modName,
  String ejbName)

**Description**

These two events are sent when the stateful session bean is moved to the ready state. Use the generic version of this event to track stateful session beans in the ready state across the entire EJB container. Use the bean-specific version of this event to track a specific stateful session bean being in the ready state. In the bean-specific version, beanspecific has the form:

appName_modName_ejbName

In this form, appName, modName, and ejbName are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the sampleBean bean in the sampleModule.jar module of the sample-App application is:

glassfish:ejb:bean__sample_App_sampleModule_jar_sampleBean__:methodReadyAddEvent

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

**beanId**
The unique identifier for the EJB container associated with this EJB.

**appName**
The name of the application associated with the EJB.

**modName**
The name of the module that contains the EJB.

**ejbName**
The name of the EJB.
**Synopsis**

glassfish:ejb:bean:methodReadyRemoveEvent (long beanId, String appName, String modName, String ejbName)

glassfish:ejb:bean__:beanspecific__:methodReadyRemoveEvent (long beanId, String appName, String modName, String ejbName)

**Description**

These two events are sent when the stateful session bean is moved from the ready state.

Use the generic version of this event to track stateful session beans in the ready state across the entire EJB container.

Use the bean-specific version of this event to track a specific stateful session bean being in the ready state. In the bean-specific version, beanspecific has the form:

appName__modName__ejbName

In this form, appName, modName, and ejbName are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the sampleBean bean in the sampleModule.jar module of the sample-App application is:

glassfish:ejb:bean__:sample_App__sampleModule__jar__sampleBean__:methodReadyRemoveEvent

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

**beanId**
The unique identifier for the EJB container associated with this EJB.

**appName**
The name of the application associated with the EJB.

**modName**
The name of the module that contains the EJB.

**ejbName**
The name of the EJB.
### glassfish:ejb:bean:methodStartEvent

start of the method execution event

**Synopsis**

```java
glassfish:ejb:bean:methodStartEvent {
    long beanId,
    String appName,
    String modName,
    String ejbName,
    Method method
}

glassfish:ejb:bean__beanspecific_:methodStartEvent {
    long beanId,
    String appName,
    String modName,
    String ejbName,
    Method method
}
```

**Description**

These two events are sent when the EJB container code starts executing a bean method. Use the generic version of this event to track EJB methods executed across the entire EJB container.

Use the bean-specific version of this event to track EJB methods executed for a specific bean. In the bean-specific version, `beanspecific` has the form:

`appName_modName_ejbName`

In this form, `appName`, `modName`, and `ejbName` are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the `sampleBean` bean in the `sampleModule.jar` module of the `sample-App` application is:

`glassfish:ejb:bean__sample_App_sampleModule_jar_sampleBean_:methodStartEvent`

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

- **beanId**
  The unique identifier for the EJB container associated with this EJB.

- **appName**
  The name of the application associated with the EJB.

- **modName**
  The name of the module that contains the EJB.
`ejbName`
The name of the EJB.

`method`
The Method object.
**Synopsis**

glassfish:ejb:cache:beanPassivatedEvent (long beanId, 
String appName, 
String modName, 
String ejbName, 
boolean success)

glassfish:ejb:cache__beanspecific_:beanPassivatedEvent (long beanId, 
String appName, 
String modName, 
String ejbName, 
boolean success)

**Description**

These two events are sent when an object is passivated by the EJB container.

Use the generic version of this event to track the number of bean passivations across the entire EJB container and to track the number of successful or failed passivations across the entire EJB container.

Use the bean-specific version of this event to track the number of bean passivations of a specific bean and to track the number of successful or failed passivations of a specific bean. In the bean-specific version, bean-specific has the form:

```
appName_modName_ejbName
```

In this form, appName, modName, and ejbName are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the sampleBean bean in the sampleModule.jar module of the sample-App application is:

```
glassfish:ejb:cache__sample_App_sampleModule_jar_sampleBean_:beanPassivatedEvent
```

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

- **beanId**
  The unique identifier for the EJB container associated with this EJB.

- **appName**
  The name of the application associated with the EJB.

- **modName**
  The name of the module that contains the EJB.
**ejbName**
The name of the EJB.

**success**
Has the value `true` if the bean was passivated successfully.
**glassfish:ejb:cache:expiredSessionsRemovedEvent**

session expired event

**Synopsis**

```java
glassfish:ejb:cache:expiredSessionsRemovedEvent (long beanId, String appName, String modName, String ejbName, long num)

glassfish:ejb:cache__beanspecific__:expiredSessionsRemovedEvent (long beanId, String appName, String modName, String ejbName, long num)
```

**Description**

These two events are sent when sessions are removed by the cleanup thread.

Use the generic version of this event to track the number of sessions removed by the cleanup thread across the entire EJB container.

Use the bean-specific version of this event to track the number of sessions removed by the cleanup thread for a specific bean. In the bean-specific version, `beanspecific` has the form:

```java
appName_modName_ejbName
```

In this form, `appName`, `modName`, and `ejbName` are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the `sampleBean` bean in the `sampleModule.jar` module of the `sample-App` application is:

```java
glassfish:ejb:cache__sample_App_sampleModule_jar_sampleBean__:expiredSessionsRemovedEvent
```

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

- **beanId**
  The unique identifier for the EJB container associated with this EJB.

- **appName**
  The name of the application associated with the EJB.

- **modName**
  The name of the module that contains the EJB.
**ejbName**
The name of the EJB.

**num**
The number of sessions removed.
**Synopsis**

```
glassfish:ejb:pool:objectAddedEvent {
    long beanId,
    String appName,
    String modName,
    String ejbName)
```

```
glassfish:ejb:pool__beanspecific__:objectAddedEvent (
    long beanId,
    String appName,
    String modName,
    String ejbName)
```

**Description**

These two events are sent when an object is added to the EJB pool.

Use the generic version of this event to track objects in the EJB pool across the entire EJB container.

Use the bean-specific version of this event to track objects in the EJB pool for a specific bean. In the bean-specific version, `beanspecific` has the form:

```
appName_modName_ejbName
```

In this form, `appName`, `modName`, and `ejbName` are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the sampleBean bean in the sampleModule.jar module of the sample-App application is:

```
glassfish:ejb:pool__sample_App_sampleModule_jar_sampleBean__:objectAddedEvent
```

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

- **beanId**
  The unique identifier for the EJB container associated with this EJB.

- **appName**
  The name of the application associated with the EJB.

- **modName**
  The name of the module that contains the EJB.

- **ejbName**
  The name of the EJB.
**Synopsis**

glassfish:ejb:pool:objectAddFailedEvent (  
  long beanId,  
  String appName,  
  String modName,  
  String ejbName)

glassfish:ejb:pool__beanspecific__:objectAddFailedEvent (  
  long beanId,  
  String appName,  
  String modName,  
  String ejbName)

**Description**

These two events are sent when an object failed to be added to the EJB pool.

Use the generic version of this event to track objects in the EJB pool across the entire EJB container.

Use the bean-specific version of this event to track objects in the EJB pool for a specific bean. In the bean-specific version, `beanspecific` has the form:

```
appName_modName_ejbName
```

In this form, `appName`, `modName`, and `ejbName` are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the `sampleBean` bean in the `sampleModule.jar` module of the `sample-App` application is:

```
glassfish:ejb:pool__sample_App_sampleModule_jar_sampleBean_:objectAddFailedEvent
```

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

- **beanId**
The unique identifier for the EJB container associated with this EJB.

- **appName**
The name of the application associated with the EJB.

- **modName**
The name of the module that contains the EJB.

- **ejbName**
The name of the EJB.
**Synopsis**

```
glassfish:ejb:pool:objectDestroyedEvent (long beanId, String appName, String modName, String ejbName)
glassfish:ejb:pool__:beanspecific__:objectDestroyedEvent (long beanId, String appName, String modName, String ejbName)
```

**Description**

These two events are sent when an object in the EJB pool had been destroyed.

Use the generic version of this event to track objects in the EJB pool across the entire EJB container.

Use the bean-specific version of this event to track objects in the EJB pool for a specific bean. In the bean-specific version, `beanspecific` has the form:

```
appName_modName_ejbName
```

In this form, `appName`, `modName`, and `ejbName` are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the `sampleBean` bean in the `sampleModule.jar` module of the `sample-App` application is:

```
glassfish:ejb:pool__:sample_App_sampleModule_jar_sampleBean__:objectDestroyedEvent
```

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

- **beanId**
  The unique identifier for the EJB container associated with this EJB.

- **appName**
  The name of the application associated with the EJB.

- **modName**
  The name of the module that contains the EJB.

- **ejbName**
  The name of the EJB.
**glassfish:ejb:timers:timerCreatedEvent**

a new EJB Timer created event

**Synopsis**

glassfish:ejb:timers:timerCreatedEvent ()

glassfish:ejb:timers__beanspecific__:timerCreatedEvent ()

**Description**

These two events are sent when a new EJB Timer is created.

Use the generic version of this event to track EJB Timers created across the entire EJB container.

Use the bean-specific version of this event to track EJB Timers created by the EJB container for a specific bean. In the bean-specific version, beanspecific has the form:

```
appName_modName_ejbName
```

In this form, appName, modName, and ejbName are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the sampleBean bean in the sampleModule.jar module of the sample-App application is:

```
glassfish:ejb:timers__sample_App_sampleModule_jar_sampleBean__:timerCreatedEvent
```

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

None.
**glassfish:ejb:timers:timerDeliveredEvent**

an EJB timeout successful delivery event

**Synopsis**

```
玻璃fish:ejb:timers:timerDeliveredEvent ()
glassfish:ejb:timers__:beanspecific__:timerDeliveredEvent ()
```

**Description**

These two events are sent when an EJB Timeout is successfully delivered.

Use the generic version of this event to track successful EJB timeouts across the entire EJB container.

Use the bean-specific version of this event to track successful EJB timeouts for a specific bean. In the bean-specific version, `beanspecific` has the form:

```
appName_modName_ejbName
```

In this form, `appName`, `modName`, and `ejbName` are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the `sampleBean` bean in the `sampleModule.jar` module of the `sample-App` application is:

```
glassfish:ejb:timers__:sample_App_sampleModule_jar_sampleBean__:timerDeliveredEvent
```

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

None.
**glassfish:ejb:timers:timerRemovedEvent**

an EJB Timer removed event

**Synopsis**

glassfish:ejb:timers:timerRemovedEvent ()
glassfish:ejb:timers__beanspecific_:timerRemovedEvent ()

**Description**

These two events are sent when an EJB Timer is removed.

Use the generic version of this event to track EJB Timers removed across the entire EJB container.

Use the bean-specific version of this event to track EJB Timers removed by the EJB container for a specific bean. In the bean-specific version, **beanspecific** has the form:

appName_modName_ejbName

In this form, **appName**, **modName**, and **ejbName** are the names of the application, the module, and the bean, respectively, with dashes (-) and periods (.) converted to underscores (_). For example, the bean-specific event name for the sampleBean bean in the sampleModule.jar module of the sample-App application is:

glassfish:ejb:timers__sample_App_sampleModule_jar_sampleBean_:timerRemovedEvent

When monitoring the activity of a specific bean, using the bean-specific version of this event provides much better end-to-end monitoring performance than does using the generic version and inspecting its parameters to associate the event with the bean.

**Parameters**

None.
**glassfish:javamail:iap-protocol:commandEnd**

finished processing of IMAP command event

**Synopsis**

glassfish:javamail:iap-protocol:commandEnd ()

**Description**

This event is sent when the JavaMail service finishes processing a low-level Internet Message Access Protocol (IMAP) command.

Use this event with the glassfish:javamail:iap-protocol:commandStart event to measure the amount of time that the JavaMail service spends communicating with the IMAP server.

**Parameters**

None.
**glassfish:javamail:iap-protocol:commandStart**

started processing of IMAP command event

**Synopsis**

```java
 glassfish:javamail:iap-protocol:commandStart (java.lang.String command)
```

**Description**

This event is sent when the JavaMail service starts to process a low-level Internet Message Access Protocol (IMAP) command.

Use this event with the `glassfish:javamail:iap-protocol:commandEnd` event to measure the amount of time that the JavaMail service spends communicating with the IMAP server.

**Parameters**

- **command**
  The name of the IMAP command, without any parameters, for example `FETCH`. 
**glassfish:javamail:pop3-protocol:multilineCommandEnd**

ended processing of POP3 command with multiline response event

**Synopsis**

glassfish:javamail:pop3-protocol:multilineCommandEnd ()

**Description**

This event is sent when the JavaMail service finishes processing a low-level Post Office Protocol - Version 3 (POP3) command. This event indicates the end of a command that expects a multiline response, for example, the RETR command.

Use this event with the glassfish:javamail:pop3-protocol:multilineCommandStart event to measure the amount of time that the JavaMail service spends communicating with the POP3 server.

**Parameters**

None.
**Synopsis**

glassfish:javamail:pop3-protocol:multilineCommandStart (java.lang.String command)

**Description**

This event is sent when the JavaMail service starts to process a low-level Post Office Protocol - Version 3 (POP3) command. This event indicates the start of a command that expects a multiline response, for example, the RETR command.

Use this event with the glassfish:javamail:pop3-protocol:multilineCommandEnd event to measure the amount of time that the JavaMail service spends communicating with the POP3 server.

**Parameters**

- **command**
  The POP3 command and its parameters.
**Synopsis**

The `glassfish:javamail:pop3-protocol:simpleCommandEnd` event is sent when the JavaMail service finishes processing a low-level Post Office Protocol - Version 3 (POP3) command. This event indicates the end of a command that accepts only single-line responses.

**Description**

This event is sent when the JavaMail service finishes processing a low-level Post Office Protocol - Version 3 (POP3) command. This event indicates the end of a command that accepts only single-line responses.

Use this event with the `glassfish:javamail:iap-protocol:commandStart` event to measure the amount of time that the JavaMail service spends communicating with the POP3 server.

**Parameters**

None.
**Synopsis**

glassfish:javamail:pop3-protocol:simpleCommandStart (java.lang.String command)

**Description**

This event is sent when the JavaMail service starts to process a low-level Post Office Protocol - Version 3 (POP3) command. This event indicates the start of a command that accepts only single-line responses.

Use this event with the *glassfish:javamail:pop3-protocol:simpleCommandEnd* event to measure the amount of time that the JavaMail service spends communicating with the POP3 server.

**Parameters**

- **command**
  The POP3 command and its parameters.
**glassfish:javamail:smtp-transport:sendMessageEnd**

finished sending message by using SMTP event

**Synopsis**

`glassfish:javamail:smtp-transport:sendMessageEnd ()`

**Description**

This event is sent when the JavaMail service finishes sending a message by using Simple Mail Transfer Protocol (SMTP).

Use this event for the following purposes:

- To count the number of messages that the JavaMail service has sent.
- To measure the amount of time that the JavaMail service spends sending messages. In this situation, use this event with the `glassfish:javamail:smtp-transport:sendMessageStart` event.

**Parameters**

None.
**Synopsis**

```
glassfish:javamail:smtp-transport:sendMessageStart (java.lang.String subject)
```
connection acquired from connector connection pool event

Synopsis

glassfish:jca:connection-pool:connectionAcquiredEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

Description

This event is sent when a connection is acquired from a connector connection pool. Use this event to count the number of logical connections that have been acquired from the pool since the last sampling.

Parameters

poolName
The name of the connector connection pool from which the connection is acquired.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:jca:connection-pool:connectionCreatedEvent**

connection created by connector connection pool event

**Synopsis**

```java
glassfish:jca:connection-pool:connectionCreatedEvent(
    java.lang.String poolName,
    java.lang.String applicationName,
    java.lang.String moduleName)
```

**Description**

This event is sent when a connection is created by the connector connection pool. Use this event to count the number of physical connections that have been created since the last reset.

**Parameters**

- **poolName**
  The name of the connector connection pool that created the connection.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:jca:connection-pool:connectionDestroyedEvent**

connection in connector connection pool destroyed event

**Synopsis**

```
glassfish:jca:connection-pool:connectionDestroyedEvent(
    java.lang.String poolName,
    java.lang.String applicationName,
    java.lang.String moduleName)
```

**Description**

This event is sent when a connection in a connector connection pool is destroyed.
Use this event to count the number of physical connections that were destroyed since the last reset.

**Parameters**

- **poolName**
  The name of the connector connection pool that contained the destroyed connection.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

glassfish:jca:connection-pool:connectionMatchedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

**Description**

This event is sent whenever a connection in the connector connection pool under test matches the current request.

Use this event to count the number of connections successfully matched.

**Parameters**

**poolName**
The name of the connector connection pool under test.

**applicationName**
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

**moduleName**
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
connection not matched event

Synopsis

glassfish:jca:connection-pool:connectionNotMatchedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

Description

This event is sent whenever a connection in the connector connection pool under test
does not match the current request.
Use this event to count of the number of connections that are rejected during
matching.

Parameters

poolName
The name of the connector connection pool under test.

applicationName
The name of the application in which the connection pool is defined. This parameter is
applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
connection returned to connector connection pool event

Synopsis

glassfish:jca:connection-pool:connectionReleasedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

Description

This event is sent when a connection is released and returned to the connector connection pool.
Use this event to count the number of logical connections released to the pool.

Parameters

poolName
The name of the connector connection pool to which the connection is returned.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
connection request removed from queue event

Synopsis

glassfish:jca:connection-pool:connectionRequestDequeuedEvent(
    java.lang.String poolName,
    java.lang.String applicationName,
    java.lang.String moduleName)

Description

This event is sent whenever a connection request is removed from the queue. Use this event to count of the number of connection requests in the queue waiting to be serviced.

Parameters

poolName
The name of the connector connection pool for which the request was removed from the queue.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
glassfish:jca:connection-pool:connectionRequestQueuedEvent

connection request queued event

Synopsis

glassfish:jca:connection-pool:connectionRequestQueuedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

Description

This event is sent whenever a connection request is added to the queue.
Use this event count of the number of connection requests in the queue waiting to be serviced.

Parameters

poolName
The name of the connector connection pool to which the connection request was submitted.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

glassfish:jca:connection-pool:connectionRequestServedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName,
long timeTakenInMillis)

**Description**

This event is sent when a request for a connection from the connector connection pool is served. The `timeTakenInMillis` parameter indicates the time required in milliseconds to service the request.

Use this event to obtain the following information:

- The wait time of the last request that was serviced by the connector connection pool
- The average wait time for successful requests for connections from the connector connection pool

**Parameters**

- **poolName**
  The name of the connector connection pool from which the connection was served.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **timeTakenInMillis**
  The time required in milliseconds to service the last request.
connection freed event

Synopsis

glassfish:jca:connection-pool:connectionsFreedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName,
int count)

Description

This event is sent whenever a connection is freed and returned to the connector connection pool.

Use this event to count the total number of free connections in the pool as of the last sampling.

Parameters

poolName
The name of the connector connection pool to which the freed connection is returned.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

count
The number of connections freed and returned to the connector connection pool.
glassfish:jca:connection-pool:connectionTimedOutEvent

connection timed-out event

Synopsis

glassfish:jca:connection-pool:connectionTimedOutEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

Description

This event is sent when a connection in the connector connection pool times out.
Use this event to get a count of the total number of connections in the pool that timed out between the start time and the last sample time.

Parameters

poolName
The name of the connector connection pool in which the connection timed out.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

glassfish:jca:connection-pool:connectionUsedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

**Description**

This event is sent whenever a connection in the connector connection pool is used. Use this event to get the total number of connections that are currently being used.

**Parameters**

- **poolName**
The name of the connector connection pool that contains the connection that is being used.

- **applicationName**
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
glassfish:jca:connection-pool:connectionValidationFailedEvent

correlation validation failed event

Synopsis

glassfish:jca:connection-pool:connectionValidationFailedEvent(
   java.lang.String poolName,
   java.lang.String applicationName,
   java.lang.String moduleName,
   int increment)

Description

This event is sent whenever validation of a connection in the connector connection pool fails.

Use this event to get the count of the number of connections in the connection pool that failed validation from the start time until the last sample time.

Parameters

poolName
The name of the connector connection pool that contains the connection that failed validation.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

increment
The number of connections that failed validation.
**Synopsis**

glassfish:jca:connection-pool:decrementConnectionUsedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

**Description**

This event is sent whenever a connection is deleted or released and returned to the connector connection pool. This event indicates that the number of connections in use has decreased by 1.

Use this event to get the total number of connections that are currently being used.

**Parameters**

- **poolName**
  The name of the connector connection pool that contains the deleted or released connection.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

```java
glassfish:jca:connection-pool:decrementNumConnFreeEvent(
    java.lang.String poolName,
    java.lang.String applicationName,
    java.lang.String moduleName)
```

**Description**

This event is sent whenever a free connection from the connector connection pool is destroyed or when a connection is retrieved from the pool. Use this event to count the total number of free connections in the pool as of the last sampling.

**Parameters**

- **poolName**
  The name of the connector connection pool that contains the destroyed or retrieved connection.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:jca:connection-pool:incrementNumConnFreeEvent**

number of free connections increased event

**Synopsis**

```java
glassfish:jca:connection-pool:incrementNumConnFreeEvent(
    java.lang.String poolName,
    java.lang.String applicationName,
    java.lang.String moduleName,
    boolean beingDestroyed,
    int steadyPoolSize)
```

**Description**

This event is sent whenever a connection is destroyed or freed and returned to the connector connection pool.

Use this event to count the total number of free connections in the pool as of the last sampling.

**Parameters**

- **poolName**
  The name of the connector connection pool that contains the destroyed or returned connection.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **beingDestroyed**
  This parameter is `true` if an error caused the connection to be destroyed and `false` otherwise.

- **steadyPoolSize**
  The steady pool size of the connector connection pool.
**Synopsis**

`glassfish:jca:connection-pool:potentialConnLeakEvent(java.lang.String poolName, java.lang.String applicationName, java.lang.String moduleName)`

**Description**

This event is sent whenever a potential connection leak is detected in the connector connection pool.

Use this event to count of the number of potential connection leaks in the connection pool.

**Parameters**

- **poolName**
  The name of the connector connection pool in which potential connection leak is detected.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:jca:work-management:workDequeued**

work dequeued event

**Synopsis**

glassfish:jca:work-management:workDequeued(
java.lang.String raName)

**Description**
This event is sent when the submitted work is taken from the wait queue.
Use this event to count the number of dequeued work instances or to calculate the current size of work queue when you know the number of queued work instances.

**Parameters**

**raName**
The name of the resource adapter that submitted the work instance.
**Synopsis**

glassfish:jca:work-management:workProcessed(
java.lang.String raName)

**Description**

This event is sent when the submitted work instance is processed successfully. Use this event to count the number of work instances that have completed execution.

**Parameters**

- **raName**
  The name of the resource adapter that submitted the work instance.
**Synopsis**

glassfish:jca:work-management:workProcessingCompleted(
java.lang.String raName)

**Description**

This event is sent when the submitted work instance is processed.

Use this event to count the number of work instances that have completed execution or have timed out.

**Parameters**

raName
The name of the resource adapter that submitted the work instance.
**Synopsis**

glassfish:jca:work-management:workProcessingStarted(
java.lang.String raName)

**Description**

This event is sent when the processing and execution of the submitted work instance starts.
Use this event to count the number of work instances being processed.

**Parameters**

- **raName**
The name of the resource adapter that submitted the work instance.
**glassfish:jca:work-management:workQueued**

work queued event

**Synopsis**

glassfish:jca:work-management:workQueued(
java.lang.String raName)

**Description**

This event is sent when a resource adapter submits a work instance to the Work Manager and the work is queued for execution.

Use this event to count the number of work instances that are queued for execution.

**Parameters**

*raName*

The name of the resource adapter that submitted the work instance.
work submitted event

Synopsis

glassfish:jca:work-management:workSubmitted(
java.lang.String raName)

Description

This event is sent when a resource adapter submits a work instance to the Work Manager for execution of the work.
Use this event to count the number of work instances submitted.

Parameters

raName
The name of the resource adapter that submitted the work instance.
**Synopsis**

glassfish:jca:work-management:workTimedOut

glassfish:jca:work-management:workTimedOut(
java.lang.String raName)

**Description**

This event is sent when the submitted work instance is timed out.
Use this event to count the number of work instances that have timed out.

**Parameters**

- **raName**
The name of the resource adapter that submitted the work instance.
**Synopsis**

glassfish:jca:work-management:workWaitedFor(
java.lang.String raName,
java.lang.Long elapsedTime)

**Description**

This event is sent when the submitted work is taken from the work queue and considered for execution.

Use this event to get the time for which a work instance has waited before it is executed.

**Parameters**

- **raName**
  The name of the resource adapter that submitted the work instance.

- **elapsedTime**
  The time, in milliseconds, for which work instance waited for execution.
**glassfish:jdbc:connection-pool:connectionAcquiredEvent**

connection acquired from JDBC connection pool event

**Synopsis**

glassfish:jdbc:connection-pool:connectionAcquiredEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

**Description**

This event is sent when a connection is acquired from a JDBC connection pool. Use this event to count the number of logical connections that have been acquired from the pool since the last sampling.

**Parameters**

- **poolName**
The name of the JDBC connection pool from which the connection is acquired.

- **applicationName**
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:jdbc:connection-pool:connectionCreatedEvent**

connection created by JDBC connection pool event

**Synopsis**

```java
glassfish:jdbc:connection-pool:connectionCreatedEvent(
    java.lang.String poolName,
    java.lang.String applicationName,
    java.lang.String moduleName)
```

**Description**

This event is sent when a connection is created by the JDBC connection pool. Use this event to count the number of physical connections that have been created since the last reset.

**Parameters**

- **poolName**
  The name of the JDBC connection pool that created the connection.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

glassfish:jdbc:connection-pool:connectionDestroyedEvent(
  java.lang.String poolName,
  java.lang.String applicationName,
  java.lang.String moduleName)

**Description**

This event is sent when a connection in a JDBC connection pool is destroyed. Use this event to count the number of physical connections that were destroyed since the last reset.

**Parameters**

- **poolName**
  The name of the JDBC connection pool that contained the destroyed connection.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
glassfish:jdbc:connection-pool:connectionMatchedEvent

connection matched event

Synopsis

glassfish:jdbc:connection-pool:connectionMatchedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

Description

This event is sent whenever a connection in the JDBC connection pool under test matches the current request.

Use this event to count the number of connections successfully matched.

Parameters

**poolName**

The name of the JDBC connection pool under test.

**applicationName**

The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

**moduleName**

The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:jdbc:connection-pool:connectionNotMatchedEvent**

connection not matched event

**Synopsis**

glassfish:jdbc:connection-pool:connectionNotMatchedEvent(  
java.lang.String poolName,  
java.lang.String applicationName,  
java.lang.String moduleName)

**Description**

This event is sent whenever a connection in the JDBC connection pool under test does not match the current request.

Use this event to count of the number of connections that are rejected during matching.

**Parameters**

- **poolName**  
The name of the JDBC connection pool under test.

- **applicationName**  
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**  
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

glassfish:jdbc:connection-pool:connectionReleasedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

**Description**

This event is sent when a connection is released and returned to the JDBC connection pool.

Use this event to count the number of logical connections released to the pool.

**Parameters**

- **poolName**
  The name of the JDBC connection pool to which the connection is returned.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
glassfish:jdbc:connection-pool:connectionRequestDequeuedEvent

collection request removed from queue event

Synopsis

glassfish:jdbc:connection-pool:connectionRequestDequeuedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

Description

This event is sent whenever a connection request is removed from the queue.
Use this event to count of the number of connection requests in the queue waiting to
be serviced.

Parameters

poolName
The name of the JDBC connection pool for which the request was removed from the
queue.

applicationName
The name of the application in which the connection pool is defined. This parameter is
applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This
parameter is applicable only for application-scoped resources.
connection request queued event

Synopsis

glassfish:jdbc:connection-pool:connectionRequestQueuedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

Description

This event is sent whenever a connection request is added to the queue. Use this event count of the number of connection requests in the queue waiting to be serviced.

Parameters

poolName
The name of the JDBC connection pool to which the connection request was submitted.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
Synopsis

glassfish:jdbc:connection-pool:connectionRequestServedEvent(
    java.lang.String poolName,
    java.lang.String applicationName,
    java.lang.String moduleName,
    long timeTakenInMillis)

Description

This event is sent when a request for a connection from the JDBC connection pool is served. The `timeTakenInMillis` parameter indicates the time required in milliseconds to service the request.

Use this event to obtain the following information:

- The wait time of the last request that was serviced by the JDBC connection pool
- The average wait time for successful requests for connections from the JDBC connection pool

Parameters

**poolName**
The name of the JDBC connection pool from which the connection was served.

**applicationName**
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

**moduleName**
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

**timeTakenInMillis**
The time required in milliseconds to service the last request.
connection freed event

Synopsis

glassfish:jdbc:connection-pool:connectionsFreedEvent(
    java.lang.String poolName,
    java.lang.String applicationName,
    java.lang.String moduleName,
    int count)

Description

This event is sent whenever a connection is freed and returned to the JDBC connection pool.
Use this event to count the total number of free connections in the pool as of the last sampling.

Parameters

poolName
The name of the JDBC connection pool to which the freed connection is returned.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

count
The number of connections freed and returned to the JDBC connection pool.
glassfish:jdbc:connection-pool:connectionTimedOutEvent

connection timed-out event

Synopsis

glassfish:jdbc:connection-pool:connectionTimedOutEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

Description

This event is sent when a connection in the JDBC connection pool times out.
Use this event to get a count of the total number of connections in the pool that timed out between the start time and the last sample time.

Parameters

poolName
The name of the JDBC connection pool in which the connection timed out.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:jdbc:connection-pool:connectionUsedEvent**

connection used event

**Synopsis**

glassfish:jdbc:connection-pool:connectionUsedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

**Description**

This event is sent whenever a connection in the JDBC connection pool is used. Use this event to get the total number of connections that are currently being used.

**Parameters**

- **poolName**
The name of the JDBC connection pool that contains the connection that is being used.

- **applicationName**
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

```java
glassfish:jdbc:connection-pool:connectionValidationFailedEvent(
    java.lang.String poolName,
    java.lang.String applicationName,
    java.lang.String moduleName,
    int increment)
```

**Description**

This event is sent whenever validation of a connection in the JDBC connection pool fails.

Use this event to get the count of the number of connections in the connection pool that failed validation from the start time until the last sample time.

**Parameters**

- **poolName**
  The name of the JDBC connection pool that contains the connection that failed validation.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **increment**
  The number of connections that failed validation.
**Synopsis**

glassfish:jdbc:connection-pool:decrementConnectionUsedEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

**Description**

This event is sent whenever a connection is deleted or released and returned to the
JDBC connection pool. This event indicates that the number of connections in use has
decreased by 1.

Use this event to get the total number of connections that are currently being used.

**Parameters**

- **poolName**
The name of the JDBC connection pool that contains the deleted or released
creation.

- **applicationName**
The name of the application in which the connection pool is defined. This parameter is
applicable only for application-scoped resources.

- **moduleName**
The name of the application module in which the connection pool is defined. This
parameter is applicable only for application-scoped resources.
**Synopsis**

glassfish:jdbc:connection-pool:decrementNumConnFreeEvent(java.lang.String poolName, java.lang.String applicationName, java.lang.String moduleName)

**Description**

This event is sent whenever a free connection from the JDBC connection pool is destroyed or when a connection is retrieved from the pool.

Use this event to count the total number of free connections in the pool as of the last sampling.

**Parameters**

- **poolName**
  The name of the JDBC connection pool that contains the destroyed or retrieved connection.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
glassfish:jdbc:connection-pool:incrementNumConnFreeEvent

number of free connections increased event

Synopsis

glassfish:jdbc:connection-pool:incrementNumConnFreeEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName,
boolean beingDestroyed,
int steadyPoolSize)

Description

This event is sent whenever a connection is destroyed or freed and returned to the
JDBC connection pool.

Use this event to count the total number of free connections in the pool as of the last
sampling.

Parameters

poolName
The name of the JDBC connection pool that contains the destroyed or returned
connection.

applicationName
The name of the application in which the connection pool is defined. This parameter is
applicable only for application-scoped resources.

moduleName
The name of the application module in which the connection pool is defined. This
parameter is applicable only for application-scoped resources.

beingDestroyed
This parameter is true if an error caused the connection to be destroyed and false
otherwise.

steadyPoolSize
The steady pool size of the JDBC connection pool.
**Synopsis**

glassfish:jdbc:connection-pool:potentialConnLeakEvent(
  java.lang.String poolName,
  java.lang.String applicationName,
  java.lang.String moduleName)

**Description**

This event is sent whenever a potential connection leak is detected in the JDBC connection pool.

Use this event to count of the number of potential connection leaks in the connection pool.

**Parameters**

- **poolName**
  The name of the JDBC connection pool in which potential connection leak is detected.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
connection acquired by application from JDBC connection pool event

Synopsis

glassfish:jdbc-pool:applications:connectionAcquiredEvent(
   java.lang.String poolName,
   java.lang.String applicationName)

Description

This event is sent when an application acquires a connection from a JDBC connection pool.

Use this event to count the number of logical connections an application has acquired from a pool since the last sampling.

Parameters

poolName
The name of the JDBC connection pool from which the connection is acquired.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

glassfish:jdbc-pool:applications:connectionReleasedEvent(
java.lang.String poolName,
java.lang.String applicationName)

**Description**

This event is sent when an application releases a connection, returning it to the JDBC
connection pool.

Use this event to count the number of logical connections an application releases to a
pool.

**Parameters**

**poolName**
The name of the JDBC connection pool to which the connection is returned.

**applicationName**
The name of the application in which the connection pool is defined. This parameter is
applicable only for application-scoped resources.
glassfish:jdbc-pool:applications:connectionReleasedEvent
glassfish:jdbc-pool:applications:connectionUsedEvent

connection used by application event

Synopsis

glassfish:jdbc-pool:applications:connectionUsedEvent(
java.lang.String poolName,
java.lang.String applicationName)

Description

This event is sent whenever an application uses a connection in the JDBC connection pool.
Use this event to get the total number of connections that are currently being used by an application.

Parameters

poolName
The name of the JDBC connection pool that contains the connection being used.

applicationName
The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

glassfish:jdbc-pool:applications:decrementConnectionUsedEvent(
  java.lang.String poolName,
  java.lang.String applicationName)

**Description**

This event is sent whenever an application destroys a connection or releases a connection back to the JDBC connection pool. This event indicates that the number of connections in use has decreased by 1.

Use this event to calculate the total number of connections that are currently being used by an application.

**Parameters**

- **poolName**
  The name of the JDBC connection pool that contains the destroyed or released connection.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

```java
glassfish:jdbcra:sqltracing:traceSqlEvent(
    java.lang.String poolName,
    java.lang.String applicationName,
    java.lang.String moduleName)
```

**Description**

This event is sent whenever an SQL statement is executed, provided that the `sql-trace-listeners` attribute of the JDBC connection pool is specified.

Use this event to trace or log SQL statement execution.

**Parameters**

- **poolName**
  The name of the JDBC connection pool.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:jdbcra:statementcache:statementCacheHitEvent**

SQL statement cache hit event

**Synopsis**

glassfish:jdbcra:statementcache:statementCacheHitEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

**Description**

This event is sent whenever an SQL statement to be executed is retrieved from the SQL statement cache, provided that statement caching is enabled; that is, when the `statement-cache-size` attribute of the JDBC connection pool has a nonzero value.

Use this event to track the number of times the statement cache is used.

**Parameters**

- **poolName**
  The name of the JDBC connection pool.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:jdbcra:statementcache:statementCacheMissEvent**

SQL statement cache miss event

**Synopsis**

glassfish:jdbcra:statementcache:statementCacheMissEvent(
java.lang.String poolName,
java.lang.String applicationName,
java.lang.String moduleName)

**Description**

This event is sent whenever an SQL statement to be executed is not found in the SQL statement cache, provided that statement caching is enabled; that is, when the statement-cache-size attribute of the JDBC connection pool has a nonzero value. Use this event to track the number of times the statement cache is not used.

**Parameters**

- **poolName**
  The name of the JDBC connection pool.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**Synopsis**

`glassfish:jdbcra:statementleak:potentialStatementLeakEvent(java.lang.String poolName, java.lang.String applicationName, java.lang.String moduleName)`

**Description**

This event is sent whenever an SQL statement acquired by an application is not returned within the amount of time specified by the `sql-trace-listeners` attribute of the JDBC connection pool. If the `sql-trace-listeners` attribute has the value zero, statement-leak checking is not performed and this event is not sent.

Use this event to track the number of SQL statement leaks in an application.

**Parameters**

- **poolName**
  The name of the JDBC connection pool.

- **applicationName**
  The name of the application in which the connection pool is defined. This parameter is applicable only for application-scoped resources.

- **moduleName**
  The name of the application module in which the connection pool is defined. This parameter is applicable only for application-scoped resources.
**glassfish:jersey:server:requestEnd**

Jersey request end event

**Synopsis**

```java
glassfish:jersey:server:requestEnd()
```

**Description**

This event is called when a request is dispatched by the Jersey framework. Use this event to keep track of the active Jersey requests on the server.

**Parameters**

None.
glassfish:jersey:server:requestStart

Jersey request start event

Synopsis

glassfish:jersey:server:requestStart(
java.lang.String requestUri)

Description

This event is called when a request is received by the Jersey framework. Use this event to keep track of the active Jersey requests on the server.

Parameters

requestUri
The URI of the incoming request.
**Synopsis**

```
glassfish:jersey:server:ruleAccept(
    java.lang.String ruleName,
    java.lang.String path,
    java.lang.String resourceClassName)
```

**Description**

This event is called when a UriRule is matched (for details see com.sun.jersey.server.impl.uri.rules).

Use this event to keep track of UriRule matching on the server.

**Parameters**

- **ruleName**
  The UriRule name.

- **path**
  The matched path.

- **resourceClassName**
  The name of the resource class associated with the matching.
**glassfish:jsf:faces-servlet:requestEnd**

JSF request end event

**Synopsis**

glassfish:jsf:faces-servlet:requestEnd()

**Description**

This event is called when a *javax.faces.webapp.FacesServlet* has finished processing a request for a JSF view or a JSF managed resource.

Use this event to keep track of the active JSF requests on the server.

**Parameters**

None.
**glassfish:jsf:faces-servlet:requestStart**

JSF request start event

**Synopsis**

glassfish:jsf:faces-servlet:requestStart(
java.lang.String requestUri)

**Description**

This event is called when a `javax.faces.webapp.FacesServlet` begins processing a request for a JSF view or a JSF managed resource.

Use this event to keep track of the active JSF requests on the server.

**Parameters**

*requestURI*

The request URI that triggered the `FacesServlet.service` method.
**glassfish:kernel:connection-queue:connectionAcceptedEvent**

connection accepted event

**Synopsis**

```java
glassfish:kernel:connection-queue:connectionAcceptedEvent(
java.lang.String listenerName,
int connectionId,
java.lang.String address)
```

**Description**

This event is called when a new network connection gets accepted by the server. Use this event to count the number of accepted, alive network connections.

**Parameters**

- `listenerName` The name of the network listener that accepted the new connection.
- `connectionId` The network connection ID.
- `address` The remote address of the connection that got accepted. The address format can differ depending on the network transport used. For TCP transport, which is used most often, the format is the same as JDK `java.net.InetSocketAddress.toString`. 
**glassfish:kernel:connection-queue:connectionClosedEvent**

connection closed event

**Synopsis**

glassfish:kernel:connection-queue:connectionClosedEvent(
java.lang.String listenerName,
int connectionId)

**Description**

This event is called when a network connection is closed.
Use this event to count the number of alive client connections.

**Parameters**

- **listenerName**
The name of the network listener that first accepted the connection.

- **connectionId**
The network connection ID.
connection connected event

Synopsis

glassfish:kernel:connection-queue:connectionConnectedEvent(
    java.lang.String listenerName,
    int connectionId,
    java.lang.String address)

Description

This event is called when a new network connection is connected to a remote server. Use this event to count the number of total or alive client connections made from the server. This event gets called only for client connections initiated from the Grizzly network framework.

Parameters

listenerName
The name of the network listener that first accepted the connection.

connectionId
The network connection ID.

address
The remote address of the connection that got connected. The address format can differ depending on the network transport used. For TCP transport, which is used most often, the format is the same as JDK java.net.InetSocketAddress.toString().
**glassfish:kernel:connection-queue:onTaskDequeuedEvent**

on task dequeued event

**Synopsis**

glassfish:kernel:connection-queue:onTaskDequeuedEvent(
java.lang.String listenerName,
int taskId)

**Description**

This event is called when a task is removed from a task queue.

Use this event to count the number of queued tasks.

**Parameters**

- **listenerName**
  The name of the network listener associated with the task queue.

- **taskId**
  The ID of the newly dequeued task.
**Synopsis**

glassfish:kernel:connection-queue:onTaskQueuedEvent(java.lang.String listenerName, int taskId)

**Description**
This event is called when a new task is added to a task queue. Use this event to count the total number of queued tasks.

**Parameters**

*listenerName*
The name of the network listener associated with the task queue.

*taskId*
The ID of the newly queued task.
**glassfish:kernel:connection-queue:onTaskQueueOverflowEvent**

on task queue overflow event

**Synopsis**

glassfish:kernel:connection-queue:onTaskQueueOverflowEvent(
java.lang.String listenerName)

**Description**

This event is called when a new task cannot be added to a task queue because the maximum number of tasks in the queue has been reached.

Use this event to count the number of tasks rejected by the server because of the task queue size limitation.

**Parameters**

*listenerName*

The name of the network listener associated with the task queue.
glassfish:kernel:connection-queue:setMaxTaskQueueSizeEvent

set maximum task queue size event

Synopsis

glassfish:kernel:connection-queue:setMaxTaskQueueSizeEvent(
java.lang.String listenerName,
int size)

Description

This event is called when the maximum size of a task queue is changed.
Use this event to get notifications of maximum queue size value changes.

Parameters

listenerName
The name of the network listener associated with the task queue.

size
The maximum number of tasks that can be queued.
**glassfish:kernel:connections-keep-alive:decrementCountConnectionsEvent**

decrement count connections event

**Synopsis**

glassfish:kernel:connections-keep-alive:decrementCountConnectionsEvent(
java.lang.String listenerName)

**Description**

This event is called when an HTTP keep-alive connection is closed.

Use this event to count the total number of active HTTP connections processed in keep-alive mode.

**Parameters**

- **listenerName**
  The name of the network listener associated with this keep-alive setting.
**Synopsis**

glassfish:kernel:connections-keep-alive:incrementCountConnectionsEvent(
java.lang.String listenerName)

**Description**

This event is called when an accepted HTTP connection is processed in keep-alive mode.

Use this event to count the total number of active HTTP connections processed in keep-alive mode.

**Parameters**

- **listenerName**
  The name of the network listener associated with this keep-alive setting.
**Synopsis**

glassfish:kernel:connections-keep-alive:incrementCountFlushesEvent(
java.lang.String listenerName)

**Description**

This event is called when an HTTP keep-alive connection gets closed normally by the client or server.

Use this event to count the number of times HTTP keep-alive connections are closed normally.

**Parameters**

*listenerName*

The name of the network listener associated with this keep-alive setting.
increment count hits event

Synopsis

glassfish:kernel:connections-keep-alive:incrementCountHitsEvent(
java.lang.String listenerName)

Description

This event is called when a new request is sent to an HTTP keep-alive connection. Use this event to count the total number of requests processed by HTTP keep-alive connections.

Parameters

listenerName
The name of the network listener associated with this keep-alive setting.
increment count refusals event

Synopsis

glassfish:kernel:connections-keep-alive:incrementCountRefusalsEvent(
java.lang.String listenerName)

Description

This event is called when an HTTP keep-alive connection is closed because the maximum number of HTTP requests that can be processed over a single connection has been reached.

Use this event to count the number of times an HTTP connection is closed because the maximum number of HTTP requests that can be processed over a single connection has been reached.

Parameters

listenerName
The name of the network listener associated with this keep-alive setting.
glassfish:kernel:connections-keep-alive:incrementCountRefusalsEvent
glassfish:kernel:connections-keep-alive:incrementCountTimeoutsEvent

increment count timeouts event

Synopsis

glassfish:kernel:connections-keep-alive:incrementCountTimeoutsEvent(
java.lang.String listenerName)

Description

This event is called when an HTTP keep-alive connection is closed because its idle
timeout is exceeded.

Use this event to count the number of times HTTP connections are closed because their
idle timeouts are exceeded.

Parameters

listenerName
The name of the network listener associated with this keep-alive setting.
**Synopsis**

glassfish:kernel:connections-keep-alive:setMaxCountRequestsEvent(
java.lang.String listenerName,
int maxRequests)

**Description**

This event is called when the maximum allowed number of HTTP requests that can be processed on a single connection is changed.

Use this event to get notification of changes to the maximum allowed number of HTTP requests that can be processed on a single connection.

**Parameters**

*listenerName*

The name of the network listener associated with this keep-alive setting.

*maxRequests*

The maximum allowed number of HTTP requests that can be processed on a single connection.
**Synopsis**

glassfish:kernel:connections-keep-alive:setTimeoutInSecondsEvent(    java.lang.String listenerName,    int timeoutInSeconds)

**Description**

This event is called when the time during which a keep-alive connection can stay idle is changed.
Use this event to get notification of changes to the time during which a keep-alive connection can stay idle.

**Parameters**

listenerName
The name of the network listener associated with this keep-alive setting.

timeoutInSeconds
The time in seconds during which a keep-alive connection can stay idle.
**Synopsis**

glassfish:kernel:file-cache:addHeapSizeEvent(
java.lang.String fileCacheName,
long size)

**Description**

This event is called when the file cache allocates additional heap memory for its resources.

Use this event to count heap memory size used by the file cache.

**Parameters**

- **fileCacheName**
  The file cache name.

- **size**
  The size of the newly allocated heap memory block.
add mapped memory size event

Synopsis

glassfish:kernel:file-cache:addMappedMemorySizeEvent(
java.lang.String fileCacheName,
long size)

Description

This event is called when the file cache allocates additional mapped memory for its resources.
Use this event to count mapped memory size used by the file cache.

Parameters

fileCacheName
The file cache name.

size
The size of the newly allocated mapped memory block.
**Synopsis**

glassfish:kernel:file-cache:countContentHitEvent(
java.lang.String fileCacheName)

**Description**

This event is called when a content resource looked up in the file cache is found.
Use this event to count the number of times a requested content resource is found in the file cache.

**Parameters**

=fileCacheName
The file cache name.
glassfish:kernel:file-cache:countContentMissEvent

count content miss event

Synopsis
glassfish:kernel:file-cache:countContentMissEvent(
java.lang.String fileCacheName)

Description
This event is called when resource content looked up in the file cache is not found.
Use this event to count the number of times requested resource content is not found in
the file cache.

Parameters

tFileCacheName
The file cache name.
**glassfish:kernel:file-cache:countHitEvent**

Count hit event

**Synopsis**

glassfish:kernel:file-cache:countHitEvent(
java.lang.String fileCacheName)

**Description**

This event is called when a resource looked up in the file cache is found. Use this event to count the number of times a requested resource is found in the file cache.

**Parameters**

- **fileCacheName**
  The file cache name.
**Synopsis**

glassfish:kernel:file-cache:countInfoHitEvent(
java.lang.String fileCacheName)

**Description**

This event is called when an information resource looked up in the file cache is found.
Use this event to count the number of times a requested information resource is found in the file cache.

**Parameters**

`fileCacheName`
The file cache name.
glassfish:kernel:file-cache:countInfoMissEvent

count information miss event

Synopsis
glassfish:kernel:file-cache:countInfoMissEvent(
java.lang.String fileCacheName)

Description
This event is called when resource information looked up in the file cache is not found. Use this event to count the number of times requested resource information is not found in the file cache.

Parameters

fileCacheName
The file cache name.
**glassfish:kernel:file-cache:countMissEvent**

count miss event

**Synopsis**

```java
glassfish:kernel:file-cache:countMissEvent(
java.lang.String fileCacheName)
```

**Description**

This event is called when a resource looked up in the file cache is not found.

Use this event to count the number of times a requested resource is not found in the file cache.

**Parameters**

- **fileCacheName**
  The file cache name.
**glassfish:kernel:file-cache:decOpenCacheEntriesEvent**

decrement open cache entries event

**Synopsis**

glassfish:kernel:file-cache:decOpenCacheEntriesEvent(
java.lang.String fileCacheName)

**Description**

This event is called when a resource is removed from the file cache.
Use this event to count the number of resources stored in the file cache.

**Parameters**

fileCacheName
The file cache name.
increment open cache entries event

**Synopsis**

glassfish:kernel:file-cache:incOpenCacheEntriesEvent(
java.lang.String fileCacheName)

**Description**

This event is called when a new resource is added to the file cache.
Use this event to count the number of resources stored in the file cache.

**Parameters**

fileCacheName
The file cache name.
glassfish:kernel:file-cache:subHeapSizeEvent

subtract heap size event

Synopsis
glassfish:kernel:file-cache:subHeapSizeEvent(
java.lang.String fileCacheName,
long size)

Description
This event is called when the file cache releases heap memory.
Use this event to count heap memory size used by the file cache.

Parameters

fileCacheName
The file cache name.

size
The size of the released heap memory block.
**Synopsis**

glassfish:kernel:file-cache:subMappedMemorySizeEvent(
java.lang.String fileCacheName,
long size)

**Description**

This event is called when the file cache releases mapped memory. Use this event to count mapped memory size used by the file cache.

**Parameters**

- **fileCacheName**
  The file cache name.

- **size**
  The size of the released mapped memory block.
**Synopsis**

glassfish:kernel:connections-thread-pool:maxNumberOfThreadsReachedEvent(
java.lang.String monitoringId,
java.lang.String threadPoolName,
int maxNumberOfThreads)

**Description**

This event is called when a thread pool reaches its maximum number of threads.
Use this event to count the number times a thread pool reaches its maximum number of threads.

**Parameters**

- **monitoringId**
  The thread pool monitoring identifier.

- **threadPoolName**
  The thread pool name in the server configuration.

- **maxNumberOfThreads**
  The maximum number of threads in a thread pool.
glassfish:kernel:thread-pool:maxNumberOfThreadsReachedEvent
**glassfish:kernel:thread-pool:setCoreThreadsEvent**

set core threads event

**Synopsis**

glassfish:kernel:connections-thread-pool:setCoreThreadsEvent(
java.lang.String monitoringId,
java.lang.String threadPoolName,
int coreNumberOfThreads)

**Description**

This event is called when the number of core threads in a thread pool is changed. Use this event to get notification of changes to the number of core threads in a thread pool.

**Parameters**

- **monitoringId**
  The thread pool monitoring identifier.

- **threadPoolName**
  The thread pool name in the server configuration.

- **coreNumberOfThreads**
  The number of core threads in a thread pool.
glassfish:kernel:thread-pool:setCoreThreadsEvent
**glassfish:kernel:thread-pool:setMaxThreadsEvent**

set maximum threads event

**Synopsis**

glassfish:kernel:connections-thread-pool:setMaxThreadsEvent(
java.lang.String monitoringId,
java.lang.String threadPoolName,
int maxNumberOfThreads)

**Description**

This event is called when the maximum number of threads in a thread pool is changed. Use this event to get notification of changes to the maximum number of threads in a thread pool.

**Parameters**

- **monitoringId**
  The thread pool monitoring identifier.

- **threadPoolName**
  The thread pool name in the server configuration.

- **maxNumberOfThreads**
  The maximum number of threads in a thread pool.
**Synopsis**

glassfish:kernel:connections-thread-pool:threadAllocatedEvent(
java.lang.String monitoringId,
java.lang.String threadPoolName,
java.lang.String threadId)

**Description**

This event is called when a new thread is created by a thread pool. Use this event to count the number of threads in a thread pool.

**Parameters**

*monitoringId*

The thread pool monitoring identifier.

*threadPoolName*

The thread pool name in the server configuration.

*threadId*

The ID of the new thread.


glassfish:kernel:thread-pool:threadDispatchedFromPoolEvent

thread dispatched from pool event

Synopsis

glassfish:kernel:connections-thread-pool:threadDispatchedFromPoolEvent(
java.lang.String monitoringId,
java.lang.String threadPoolName,
java.lang.String threadId)

Description

This event is called when a thread is taken from a pool to execute a task.
Use this event to count the number of tasks executed by a specific thread or an entire
thread pool, or to count the number of busy threads in a thread pool.

Parameters

monitoringId
The thread pool monitoring identifier.

threadPoolName
The thread pool name in the server configuration.

threadId
The ID of a dispatched thread.
**Synopsis**

glassfish:kernel:connections-thread-pool:threadReleasedEvent(
java.lang.String monitoringId,
java.lang.String threadPoolName,
java.lang.String threadId)

**Description**

This event is called when a thread is released by a thread pool.
Use this event to count the number of threads in a thread pool.

**Parameters**

- **monitoringId**
  The thread pool monitoring identifier.

- **threadPoolName**
  The thread pool name in the server configuration.

- **threadId**
  The ID of the released thread.
thread returned to pool event

Synopsis

glassfish:kernel:connections-thread-pool:threadReturnedToPoolEvent(
java.lang.String monitoringId,
java.lang.String threadPoolName,
java.lang.String threadId)

Description

This event is called when a thread has completed its task execution and returns to the pool.

Use this event to count the number of tasks executed by a specific thread or an entire thread pool, or to count the number of busy threads in a thread pool.

Parameters

monitoringId
The thread pool monitoring identifier.

threadPoolName
The thread pool name in the server configuration.

threadId
The ID of a returned thread.
**Synopsis**

glassfish:orb:inboundconnection:inboundConnectionClosed (java.lang.String connection)

**Description**

This event is sent whenever a server object request broker (ORB) closes a Transmission Control Protocol (TCP) connection in the inbound connection cache. A server ORB accepts requests from a client ORB.

**Parameters**

- **connection**
  The ORB object that represents the TCP connection.
connection accepted by server ORB event

Synopsis

Description
This event is sent whenever a server object request broker (ORB) accepts a Transmission Control Protocol (TCP) connection that a client ORB has opened. The client ORB opens a connection when preparing to send a request to the server ORB.

After the connection is opened, the connection is cached and remains open until either of the following happens:

- A communications failure occurs.
- The number of connections in the inbound connection cache exceeds the ORB high water mark and the connection is the least recently used cached connection in the inbound cache. In this situation, the connection is closed.

Parameters

acceptor
The server ORB object. This object represents the end point to which the connection refers and contains the TCP port on which the server ORB listens for requests.

connection
The client ORB object. This object represents the TCP connection that the server ORB accepted.
glassfish:orb:outboundconnection:outboundConnectionClosed

connection closed by client ORB event

Synopsis

glassfish:orb:outboundconnection:outboundConnectionClosed (  
java.lang.String contactInfo,  
java.lang.String connection)

Description

This event is sent whenever a client object request broker (ORB) closes a Transmission Control Protocol (TCP) connection to a server ORB object. When a client ORB closes a connection, the connection is removed from the cache.

A client ORB originates requests to a server ORB. A server ORB accepts requests from a client ORB.

Parameters

contactInfo
The server ORB object. This object represents the remote end point to which the connection refers. This object contains the host and port of the remote end point.

connection
The client ORB object. This object represents the TCP connection that the client ORB closed.
**glassfish:orb:outboundconnection:outboundConnectionOpened**

connection opened by client ORB event

**Synopsis**

```java
glassfish:orb:outboundconnection:outboundConnectionOpened (java.lang.String contactInfo,
java.lang.String connection)
```

**Description**

This event is sent whenever a client object request broker (ORB) opens a new Transmission Control Protocol (TCP) connection to a server ORB. The client ORB opens a connection when preparing to send a request to the server ORB.

After the connection is opened, the connection is cached and reused until either of the following happens:

- A communications failure occurs.
- The number of connections in the outbound connection cache exceeds the ORB high water mark, and the connection is the least recently used cached connection in the outbound cache. In this situation, the connection is closed.

**Parameters**

- **contactInfo**
  The server ORB object. This object represents the remote end point to which the connection refers. This object contains the host and port of the remote end point.

- **connection**
  The client ORB object. This object represents the TCP connection that the client ORB opened.
**Synopsis**

glassfish:security:ejb:policyDestructionEvent(
java.lang.String contextId)

**Description**

This event is sent when security policy configuration destruction starts undeploying a standalone EJB module or an EJB module in an enterprise archive file.

Use this event to count the number of EJB security policy configuration objects that are currently present. This event decrements the EJB policy configuration count.

**Parameters**

contextId
The web context id of the specific EJB application.
**Synopsis**

glassfish:security:ejbpolicy:policyCreationEvent(
java.lang.String contextId)

**Description**

This event is sent when security policy configuration creation starts deploying a standalone EJB module or an EJB module in an enterprise archive file.

Use this event to count the number of EJB security policy configuration objects that are currently present. This event increments the EJB policy configuration count.

**Parameters**

- **contextId**
  The EJB context id of the specific EJB application.
**glassfish:security:ejb:securityManagerCreationEvent**

EJB security manager creation event

**Synopsis**

glassfish:security:ejb:securityManagerCreationEvent(
java.lang.String appName)

**Description**

This event is sent when security manager creation starts deploying a standalone EJB module or an EJB module in an enterprise archive file.

Use this event to count the number of EJB security manager objects that are currently present. This event increments the EJB security manager count.

**Parameters**

- **appName**
  The name of the application to which the EJB module belongs.
glassfish:security:ejb:securityManagerDestructionEvent

EJB security manager destruction event

Synopsis
glassfish:security:ejb:securityManagerDestructionEvent(
java.lang.String appName)

Description
This event is sent when security manager destruction starts undeploying a standalone
EJB module or an EJB module in an enterprise archive file.
Use this event to count the number of EJB security manager objects that are currently
present. This event decrements the EJB security manager count.

Parameters

appName
The name of the application to which the EJB module belongs.
glassfish:security:realm:realmAddedEvent

security realm added event

Synopsis

glassfish:security:realm:realmAddedEvent(
    java.lang.String realmName)

Description

This event is sent when a new security realm is created by the `asadmin` CLI interface or by the Admin GUI interface.

Use this event to count the number security realms that are currently present. This event increments the number of realms.

Parameters

-realmName
The name of the security realm that is created.
**Synopsis**

glassfish:security:realm:realmRemovedEvent(
java.lang.String realmName)

**Description**

This event is sent when a specific security realm is removed by the asadmin CLI interface or by the Admin GUI interface. Use this event to count the number security realms that are currently present. This event decrements the number of realms.

**Parameters**

*realmName*

The name of the security realm that is removed.
**glassfish:security:web:policyConfigurationCreationEvent**

policy configuration creation event

**Synopsis**

glassfish:security:web:policyConfigurationCreationEvent(
    java.lang.String contextId)

**Description**

This event is sent when security policy configuration creation starts deploying a standalone web module or a web module in an enterprise archive file. Use this event to count the number of web security policy configuration objects that are currently present. This event increments the web policy configuration count.

**Parameters**

`contextId`

The web context id of the web application.
glassfish:security:web:policyConfigurationDestructionEvent

policy configuration destruction event

Synopsis

glassfish:security:web:policyConfigurationDestructionEvent(
   java.lang.String contextId)

Description

This event is sent when security policy configuration destruction starts undeploying a standalone web module or a web module in an enterprise archive file.

Use this event to count the number of web security policy configuration objects that are currently present. This event decrements the web policy configuration count.

Parameters

contextId
The web context id of the web application.
**Synopsis**

`glassfish:security:web:securityManagerCreationEvent(java.lang.String appName)`

**Description**

This event is sent when the security manager creation starts on deploying a standalone web module or a web module in an enterprise archive file.

Use this event to count the number of web security managers that are currently present. This event increments the web security manager count.

**Parameters**

- **appName**
  The name of the application to which the web module belongs.
glassfish:security:web:securityManagerDestructionEvent

security manager destruction event

Synopsis

glassfish:security:web:securityManagerDestructionEvent(
java.lang.String appName)

Description

This event is sent when the security manager destruction starts on deploying a standalone web module or a web module in an enterprise archive file.

Use this event to count the number of web security managers that are currently present. This event decrements the web security manager count.

Parameters

appName
The name of the application to which the web module belongs.
**glassfish:transaction:transaction-service:activated**

transaction created event

**Synopsis**

```
glassfish:transaction:transaction-service:activated ()
```

**Description**

This event is sent when a new transaction instance is created by the transaction manager. Transaction can be destroyed and another transaction created instead if a non-XA transaction had been upgraded to an XA transaction.

Use this event to count the number of active transactions in the server.

**Parameters**

None.
**glassfish:transaction:transaction-service:committed**

transaction committed event

**Synopsis**

glassfish:transaction:transaction-service:committed ()

**Description**

This event is sent when an active transaction had been successfully committed. Use this event to count the number of committed transactions in the server.

**Parameters**

None.
**glassfish:transaction:transaction-service:deactivated**

transaction destroyed event

**Synopsis**

glassfish:transaction:transaction-service:deactivated ()

**Description**

This event is sent when a new transaction instance is destroyed by the transaction manager. Transaction can be destroyed and another transaction created instead if a non-XA transaction had been upgraded to an XA transaction.

Use this event to count the number of destroyed transactions in the server.

**Parameters**

None.
**Synopsis**

glassfish:transaction:transaction-service:freeze (boolean isFrozen)

**Description**

This event is sent as a result of the state change in transaction manager to or from frozen (locked).

Use this event to track the frozen (locked) state of the transaction manager.

**Parameters**

*isFrozen*

true if the transaction manager became frozen (locked), false otherwise.
**glassfish:transaction:transaction-service:rolledback**

transaction rolled back event

**Synopsis**

glassfish:transaction:transaction-service:rolledback ()

**Description**

This event is sent when an active transaction had been rolled back.

Use this event to count the number of rolled back transactions in the server.

**Parameters**

None.
glassfish:web:http-service:requestEndEvent

event

Synopsis
glassfish:web:http-service:requestEndEvent(
    java.lang.String appName,
    java.lang.String hostName,
    java.lang.String serverName,
    int serverPort,
    java.lang.String contextPath,
    java.lang.String servletPath,
    int statusCode)

Description
This event is sent whenever an HTTP request has been processed by the web container and the corresponding HTTP response is about to be returned to the client.
Use this event to determine the response time per application.

Parameters

appName
The name of the web application that has finished processing the HTTP request and produced an HTTP response.

hostName
The name of the virtual server on which the application has been deployed.

serverName
The server name of the HTTP request.

serverPort
The server port of the HTTP request.

contextPath
The context path portion of the URI of the HTTP request.

servletPath
The servlet path portion of the URI of the HTTP request.

statusCode
The status code of the HTTP response.
**Synopsis**

glassfish:web:http-service:requestStartEvent(
    java.lang.String appName,
    java.lang.String hostName,
    java.lang.String serverName,
    int serverPort,
    java.lang.String contextPath,
    java.lang.String servletPath)

**Description**

This event is sent whenever an HTTP request has been received by the web container and is about to be dispatched to its target application.

Use this event to determine the number of hits per application.

**Parameters**

- **appName**
  The name of the web application to which the HTTP request was mapped and that is about to process it.

- **hostName**
  The name of the virtual server on which the application has been deployed.

- **serverName**
  The server name of the HTTP request.

- **serverPort**
  The server port of the HTTP request.

- **contextPath**
  The context path portion of the URI of the HTTP request.

- **servletPath**
  The servlet path portion of the URI of the HTTP request.
**Synopsis**

glassfish:web:jsp:jspDestroyedEvent(
    java.lang.String jspUri,
    java.lang.String appName,
    java.lang.String hostName)

**Description**

This event is sent whenever a JSP is being destroyed. That is, when the servlet corresponding to the JSP is called at its `destroy` method either because the JSP is being reloaded or because the application to which the JSP belongs is being stopped (for example, as part of its undeployment).

**Parameters**

- **jspUri**
  The path (relative to the root of the application) to the JSP that was destroyed.

- **appName**
  The name of the web application to which the JSP belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
**glassfish:web:jsp:jspErrorEvent**

**Synopsis**

glassfish:web:jsp:jspErrorEvent(
    java.lang.String jspUri,
    java.lang.String appName,
    java.lang.String hostName)

**Description**

This event is sent whenever access to a JSP has resulted in an error.

**Parameters**

- **jspUri**
  The path (relative to the root of the application) to the JSP that produced an error.

- **appName**
  The name of the web application to which the JSP belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
**Synopsis**

glassfish:web:jsp:jspLoadedEvent(
   java.lang.String jspUri,
   java.lang.String appName,
   java.lang.String hostName)

**Description**

This event is sent whenever a JSP has been accessed for the first time and its corresponding servlet has been loaded and initialized.

**Parameters**

- **jspUri**
  The path (relative to the root of the application) to the JSP that was loaded.

- **appName**
  The name of the web application to which the JSP belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
glassfish:web:jsp:jspReloadedEvent

event

Synopsis

glassfish:web:jsp:jspReloadedEvent(
    java.lang.String jspUri,
    java.lang.String appName,
    java.lang.String hostName)

Description

This event is sent whenever a JSP whose source code has changed since it was first deployed is accessed again and recompiled, and its corresponding servlet is reloaded and reinitialized.

Parameters

jspUri
The path (relative to the root of the application) to the JSP that was reloaded.

appName
The name of the web application to which the JSP belongs.

hostName
The name of the virtual server on which the application has been deployed.
**Synopsis**


**Description**

This event indicates the end of a web service request for an application that conforms to Java Specification Request (JSR) 109: Implementing Enterprise Web Services. Use this event to calculate the processing time for a web service request.

**Parameters**

**endpointAddress**
The full path of the web service request without `host:port`. 
**Synopsis**


**Description**

This event indicates the start of a web service request for an application that conforms to Java Specification Request (JSR) 109: Implementing Enterprise Web Services. Use this event to count the number of requests for the end point of a web service.

**Parameters**

- **endpointAddress**
The full path of the web service request without `host:port`. 
glassfish:webservices:servlet-ri:endedEvent

JAX-WS RI web service request ended event

Synopsis

glassfish:webservices:servlet-ri:endedEvent(
java.lang.String endpointAddress)

Description

This event indicates the end of a web service request for an application that uses the reference implementation (RI) of the Java API for XML Web Services (JAX-WS). A JAX-WS RI application is deployed by using the sun-jaxws.xml deployment descriptor.

Use this event to calculate the processing time for a web service request.

Parameters

endpointAddress
The full path of the web service request without host:port.
**glassfish:webservices:servlet-ri:startedEvent**

JAX-WS RI web service request started event

**Synopsis**

glassfish:webservices:servlet-ri:startedEvent(
java.lang.String endpointAddress)

**Description**

This event indicates the start of a web service request for an application that uses the reference implementation (RI) of the Java API for XML Web Services (JAX-WS). A JAX-WS RI application is deployed by using the `sun-jaxws.xml` deployment descriptor.

Use this event to count the number of requests for the end point of a web service.

**Parameters**

*endpointAddress*

The full path of the web service request without `host:port`. 
**Synopsis**

glassfish:web:servlet:afterServiceEvent(
    java.lang.String servletName,
    int responseStatus,
    java.lang.String appName,
    java.lang.String hostName)

**Description**

This event is sent whenever a servlet has processed a request.

**Parameters**

- **servletName**
  
The name of the servlet whose service method has finished execution.

- **appName**
  
The name of the web application to which the servlet belongs.

- **hostName**
  
The name of the virtual server on which the application has been deployed.
**glassfish:web:servlet:beforeServiceEvent**

event

**Synopsis**

glassfish:web:servlet:beforeServiceEvent(
    java.lang.String servletName,
    java.lang.String appName,
    java.lang.String hostName)

**Description**

This event is sent whenever a servlet is about to process a request.

**Parameters**

- **servletName**
  The name of the servlet whose service method is being entered.

- **appName**
  The name of the web application to which the servlet belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
**Synopsis**

glassfish:web:servlet:servletDestroyedEvent{
    java.lang.String servletName,
    java.lang.String appName,
    java.lang.String hostName
}

**Description**

This event is sent whenever a servlet has been removed from service, that is, after the servlet has been invoked at its `destroy` method.

**Parameters**

- **servletName**
  The name of the servlet that has been destroyed.

- **appName**
  The name of the web application to which the servlet belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
**Synopsis**

glassfish:web:servlet:servletInitializedEvent(
    java.lang.String servletName,
    java.lang.String appName,
    java.lang.String hostName)

**Description**

This event is sent whenever a servlet has been initialized, that is, after the servlet has been invoked at its `init` method.

**Parameters**

- **servletName**
  The name of the servlet that has been initialized.

- **appName**
  The name of the web application to which the servlet belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
glassfish:web:session:sessionActivatedEndEvent

event

Synopsis
glassfish:web:session:sessionActivatedEndEvent(
   java.lang.String sessionId,
   java.lang.String appName,
   java.lang.String hostName)

Description
This event is sent whenever an HTTP session has been reactivated (for example, after it has been restored from file).

Parameters

sessionId
The id of the HTTP session that has been activated.

appName
The name of the web application to which the HTTP session belongs.

hostName
The name of the virtual server on which the application has been deployed.
event

Synopsis
glassfish:web:session:sessionActivatedStartEvent(
    java.lang.String sessionId,
    java.lang.String appName,
    java.lang.String hostName)

Description
This event is sent whenever an HTTP session is about to be reactivated (for example, after it has been restored from file).

Parameters

sessionId
The ID of the HTTP session that is about to be activated.

appName
The name of the web application to which the HTTP session belongs.

hostName
The name of the virtual server on which the application has been deployed.
**glassfish:web:session:sessionCreatedEvent**

Event

**Synopsis**

glassfish:web:session:sessionCreatedEvent(
    java.lang.String sessionId,
    java.lang.String appName,
    java.lang.String hostName)

**Description**

This event is sent whenever an HTTP session has been created.

**Parameters**

- **sessionId**
  The ID of the HTTP session that has been created.

- **appName**
  The name of the web application to which the HTTP session belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
**Synopsis**

glassfish:web:session:sessionDestroyedEvent(
    java.lang.String sessionId,
    java.lang.String appName,
    java.lang.String hostName)

**Description**

This event is sent whenever an HTTP session has been destroyed due to expiration or explicit invalidation.

**Parameters**

- **sessionId**
  The ID of the HTTP session that has been destroyed.

- **appName**
  The name of the web application to which the HTTP session belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
glassfish:web:session:sessionExpiredEvent

event

Synopsis
glassfish:web:session:sessionExpiredEvent(
   java.lang.String sessionId,
   java.lang.String appName,
   java.lang.String hostName)

Description
This event is sent whenever an HTTP session has expired because it has been inactive
for too long.
Use this event to determine if the length of time during which an HTTP session must
be accessed to prevent it from expiring needs to be increased.

Parameters

   sessionId
   The ID of the HTTP session that has expired.

   appName
   The name of the web application to which the HTTP session belongs.

   hostName
   The name of the virtual server on which the application has been deployed.
**glassfish:web:session:sessionPassivatedEndEvent**

*event*

**Synopsis**

```java
glassfish:web:session:sessionPassivatedEndEvent(
    java.lang.String sessionId,
    java.lang.String appName,
    java.lang.String hostName)
```

**Description**

This event is sent whenever an HTTP session has been passivated (for example, in preparation for being persisted).

**Parameters**

- **sessionId**
  The ID of the HTTP session that has been passivated.

- **appName**
  The name of the web application to which the HTTP session belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
**Synopsis**

glassfish:web:session:sessionPassivatedStartEvent(
    java.lang.String sessionId,
    java.lang.String appName,
    java.lang.String hostName)

**Description**

This event is sent whenever an HTTP session is about to be passivated (for example, in preparation for being persisted).

**Parameters**

- **sessionId**
  The ID of the HTTP session that is about to be passivated

- **appName**
  The name of the web application to which the HTTP session belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
**Synopsis**

glassfish:web:session:sessionPersistedEndEvent{
java.lang.String sessionId,
java.lang.String appName,
java.lang.String hostName)

**Description**

This event is sent whenever an HTTP session has been persisted (for example, to file).

**Parameters**

```java
sessionId
The ID of the HTTP session that has been persisted.
```

```java
appName
The name of the web application to which the HTTP session belongs.
```

```java
hostName
The name of the virtual server on which the application has been deployed.
```
**Synopsis**

glassfish:web:session:sessionPersistedStartEvent(
    java.lang.String sessionId,
    java.lang.String appName,
    java.lang.String hostName)

**Description**

This event is sent whenever an HTTP session is about to be persisted (for example, to file).

**Parameters**

- **sessionId**
  The ID of the HTTP session that is about to be persisted.

- **appName**
  The name of the web application to which the HTTP session belongs.

- **hostName**
  The name of the virtual server on which the application has been deployed.
**Synopsis**

glassfish:web:session:sessionRejectedEvent(
    int maxSessions,
    java.lang.String appName,
    java.lang.String hostName)

**Description**

This event is sent whenever an HTTP session could not be created because the maximum number of concurrent HTTP sessions for the application was reached.

Use this event to determine if the maximum number of concurrent sessions for the application needs to be increased (by default, there is no limit).

**Parameters**

- **maxSessions**
  The maximum number of active HTTP sessions allowed.

- **appName**
  The name of the web application in which an HTTP session was not created.

- **hostName**
  The name of the virtual server on which the application has been deployed.
**Synopsis**

glassfish:web:web-module:webModuleStartedEvent{
java.lang.String appName,
java.lang.String hostName)

**Description**

This event is sent whenever an application has been started (for example, as part of its deployment).

**Parameters**

- **appName**
  The name of the web application that has been started.

- **hostName**
  The name of the virtual server on which the application has been deployed.
**Synopsis**

glassfish:web:web-module:webModuleStoppedEvent

```java
    glassfish:web:web-module:webModuleStoppedEvent(
        java.lang.String appName,
        java.lang.String hostName)
```

**Description**

This event is sent whenever an application is about to be stopped (for example, as part of its undeployment).

**Parameters**

- **appName**
  The name of the web application that has been stopped.

- **hostName**
  The name of the virtual server on which the application has been deployed.