Sun GlassFish Enterprise Server v3 Reference Manual

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–7701–10
December 2009

091207@23031

Contents

Preface

Both novice users and those familiar with Sun GlassFish Enterprise 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 Enterprise Server utility commands.
- Section 5ASC describes concepts that are related to Enterprise Server administration.

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.

REFERENCE

Sun GlassFish v3 Preview Enterprise Server Section 1: asadmin Utility Subcommands

Name add-resources – creates the resources specified in an XML file

Synopsis `add-resources [--help] [--target` *target*`]` *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 *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://www.sun.com/software/appserver/dtds/sun-resources_1_4.dtd` in the `resources.xml` file.

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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

xml_file_name
The path to the XML file that contains the resource(s) to be created. If you specify an absolute path, the XML file can be anywhere. If you specify only the file, then the XML file must reside in the *install-dir*`/domains/domain1/config` directory. If you specify a relative path, then the XML file must be in the relative directory.

An example XML file follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC
   "-//Sun Microsystems Inc.//DTD Application Server 9.1 Domain//EN"
   "*http://www.sun.com/software/appserver/dtds/sun-resources_1_2.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)`

Name change-admin-password – changes the administrator password

Synopsis
```
change-admin-password
[--help]
```

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 in anonymous, and this user is not passworded, this command will fail. For security purposes, it is recommended that you create a passworded user with administrator privileges using the create-file-user command or the GlassFish Administration Console, which you can access by opening http://localhost:4848 in a web browser. When this is complete, remove the anonymous user to restrict unauthorized access to GlassFish server settings.

If more than one administrator is configured for GlassFish, you must start the `asadmin` command with the `---user` option, as shown in the example below, to change the password for that user.

This command is supported in remote mode only.

Options

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

Examples **EXAMPLE 1** Changing the Administrator Password For a Single User in Multi-Mode

```
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 `delete-password-alias(1)`, `list-password-aliases(1)`, `update-password-alias(1)`

`asadmin(1M)`

Name change-master-password – changes the master password

Synopsis

```
change-master-password
[--help]
 [ --domaindir  domain_path | --agentdir node-agent_path]
 [--savemasterpassword={false|true}] [domain_name | node_agent_name]
```

Description This local 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, with the node agent stopped.

Options

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

`--domaindir`
This option specifies the 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`.

Do not specify the `--domaindir` option and the `--agentdir` option in the same command. Use one option or the other.

`--agentdir`
Like a domain administration server (DAS), each node agent resides in a top level directory named *agentdir*/*nodeagent-name*. If the `--agentdir` option is not specified, the directory `$AS_DEF_DOMAINS_PATH/../nodeagents` is used.

Do not specify the `--domaindir` option and the `--agentdir` option in the same command. Use one option or the other.

`--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*
This is the domain name whose password is to be changed. If there is only a single domain, this is optional.

node_agent_name
This is the name of the node agent whose password is to be changed.

Examples

EXAMPLE 1 Changing the Master Password

This example assumes that you have used the `asadmin login` command before using the `change-master-password` command.

```
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	command executed successfully
1	error in executing the command

See Also

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

`asadmin(1M)`

Name configure-jruby-container – configures the Enterprise Server JRuby container

Synopsis

```
configure-jruby-container [--help]
[--monitoring={false|true}]
[--jruby-home jruby-home]
[--jruby-runtime jruby-runtime]
[--jruby-runtime-min jruby-runtime-min]
[--jruby-runtime-max jruby-runtime-max]
[--show={true|false}]
```

Description The `configure-jruby-container` subcommand configures the JRuby container of Sun GlassFish™ Enterprise Server. This subcommand also shows the current settings of the Enterprise Server JRuby container.

The Enterprise Server JRuby container enables JRuby applications to be deployed in Enterprise Server.

JRuby is an implementation of the Ruby programming language in the Java™ language. JRuby consists of the JRuby interpreter, the Ruby library, and Ruby gems. JRuby is available from Update Tool or from the JRuby community site (`http://jruby.org`).

The Enterprise Server JRuby container maintains a pool of JRuby runtime instances for use by JRuby applications. The `configure-jruby-container` subcommand enables you to set the initial size, minimum size, and maximum size of this pool. The minimum size must be greater than zero. The initial size must be greater than or equal to the minimum size and less than or equal to the maximum size.

This subcommand is supported in remote mode only.

Options

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

`--monitoring`
: If set to `true`, enables monitoring for the Enterprise Server JRuby container. The default is `false`.

`--jruby-home`
: The directory where JRuby itself (*not* the Enterprise Server JRuby container) is installed.

  The directory must exist. Otherwise, an error occurs. However, the subcommand does *not* check whether JRuby is installed in the directory.

  The default is *as-install*`/jruby`, which is the directory where Update Tool installs JRuby. Therefore, if you obtained JRuby from Update Tool, this option is not required.

`--jruby-runtime`
: The initial number of JRuby runtime instances in the pool.

This number must be greater than zero, greater than or equal to `--jruby-runtime-min`, and less than or equal to `--jruby-runtime-max`.

The default is 1.

`--jruby-runtime-min`
The minimum number of JRuby runtime instances in the pool.

This number must be greater than zero, and less than or equal to `--jruby-runtime` and `--jruby-runtime-max`.

The default is 1.

`--jruby-runtime-max`
The maximum number of JRuby runtime instances in the pool.

This number must be greater than zero, and greater than or equal to `--jruby-runtime` and `--jruby-runtime-min`.

The default is 1.

`--show`
If set to `true`, displays the current settings of the Enterprise Server JRuby container. The default is `true`.

Examples

EXAMPLE 1 Setting the Directory Where JRuby Is Installed

This example sets the directory where JRuby is installed to `/tools/jruby`.

```
asadmin> configure-jruby-container --jruby-home=/tools/jruby
Successfully updated jruby-home to the new value: /tools/jruby

Current JRuby Container configuration:
jruby-home=/tools/jruby
max-pool-size=1
initial-pool-size=1
min-pool-size=1
monitoring=false

Command configure-jruby-container executed successfully.
```

EXAMPLE 2 Configuring the JRuby Runtime Pool

This example configures the JRuby runtime pool as follows:

- The initial number of JRuby runtime instances in the pool is 3.
- The minimum number of JRuby runtime instances in the pool is 2.
- The maximum number of JRuby runtime instances in the pool is 5.

The current settings of the Enterprise Server JRuby Container are not displayed.

EXAMPLE 2 Configuring the JRuby Runtime Pool *(Continued)*

```
asadmin> configure-jruby-container --show=false
--jruby-runtime=3
--jruby-runtime-min=2
--jruby-runtime-max=5
Successfully updated JRuby runtime pool configuration. Updated values are,
jruby-runtime: 3, jruby-runtime-min: 2, jruby-runtime-max: 5


Command configure-jruby-container executed successfully.
```

EXAMPLE 3 Displaying the Current Settings of the Enterprise Server JRuby Container

This command displays the current settings of an Enterprise Server JRuby container that is configured as follows:

- The directory where JRuby is installed is `/tools/jruby`.
- The initial number of JRuby runtime instances in the pool is 3.
- The minimum number of JRuby runtime instances in the pool is 2.
- The maximum number of JRuby runtime instances in the pool is 5.
- Monitoring for the Enterprise Server JRuby Container is not enabled.

```
asadmin> configure-jruby-container --show=true
Current JRuby Container configuration:
jruby-home=/tools/jruby
min-pool-size=2
initial-pool-size=3
max-pool-size=5
monitoring=false

Command configure-jruby-container executed successfully.
```

Exit Status

0	command executed successfully
1	error in executing the command

See Also `asadmin`(1M)

Chapter 1, "Using JRuby on Rails With Sun GlassFish Enterprise Server," in *Sun GlassFish Enterprise Server v3 Scripting Framework Guide*

Name 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. The application server must be running.

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)`

Name 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]*]
jndi_name
```

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`
: Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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)`

`asadmin(1M)`

Name create-audit-module – adds an audit module

Synopsis

```
create-audit-module   --classname classname
[--help]

 [ --property (name=value)[:name=value]*]
  audit_module_name
```

Description This subcommand adds the named audit module for the plug-in module that implements the audit capabilities. Audit modules collect and store information on incoming requests (servlets, 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, defaults to `com.sun.enterprise.security.Audit`.

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

`--property`
Optional attributes name/value pairs of provider implementation specific attributes.

The only valid property is `auditOn`, which can be specified as `true` or `false`. If `true`, causes the loading of the audit module and ensures that it is called by the Enterprise Server's audit library at audit points.

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name 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`
This option is valid for Enterprise Server release prior to v3. This option has been retained for backward compatibility. In v3 and later, this option is ignored.

`--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`, 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 `FileRealm`, `CertificateRealm`, `JDBCRealm`, `LDAPRealm`, and `SolarisRealm`.

`jaas-context`
Specifies the JAAS (Java Authentication and Authorization Service) 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`:

`clientAuth`
If `true`, specifies that client authentication is required for all applications that use the certificate realm. The default is `false`.

To require client authentication for a specific web application, set the method of authentication in the `web.xml` file to `CLIENT-CERT`.

- 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-table`
Specify the group table for an authentication realm of class `JDBCRealm`.

`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 `MD5`. You can use any algorithm supported in the JDK, or none.

`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.

`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

```
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
Command create-auth-realm executed successfully
```

Where `file` is the authentication realm created.

Exit Status

0	command executed successfully
1	error in executing the command

See Also

`delete-auth-realm(1)`, `list-auth-realms(1)`

`asadmin(1M)`

Name 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]
[--poolresize poolresize]
[--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]
[--description description]
[--ping {false|true}]
[--pooling {true|false}]
[--property (name=value)[:name=value]*]
poolname
```

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 Admin 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. By limiting the maximum number of times that a connection can be reused, you can avoid statement leaks. 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.

`--raname`
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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently 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 request 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 specifies that the connection is never validated.

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`.

```
asadmin> create-connector-connection-pool --raname 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)`

Name 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 ovjecttype]
[--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 application server.

  `system-admin`
  : A system resource only for the domain application server.

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

  `user`
  : A user resource.

`--description`
: Text providing details about the connector resource.

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

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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
```

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

See Also `delete-connector-resource(1)`, `list-connector-resources(1)`

`asadmin(1M)`

Name 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
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 transaction-based 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.

Operands

mapname
The name of the security map to be created or updated.

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)`

Name 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 application 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 application 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 application 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 application server principal.

```
asadmin create-connector-work-security-map --raname my-resource-adapter
--principalsmap eis-principal-1=server-principal-1,eis-principal-2
```

EXAMPLE 1 Creating a Connector Work Security Map (Principal) *(Continued)*

```
=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 application server user group.

```
asadmin create-connector-work-security-map --raname my-resource-adapter
--groupsmap 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)`

Name create-custom-resource – creates a custom resource

Synopsis
```
create-custom-resource [--help] --restype type --factoryclassname classname [--enabled={true|false}]
[--property (name=value)[:name=value]*] jndi-name
```

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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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 `delete-custom-resource(1)`, `list-custom-resources(1)`

`asadmin(1M)`

Name 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}]
[--domainproperties (name=value)[:name=value]*]
[--keytooloptions (name=value)[:name=value]*]
[--savelogin={false|true}]
[--checkports={true|false}]
[--nopassword={false|true}]
domain_name
```

Description An Enterprise Server domain is a Java EE-6 compliant administrative namespace. 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 Enterprise Server. A domain can exist independent 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 choice. 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 `domain.xml` file when the `create-domain` subcommand is run, the customizers are processed.

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.

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.

`--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.

`--portbase`
Determines the number with which the port assignment should start. A domain uses a certain number of ports that are statically assigned. The portbase value determines where the assignment should start. Choose this value carefully. The values for the ports are calculated as follows: Admin port: portbase + 48, HTTP listener port: portbase + 80, IIOP listener port: portbase + 37, JMX port: portbase + 86. See the output of this subcommand for a complete list of occupied ports, when `--portbase` option is specified. The `--portbase` option cannot be used with the `--adminport`, `--instanceport`, or the `--domainproperties` option.

Note – This subcommand uses some ports that are not required. This behavior is retained for compatibility with other releases.

`--profile`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--template`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--domaindir`
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*`glassfish/domains`.

`--savemasterpassword`
Setting this option to true allows the `masterpassword` to be written to the file system. 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.

`--domainproperties`
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:

`jms.port`
Specifies the port number for JMS. Valid value is 7676.

`domain.jmxPort`
Specifies the port on which the JMX connector is initialized. The valid values are 1-65535.

`orb.listener.port`
Specifies the ORB listener port for IIOP connections on which `orb-listener-1` listens.

`http.ssl.port`
Specifies the port number for `http-listener-2`. Valid values are 1 to 65535. On UNIX, to create sockets that listen on ports 1–1024, you need superuser privileges.

`orb.ssl.port`
Specifies the ORB listener port for IIOP connections on which the IIOP listener called `SSL` listens.

`orb.mutualauth.port`
Specifies the ORB listener port for IIOP connections on which the IIOP listener called `SSL_MUTUALAUTH` listens.

`osgi.shell.telnet.port`
Specifies the port for connecting to the Felix shell service that Enterprise Server provides to interact with the OSGi runtime. The default value is 6666.

`--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 machine where the `create-domain` subcommand is run.

`--savelogin`
If set to true, this option saves the admin 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 admin 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 having the same admin port number on the same or different machines (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 Admin, 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.

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.
```

EXAMPLE 1 Creating a Domain *(Continued)*

```
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=Sun Microsystems,L=Santa Clara,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=Sun Microsystems,L=Santa Clara,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 Admin Username 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.
```

EXAMPLE 3 Creating a Domain and Saving the Admin Username and Password *(Continued)*

```
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 ServiceDistinguished Name of the self-signed X.509 Server Certificate is:
[CN=sr1-usca-22,OU=GlassFish,O=Sun Microsystems,L=Santa Clara,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=Sun Microsystems,L=Santa Clara,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.
```

EXAMPLE 4 Creating a Domain and Designating the Certificate Host *(Continued)*

```
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)`

Name create-file-user – creates a new file user

Synopsis

```
create-file-user
[--help]

 [ --authrealmname  auth_realm_name]
[--groups  user_groups[:user_groups]*]
 user_name
```

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`.

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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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

`--authrealmname`
This is the file where the file users are stored.

Operands

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

Examples

EXAMPLE 1 Creating a User in the File Realm

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
```

Where, the `sample_user` is the file user created.

Exit Status

0	command executed successfully
1	error in executing the command

See Also create-auth-realm(1), delete-file-user(1), list-file-users(1), update-file-user(1), list-file-groups(1)

asadmin(1M)

Name create-http – sets HTTP parameters for a protocol

Synopsis

```
create-http
[--help]
[ --request-timeout-seconds  timeout]
[--timeout-seconds timeout]
[--max-connection max-keepalive]
--default-virtual-server virtual-server
[--dns-lookup-enabled={false|true}] 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.

`--request-timeout-seconds`
The time in seconds at which the request times out. The default is `30`.

`--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 `250`.

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

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

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)

Name 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]
[--acceptorthreads 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.

`--default-virtual-server` or `--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/2.4` and `X-Powered-By: JSP/2.0` headers to the appropriate responses. The Servlet 2.4 specification defines the `X-Powered-By: Servlet/2.4` 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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

listener_id	The listener ID of the HTTP network listener.

Examples

EXAMPLE 1 Using the `create-http-listener` subcommand

The following command creates an HTTP network listener named `sampleListener` that uses a non-default 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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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

Examples **EXAMPLE 1** Using the create-iiop-listener subcommand

The following command creates an IIOP listener named `sample_iiop_listener`:

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

Exit Status

0	command executed successfully
1	error in executing the command

See Also `delete-iiop-listener(1)`, `list-iiop-listeners(1)`, `create-ssl(1)`

`asadmin(1M)`

Name create-javamail-resource – creates a JavaMail session resource

Synopsis

```
create-javamail-resource [--help] [--target target] --mailhost hostname
--mailuser username --fromaddress address [--storeprotocol storeprotocol]
[--storeprotocolclass storeprotocolclass] [--transprotocol transprotocol]
[--transprotocolclass transprotocolclass] [--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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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*.

`--storeprotocol`
The mail server store protocol. The default is `imap`. Change this value only if you have reconfigured the Application 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 Application 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 Application 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 Application 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. Enterprise Server-specific `mail-` prefix is converted to 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
1	error in executing the subcommand

See Also

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

`asadmin(1M)`

Name 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]
[--poolresize=poolresizelimit]
[--idletimeout=idletimeout]
[--initsql=initsqlstring]
[--isolationlevel=isolationlevel]
[--isisolationguaranteed={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}]
[--creationretryattempts=creationretryattempts]
[--creationretryinterval=creationretryinterval]
[--sqltracelisteners=sqltracelisteners[,sqltracelisteners]
[--statementtimeout=statementtimeout]
[--lazyconnectionenlistment={false|true}]
[--lazyconnectionassociation={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. You can also specify the `--datasourceclassname` option.
- If `--restype` = `javax.sql.DataSource`, `javax.sql.XADataSource` or , `javax.sql.ConnectionPoolDataSource`, then the `--datasourceclassname` option is required. You can also specify the `--driverclassname` option.
- 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 programmatically risk polluting the pool. This could lead to program errors.

`--isisolationguaranteed`
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 within which a connection is validated at most once. Setting this attribute to an appropriate value minimizes the number of validation requests by a connection. The default value is 0, which specifies that the connection is never validated.

`--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).

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

`--creationretryattempts`
Specifies the maximum number of times that Enterprise Server retries to create a connection if the initial attempt fails. The default value is 0, which specifies that Enterprise 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, Enterprise 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 the application 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.

`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.

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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently 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`.

```
asadmin> create-jdbc-connection-pool
--host localhost --port 7070
--datasourceclassname org.apache.derby.jdbc.ClientDataSource
--restype javax.sql.XADataSource
```

EXAMPLE 1 Creating a JDBC Connection Pool *(Continued)*

```
--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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

`asadmin(1M)`

Name create-jmsdest – creates a JMS physical destination

Synopsis

```
create-jmsdest [--help]
[--target target]
--desttype dest_type [--property (name=value)[:name=value]*]
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.

Options

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

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--desttype`
`-T`
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 MQ 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_OLDEST`	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_OLDEST` 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_OLDEST`, 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:

```
"http://foo/flap.xsd http://test.com/test.xsd"
```

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 Chapter 18, "Physical Destination Property Reference," in *Sun GlassFish Message Queue 4.4 Administration Guide* for more information.

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)`

Name create-jms-host – creates a JMS host

Synopsis

```
create-jms-host [--help]
[--target target]
[--mqhost mq-host] [--mqport mq-port]
[--mquser mq-user] [--mqpassword mq-password]
jms_host_name
```

Description Creates a Java Message Service (JMS) host within the JMS service. 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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--mqhost`
The host name for the JMS service. The default value is `localhost`.

`--mqport`
The port number used by the JMS service. The default value is 7676.

`--mquser`
The user name for the JMS service. The default value is `admin`.

`--mqpassword`
The password for the JMS service. The default value is `admin`.

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`.

```
asadmin> create-jms-host --mqhost pigeon --mqport 7677 MyNewHost
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)`

Name create-jms-resource – creates a JMS resource

Synopsis

```
create-jms-resource [--help]
[--target target]
--restype type [--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.

Options

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

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--restype`
The JMS resource type, which can be `javax.jms.Topic`, `javax.jms.Queue`, `javax.jms.ConnectionFactory`, `javax.jms.TopicConnectionFactory`, or `javax.jms.QueueConnectionFactory`.

`--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). The default value is an address list 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
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
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)`

Name 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 (property=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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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 attribute name/value pairs for configuring the resource. The following properties are available:

`http-listener-1-port`
Specifies the port number for `http-listener-1`. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

`http-listener-2-port`
Specifies the port number for `http-listener-2`. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.

`orb-listener-1-port`
Specifies the ORB listener port for IIOP connections that `orb-listener-1` listens on.

`IIOP_SSL_LISTENER_PORT`
Specifies the ORB listener port for IIOP connections that the IIOP listener called `SSL` listens on.

`IIOP_SSL_MUTUALAUTH_PORT`
Specifies the ORB listener port for IIOP connections that the IIOP listener called `SSL_MUTUALAUTH` listens on.

`JMX_SYSTEM_Connector-port`
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.

Operands

jndi_name
The unique name of the JNDI resource to be created.

Examples

EXAMPLE 1 Creating a JNDI Resource

This example creates a new JNDI resource called `sample_jndi_resource`.

```
asadmin> create-jndi-resource --restype queue --factoryclass sampleClass
--jndilookupname sample_jndi --description "a sample JNDI resource" sample_jndi_resource
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)`

`asadmin(1M)`

Name create-jvm-options – creates options for the Java application launcher

Synopsis

```
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 Enterprise Server is started. The options that this subcommand creates are in addition to the options that are preset with Enterprise 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:

  ```
  -Djava.security.manager
  ```

  ```
  -Denvironment=Production
  ```

- **Startup parameters for the Java application launcher.** These options are preceded by the dash character (`-`). For example:

  ```
  --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 Enterprise Server starts. Options for the Java application launcher are written to the `server.log` file before any other information when Enterprise 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`
: Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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`.

  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-profiler(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:

- Solaris™ Operating System (Solaris OS) 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`)

Name create-lifecycle-module – adds 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 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--classpath`
This option indicates where this module is actually located if it is not under the applications root directory.

`--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.

`--failurefatal`
This options 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 resource is enabled at runtime. The default value is true.

`--description`
This is the text description of the resource associated with this module.

`--property`
This is an optional attribute containing name/value pairs used to configure the resource.

Operands *module_name*
This operand is a unique identifier for the deployed server lifecycle event listener module.

Examples **EXAMPLE 1** Using create-lifecycle-module

```
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
```

Where: `customSetup` is the lifecycle module created. The escape character `\` is used in the property option to distinguish the colons (:).

Exit Status

0	command executed successfully
1	error in executing the command

See Also `delete-lifecycle-module(1)`, `list-lifecycle-modules(1)`

`asadmin(1M)`

Name 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]

  --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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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	error in executing the command

See Also `delete-message-security-provider(1)`, `list-message-security-providers(1)`

`asadmin(1M)`

Name 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}] 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.

Operands

listener-name	The name of the network listener.

Examples

EXAMPLE 1 Using the `create-network-listener` subcommand

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)`

Name create-password-alias – creates a password alias

Synopsis
```
create-password-alias
[--help]
 aliasname
```

Description This subcommand creates an alias for a password and stores it in `domain.xml`. 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 `create-password-alias` command 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
The name of the alias password as it appears in the `domain.xml` file.

Examples **EXAMPLE 1** Creating a Password Alias

```
asadmin> create-password-alias
--interactive=true jmspassword-alias
Please enter the alias password>
Please enter the alias password again>
Command create-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)`, `update-password-alias(1)`

`asadmin(1M)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--classpath`
Java classpath string that specifies the classes needed by the profiler.

`--nativelibpath`
This path is automatically constructed to be a concatenation of the Application 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)`

Name create-protocol – adds a new protocol

Synopsis

```
create-protocol
[--help]
[ --securityenabled ={false|true}] protocol_name
```

Description The `create-protocol` subcommand creates a protocol for a network listener. 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`.

Operands

protocol-name	The name of the protocol.

Examples

EXAMPLE 1 Using the `create-protocol` subcommand

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)`

Name create-resource-adapter-config – creates the configuration information for the connector module

Synopsis

```
create-resource-adapter-config [--help] [--threadpoolid threadpool] [--objecttype user]
[--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 command 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`

`--property`
This option specifies the configuration properties of the resource adapter java bean. The properties can be specified as name value pairs separated by a colon (:). Names of setter methods of the class 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 Con figuration

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)`

Name create-service – configures the starting of a DAS on an unattended boot

Synopsis
```
create-service [--name name] [--serviceproperties serviceproperties]
[--dry-run={false|true}] [--domaindir domaindir] [domain_name]
```

Description The `create-service` subcommand configures the starting of a domain administration server (DAS) on an unattended boot by using the Solaris 10 Service Management Facility (SMF). This subcommand must be run as the OS-level user with superuser privileges. After this subcommand creates the service, you must start, enable, disable, delete, or stop the service. The DAS configuration must be stored on a folder to which the superuser has access and cannot be stored on a network file system. The service is created such that it is controlled by the OS-level user who owns the folder where the configuration of the DAS resides. If no arguments are specified, the subcommand uses the default domain.

To run this subcommand, you must have `solaris.smf.*` authorization. (Refer to the `useradd` and `usermod` man pages.) On Solaris, the manifest file is created in the `/var/svc/manifest/application/GlassFish/`*domain-name_domain-root-dir* directory. You must also have write permission in the directory tree `/var/svc/manifest/application/GlassFish`. Usually, the superuser has both these permissions. To run these commands as non-root user, the system administrator must be contacted so that the relevant authorizations are granted. You need to also ensure that:

- Solaris 10 administration commands such as `svccfg, svcs`, and `auths` are available in the PATH, so that these commands can be executed. A simple test to do so is to issue the subcommand, `which svccfg` on a bash shell.
- You must have write permission for the path `/var/svc/manifest/application/GlassFish`.

This subcommand is supported in local mode only.

Options

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

`--domaindir`
The directory where the domain is to be started. This is the absolute path of the directory on the disk that contains the configuration of the domain.

`--dry-run`
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.

`--name`
Tthe name of the service that you will use to run the Solaris SMF commands. If a default is present, this name overrides the default.

`--serviceproperties`
Specifies a : (colon)-separated list of various properties that are specific to the service. For Solaris 10, 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.

Operands *domain_name*
The name of the domain to be started. If no domain is specified, the default domain is used.

Examples **EXAMPLE 1** Creating a Service

This example creates a service for the default domain on a Solaris system.

```
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
/glassfishv3/glassfish/domains
Manifest file location on the system:/var/svc/manifest/application
/GlassFish/domain1_home_gfuser_glassfish-installations_glassfishv3
_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)`

Name create-ssl – creates and configures the SSL element in the selected HTTP listener, IIOP listener, or IIOP service

Synopsis

```
create-ssl
[--help]

 --type  listener_or_service_type  --certname cert_name
[--ssl2enabled=false ] [--ssl2ciphers ss12ciphers ]
[--ssl3enabled=true ] [--tlsenabled=true ]
[--ssl3tlsciphers ssl3tlsciphers ] [--tlsrollbackenabled=true ]
[--clientauthenabled=false ] [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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--type`
The type of service or listener for which the SSL is created. The type can be:

- `http-listener`
- `iiop-listener`
- `iiop-service`

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. Use the prefix + to enable or – to disable a particular cipher. Allowed values are:

- `rc4`
- `rc4export`
- `rc2`
- `rc2export`
- `idea`
- `des`
- `desede3`

If no value is specified, all supported ciphers are assumed to be enabled.

`--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. Use the prefix + to enable or – to disable a particular cipher. 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`

If no value is specified, all supported ciphers are assumed to be enabled.

`--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	command executed successfully
1	error in executing the command

See Also

`delete-ssl(1)`

`asadmin(1M)`

Name 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] [name=value)[:name=value]*]
```

Description The `create-system-properties` subcommand adds or updates system properties that can be referenced elsewhere on the server.

Enterprise Server provides hooks where tokens (system properties) can be specified. Because Enterprise 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

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.

```
asadmin> create-system-properties 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)`

Name create-threadpool – adds a thread pool

Synopsis
```
create-threadpool [--help] [--target target_name]
[--maxthreadpoolsize maxthreadpoolsize]
[--minthreadpoolsize minthreadpoolsize]
[--idletimeout idletimeout] [--maxqueuesize maxqueuesize]
[--workqueues workqueues] threadpool_id
```

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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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.

`--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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently 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)`

Name 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]
[--readtimeoutmillis read-timeout]
[--writetimeoutmillis write-timeout]
[--selectionkeyhandler selection-key-handler]
[--selectorpolltimeoutmillis selector-poll-timeout]
[--tcpnodelay={false|true}] 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 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`.

`--bytebuffertype`
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 `4096`.

`--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).

`--tcpnodelay`
If `true`, the default, enables `TCP_NODELAY` (also called Nagle's algorithm). The default is `false`.

Operands

transport-name	The name of the transport.

Examples

EXAMPLE 1 Using the `create-transport` subcommand

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)`

Name create-virtual-server – creates the named virtual server

Synopsis

```
create-virtual-server
[--help]
[ --target  target]
--hosts hosts
[--httplisteners http_listeners]
[--networklisteners network_listeners]
[--defaultwebmodule default_web_module]
[--state={on|off}]
[ --logfile  log_file]
[--property (name=value)[:name=value]*]
virtual_server_id
```

Description The `create-virtual-server` subcommand creates the named virtual server. Virtualization in the Enterprise 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.

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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.

`--httplisteners`
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 `--networklisteners` instead.

`--networklisteners`
A comma-separated (,) list of network listener IDs. Required only for a virtual server that is not the default virtual server.

`--defaultwebmodule`
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`
Specifies a comma-separated list of `Cache-Control` response directives. For a list of valid directives, see section 14.9 of the document at http://www.ietf.org/rfc/rfc2616.txt.

`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 stand-alone web applications deployed to this virtual server. A realm defined in a stand-alone 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 "sun-web-app" in *Sun GlassFish Enterprise Server v3 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 *Sun GlassFish Enterprise Server v3 Application Development Guide* and The Context Container (`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 Enterprise 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.

```
<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 "sun-web-app" in *Sun GlassFish Enterprise Server v3 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:

```
<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 Enterprise Server releases, or the `org.glassfish.web.valve.GlassFishValve` interface from the current Enterprise 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 "sun-web-app" in *Sun GlassFish Enterprise Server v3 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 "sun-web-app" in *Sun GlassFish Enterprise Server v3 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.

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 Using the create-virtual-server subcommand

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)

Name delete-admin-object – removes the administered object with the specified JNDI name.

Synopsis `delete-admin-object [--help] [--target` *target*`]` *jndi_name*

Description This subcommand removes the 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

jndi_name
JNDI name of the administered object to be deleted.

Examples **EXAMPLE 1** Deleting an Administered Object

This example deletes the administered object named `jms/samplelqueue`.

```
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)`

Name delete-audit-module – removes the named audit-module

Synopsis

```
delete-audit-module
[--help]
  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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name delete-auth-realm – removes the named authentication realm

Synopsis

```
delete-auth-realm
[--help]

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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

auth_realm_name	Name of the realm to be deleted.

Examples

EXAMPLE 1 Deleting an authentication realm

```
asadmin> delete-auth-realm
db
Command delete-auth-realm executed successfully
```

Where `db` is the authentication realm deleted.

Exit Status

0	command executed successfully
1	error in executing the command

See Also `create-auth-realm(1)`, `list-auth-realms(1)`

`asadmin(1M)`

Name 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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently 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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

jndi_name
The JNDI name of this connector resource.

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
```

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

See Also `create-connector-resource(1)`, `list-connector-resources(1)`

`asadmin(1M)`

Name 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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently 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`.

```
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)`

Name 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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name 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 in the default *install-dir*`/domains` directory 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	error in executing the subcommand

See Also `create-domain(1)`, `start-domain(1)`, `stop-domain(1)`, `list-domains(1)`

`asadmin(1M)`

Name delete-file-user – removes the named file user

Synopsis

```
delete-file-user
[--help]
[ --authrealmname  auth_realm_name]
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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

username	This is the name of file user to be deleted.

Examples

EXAMPLE 1 Deleting a User From a File Realm

```
asadmin> delete-file-user
 sample_user
Command delete-file-user executed successfully
```

Exit Status

0	command executed successfully
1	error in executing the command

See Also `create-file-user(1)`, `list-file-users(1)`, `update-file-user(1)`, `list-file-groups(1)`

`asadmin(1M)`

Name delete-http – removes HTTP parameters from a protocol

Synopsis

```
delete-http
[--help]
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.

Operands

protocol-name	The name of the protocol from which to delete the HTTP parameter set.

Examples **EXAMPLE 1** Using the delete-http subcommand

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)`

Name delete-http-listener – removes an HTTP network listener

Synopsis

```
delete-http-listener
[--help]
[ --target  target]
listener_id
```

Description The `delete-http-listener` subcommand removes the specified HTTP network listener. 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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

listener_id	The unique identifier for the HTTP network listener to be deleted.

Examples **EXAMPLE 1** Using the delete-http-listener subcommand

The following command deletes the HTTP 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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands *listener_id*
The unique identifier for the IIOP listener to be deleted.

Examples **EXAMPLE 1** Using the delete-iiop-listener subcommand

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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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`.

```
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)`

Name delete-jdbc-connection-pool – removes the specified JDBC connection pool

Synopsis

```
delete-jdbc-connection-pool [--help]
[--cascade={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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently 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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name delete-jmsdest – removes a JMS physical destination

Synopsis

```
delete-jmsdest [--help]
[--target target]
--desttype type
dest_name
```

Description The `delete-jmsdest` subcommand removes the specified Java Message Service (JMS) physical destination. 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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--desttype`
`-T`
The type of the JMS destination. Valid values are `topic` and `queue`.

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`.

```
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)`

Name delete-jms-host – removes a JMS host

Synopsis

```
delete-jms-host [--help]
[--target target]
jms_host_name
```

Description The subcommand removes the specified Java Message Service (JMS) host. This subcommand is supported in remote mode only.

Deleting the default JMS host, named `default_JMS_host`, is not recommended.

Options

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

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name 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.

Options

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

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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/Queue`.

```
asadmin> delete-jms-resource jms/Queue
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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name 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:

  ```
  -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`
: Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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:

- Solaris Operating System (Solaris OS) 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`)

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

module_name
This operand is a unique identifier for the deployed server lifecycle event listener module.

Examples

EXAMPLE 1 Using delete-lifecycle-module

```
asadmin> delete-lifecycle-module customSetup
Command delete-lifecycle-module executed successfully
```

Where: `customSetup` is the lifecycle module deleted.

Exit Status

0	command executed successfully
1	error in executing the command

See Also `create-lifecycle-module(1)`, `list-lifecycle-modules(1)`

`asadmin(1M)`

Name delete-message-security-provider – enables administrators to delete a message security provider

Synopsis

```
delete-message-security-provider
[--help]
   --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 Enterprise 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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)`

Name delete-network-listener – removes a network listener

Synopsis

```
delete-network-listener
[--help]
listener-name
```

Description The `delete-network-listener` command removes the specified network listener. This command is supported in remote mode only.

Options

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

Operands

listener-name	The name of the network listener to be deleted.

Examples

EXAMPLE 1 Using the delete-network-listener command

The following command deletes the network listener named `sampleListener`:

```
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)`

Name delete-password-alias – deletes 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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name delete-protocol – removes a protocol

Synopsis
```
delete-protocol
[--help]
protocol-name
```

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.

Operands

protocol-name	The name of the protocol to be deleted.

Examples

EXAMPLE 1 Using the delete-protocol subcommand

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)`

Name delete-resource-adapter-config – deletes the resource adapter configuration

Synopsis

```
delete-resource-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 `ra1`.

```
asadmin> delete-resource-adapter-config ra1
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)`

Name delete-ssl – deletes the SSL element in the selected HTTP listener, IIOP listener, or IIOP service

Synopsis

```
delete-ssl
[--help]
 --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`
: Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--type`
: The type of service or listener for which the SSL is deleted. The type must be one of the following types:

  - `http-listener`
  - `iiop-listener`
  - `iiop-service`

Operands *listener_id*
: The ID of the listener from which the SSL element is to be deleted.

  The *listener_id* operand is not required if the `--type` is `iiop-service`.

Examples **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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name delete-threadpool – removes a thread pool

Synopsis `delete-threadpool [--help] [--target` *target_name* `]` *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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name delete-transport – removes a transport

Synopsis

```
delete-transport
[--help]
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.

Operands

transport-name	The name of the transport to be deleted.

Examples

EXAMPLE 1 Using the delete-transport subcommand

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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

virtual_server_id	The unique identifier for the virtual server to be deleted.

Examples **EXAMPLE 1** Using the delete-virtual-server subcommand

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)`

Name 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}]
[--libraries jar_file[,jar_file]*]
[--target target]
[--type pkg-type]
[--properties(name=value)[:name=value]*]
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`).

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 Center, the syntax and semantics of the deployment descriptor is verified. Default is `false`.

`--name`
Name of the deployable component.

`--upload`
Uploads the deployable file to the administration server. The deployable file must be accessible from the client. If the file is accessible to both server and client, set the `--upload` option to `false`. The default value depends on whether the server you are deploying to is local or remote. If the server is local, the option defaults to false. If the server is remote, the option defaults to true. Explicitly specifying true or false overrides the default.

`--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`, `pointbase`, and `sybase`, case-insensitive. If not specified, the value of the `database-vendor-name` attribute in `sun-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 `sun-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 `sun-ejb-jar.xml` file determines whether or not tables are created.

`--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 `sun-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 `sun-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 containing Sun-specific descriptors. This should be passed along when deploying a pure EAR file. A pure EAR file is an EAR without Sun-specific descriptors.

`--enabled`
Allows users to access the application. If set to `false`, users will not be able to access the application. 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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 *instance-root*`/lib/applibs`. The libraries are made available to the application in the order specified.

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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.

Note – For properties that contain . (dot) separators in their names, using the `set` subcommand to change these properties requires a server restart. A better approach is to use the `redeploy` subcommand with the changed properties. If you do use the `set` subcommand, the . (dot) separators in these properties names must be escaped.

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, Enterprise 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 Enterprise 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.

`jruby.home`
: Specifies the directory where JRuby itself (not the Enterprise Server JRuby container) is installed. Default is *as-install*`/jruby`.

`jruby.runtime`
: Specifies the initial number of JRuby runtimes to start. Must be greater than zero, greater than or equal to `jruby.runtime.min`, and less than or equal to `jruby.runtime.max`. Default is 1. Overrides JRuby container runtime pool settings. For more information, see the `configure-jruby-container(1)` help page.

`jruby.runtime.min`
: Specifies the minimum number of JRuby runtimes in the pool. Must be greater than zero, less than or equal to `jruby.runtime` and `jruby.runtime.max`. Default is 1. Overrides JRuby container runtime pool settings. For more information, see the `configure-jruby-container(1)` help page.

`jruby.runtime.max`
: Specifies the maximum number of JRuby runtimes in the pool. Must be greater than zero, greater than or equal to `jruby.runtime` and `jruby.runtime.min`. Overrides JRuby container runtime pool settings. Default is 1. For more information, see the `configure-jruby-container(1)` help page.

`jruby.rackEnv`
: Specifies the environment in which a JRuby application such as Rails or Merb runs. Allowed values are development, production, or test. Default is development.

`jruby.applicationType`
: Specifies the name of a supported framework or the path to a script that initializes the user's framework. Allowed values corresponding to supported frameworks are Rails,

Merb, or Sinatra. Setting this property bypasses the normal, and potentially lengthy, auto-detection process and forces deployment on the specified framework. If the deployed application is not written for the specified framework, errors result. Default is computed through auto-detection.

`jruby.MTSafe`
If `true`, specifies that a framework being started using `jruby.applicationType` is thread-safe and therefore does not need a pool created for it. This property affects applications started using an auto-detected user-provided startup script. If `jruby.applicationType` is set and `jruby.MTsafe` is not set or is set to `false`, the application starts with a pool of application instances, and each instance of the application is accessed by one thread at a time. This property only affects frameworks being launched where the thread safety cannot be automatically determined. Setting `jruby.MTsafe` to `true` does not cause an auto-detected Rails 2.1.x application to be launched in thread-safe mode, nor can it be used to force a thread-safe framework to start in pooled mode. Default is computed through auto-detection.

`compatibility`
Specifies the Enterprise 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 GlassFish version 2 or Enterprise 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}`
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, Enterprise 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.

Other available properties are determined by the implementation of the component that is being redeployed.

Operands *filepath*

if the `--upload` option is set to `true`, this is the path to the deployable file on the local client machine. Otherwise, this is the absolute path to the file on the server machine.

Examples **EXAMPLE 1** Deploying an Enterprise Application

This example deploys the enterprise application packaged in the `Cart.ear` file.

```
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.

```
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.

```
asadmin> deploy --createtables=true 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.

```
asadmin> deploy 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)`, `configure-jruby-container(1)`

`asadmin(1M)`

Sun GlassFish Enterprise Server v3 Application Deployment Guide

Name 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]
[--uniquetablenames={true|false}]
[--dbvendorname dbvendorname]
[--createtables={false|true}|--dropandcreatetables={false|true}]
[--generatermistubs={false|true}]
[--availabilityenabled={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

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 Center, the syntax and semantics of the deployment descriptor is verified. Default is `false`.

`--name`
Name of the deployable component.

`--upload`
Uploads the deployable file to the administration server. The deployable file must be accessible from the client. If the file is accessible to both server and client, set the `--upload` option to `false`. The default value depends on whether the server you are deploying to is local or remote. If the server is local, the option defaults to false. If the server is remote, the option defaults to true. Explicitly specifying true or false overrides the default.

`--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`, `pointbase`, and `sybase`, case-insensitive. If not specified, the value of the `database-vendor-name` attribute in `sun-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 `sun-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 `sun-ejb-jar.xml` file determines whether or not tables are created.

`--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 `sun-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 `sun-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 containing Sun-specific descriptors. This should be passed along when deploying a pure EAR file. A pure EAR file is an EAR without Sun-specific descriptors.

`--enabled`
Allows users to access the application. If set to `false`, users will not be able to access the application. 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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 *instance-root*`/lib/applibs`. The libraries are made available to the application in the order specified.

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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.

Note – For properties that contain . (dot) separators in their names, using the `set` subcommand to change these properties requires a server restart. A better approach is to use the `redeploy` subcommand with the changed properties. If you do use the `set` subcommand, the . (dot) separators in these properties names must be escaped.

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, Enterprise 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 Enterprise 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.

`jruby.home`
Specifies the directory where JRuby itself (not the Enterprise Server JRuby container) is installed. Default is *as-install*`/jruby`.

`jruby.runtime`
Specifies the initial number of JRuby runtimes to start. Must be greater than zero, greater than or equal to `jruby.runtime.min`, and less than or equal to `jruby.runtime.max`. Default is 1. Overrides JRuby container runtime pool settings. For more information, see the `configure-jruby-container(1)` help page.

`jruby.runtime.min`
Specifies the minimum number of JRuby runtimes in the pool. Must be greater than zero, less than or equal to `jruby.runtime` and `jruby.runtime.max`. Default is 1. Overrides JRuby container runtime pool settings. For more information, see the `configure-jruby-container(1)` help page.

`jruby.runtime.max`
Specifies the maximum number of JRuby runtimes in the pool. Must be greater than zero, greater than or equal to `jruby.runtime` and `jruby.runtime.min`. Overrides JRuby container runtime pool settings. Default is 1. For more information, see the `configure-jruby-container(1)` help page.

`jruby.rackEnv`
Specifies the environment in which a JRuby application such as Rails or Merb runs. Allowed values are development, production, or test. Default is development.

`jruby.applicationType`
Specifies the name of a supported framework or the path to a script that initializes the user's framework. Allowed values corresponding to supported frameworks are Rails, Merb, or Sinatra. Setting this property bypasses the normal, and potentially lengthy, auto-detection process and forces deployment on the specified framework. If the deployed application is not written for the specified framework, errors result. Default is computed through auto-detection.

`jruby.MTSafe`
If `true`, specifies that a framework being started using `jruby.applicationType` is thread-safe and therefore does not need a pool created for it. This property affects applications started using an auto-detected user-provided startup script. If `jruby.applicationType` is set and `jruby.MTsafe` is not set or is set to `false`, the application starts with a pool of application instances, and each instance of the application is accessed by one thread at a time. This property only affects frameworks being launched where the thread safety cannot be automatically determined. Setting `jruby.MTsafe` to `true` does not cause an auto-detected Rails 2.1.x application to be launched in thread-safe mode, nor can it be used to force a thread-safe framework to start in pooled mode. Default is computed through auto-detection.

`compatibility`
Specifies the Enterprise 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 GlassFish version 2 or Enterprise 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}`
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, Enterprise 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.

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.

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.

```
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)`, `configure-jruby-container(1)`

`asadmin(1M)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

component_name	name of the component to be disabled.

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)`

Sun GlassFish Enterprise Server v3 Application Deployment Guide

Name 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 Enterprise 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.

Examples

EXAMPLE 1 Disabling the Monitoring Service for Enterprise Server

This example disables monitoring for Enterprise 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)`

Chapter 8, "Administering the Monitoring Service," in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

component_name	name of the component to be enabled.

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)`

Sun GlassFish Enterprise Server v3 Application Deployment Guide

Name 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 Enterprise 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 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`, `jruby-container`, and `web-services-container`. Additional modules can be listed by using the `get` subcommand.

To set the level of JRuby container monitoring you need to deploy at least one Ruby application or use the `configure-jruby-container` subcommand to enable monitoring. For more information, see `configure-jruby-container(1)`.

`--pid`
Specifies the Enterprise 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 Enterprise Serverprocess.

`--options`
Sets the following `btrace-agent` options:

`debug`
Enables debugging for BTrace. Default value is false.

Examples

EXAMPLE 1 Enabling the Monitoring Service for Enterprise Server

This example enables monitoring for Enterprise 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)`, `configure-jruby-container(1)`

`monitoring(5ASC)`

`asadmin(1M)`

Chapter 8, "Administering the Monitoring Service," in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name 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.

```
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.

EXAMPLE 3 Setting Multiple Environment Variables *(Continued)*

```
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)

Name flush-connection-pool – reintializes all connections established in the specified connection pool

Synopsis

```
flush-connection-pool [--help] pool_name
```

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.

Options

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

Operands

pool_name
Name of the connection pool to be reinitialized.

Examples 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)`

Name 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.

Options

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

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--desttype`
`-T`
This option indicates the type of physical destination from which you want to purge messages. The supported destination types are `topic` and `queue`.

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)`

Name freeze-transaction-service – freezes the transaction subsystem

Synopsis

```
freeze-transaction-service
[--help]
 [ --target  target]
```

Description The `freeze-transaction-service` subcommand freezes the transaction subsystem, suspending 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. This subcommand is supported in remote mode only.

Options

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

`--target`
The target server instance on which the subcommand is run.

Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands *target*
The name of the target server instance, typically `server`.

Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Chapter 15, "Using the Transaction Service," in *Sun GlassFish Enterprise Server v3 Application Development Guide*

Chapter 26, "Transactions," in *The Java EE 6 Tutorial, Volume I*

Name 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 Enterprise Server instance processes. This subcommand provides an alternative to sending Ctrl+Break or `kill -3` signals to Enterprise 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 Enterprise Server is being used.

If Enterprise 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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.

```
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
```

[1] The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform.

EXAMPLE 2 Obtaining Information for a Particular JVM Machine Type *(Continued)*

```
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	error in executing the subcommand

See Also `create-jvm-options(1)`, `delete-jvm-options(1)`, `list-jvm-options(1)`

`asadmin(1M)`

Name 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 Enterprise 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`.

```
asadmin> get server.http-service.http-listener.http-listener-1.*
server.http-service.http-listener.http-listener-1.acceptor-threads = 1
server.http-service.http-listener.http-listener-1.address = 0.0.0.0
server.http-service.http-listener.http-listener-1.blocking-enabled = false
server.http-service.http-listener.http-listener-1.default-virtual-server = server
server.http-service.http-listener.http-listener-1.enabled = true
server.http-service.http-listener.http-listener-1.external-port =
server.http-service.http-listener.http-listener-1.family = inet
server.http-service.http-listener.http-listener-1.id = http-listener-1
server.http-service.http-listener.http-listener-1.port = 8080
server.http-service.http-listener.http-listener-1.redirect-port =
server.http-service.http-listener.http-listener-1.security-enabled = false
server.http-service.http-listener.http-listener-1.server-name =
server.http-service.http-listener.http-listener-1.xpowered-by = true
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
```

EXAMPLE 3 Getting Attributes and Values for a Monitorable Object *(Continued)*

```
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)

Sun GlassFish Enterprise Server v3 Administration Guide

Name get-client-stubs – retrieves the client stub JAR file.

Synopsis

```
get-client-stubs
[--help]
 --appname  application_name[--target target] [local_directory_path]
```

Description The `get-client-stubs` subcommand copies the required JAR files for an `AppClient` stand-alone module or each `AppClient` module in an application from the server machine to the local directory. Each client's stub JAR file is retrieved separately, along with any required supporting JAR files . The client JAR file name is of the form *app-name*`Client.jar`. Before executing the `get-client-stubs` subcommand, you must deploy the application or module. The client stub 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.

`--target`
: Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

local_directory_path
: The path to the local directory where the client stub JAR file should be stored. The default is the current directory.

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)`

`appclient(1M)`, `asadmin(1M)`, `package-appclient(1M)`

Name jms-ping – checks if the JMS service is up and running

Synopsis

```
jms-ping [--help]
[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 Enterprise 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.

Options

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

Operands

target
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Examples **EXAMPLE 1** Verifying that the JMS service is running

The following subcommand checks to see if the JMS service is running.

```
asadmin> jms-ping
JMS-ping command executed successfully
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)`

Name 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 Enterprise 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:

  ```
  asamin> 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:

  ```
  asamin> 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` 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-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.

EXAMPLE 1 Listing Dotted Names of Configurable Elements *(Continued)*

```
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.jmx-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.iiop-service
configs.config.server-config.iiop-service.iiop-listener.SSL
configs.config.server-config.iiop-service.iiop-listener.SSL.ssl
configs.config.server-config.iiop-service.iiop-listener.SSL_MUTUALAUTH
configs.config.server-config.iiop-service.iiop-listener.SSL_MUTUALAUTH.ssl
configs.config.server-config.iiop-service.iiop-listener.orb-listener-1
configs.config.server-config.iiop-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
servers.server.server.resource-ref.jdbc/__default
```

EXAMPLE 1 Listing Dotted Names of Configurable Elements *(Continued)*

```
system-applications
Command list executed successfully.
```

EXAMPLE 2 Listing Attributes of a Configurable Element

This example lists the attributes of the web container.

```
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.

```
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)

Sun GlassFish Enterprise Server v3 Administration Guide

Name list-admin-objects – gets all the administered objects

Synopsis

```
list-admin-objects [--help]
```

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.

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
1	error in executing the subcommand

See Also `create-admin-object(1)`, `delete-admin-object(1)`

`asadmin(1M)`

Name list-applications – lists deployed applications

Synopsis
```
list-applications [--help] [--type type]
```

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.

This subcommand is supported in remote mode only.

Options

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

`--type`
Specifies the type of the applications that are to be listed. The options are as follows:

- `application`
- `appclient`
- `connector`
- `ejb`
- `jruby`
- `web`
- `webservice`

If no type is specified, all applications are listed.

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)`

Sun GlassFish Enterprise Server v3 Application Deployment Guide

Name list-audit-modules – gets all audit modules and displays them

Synopsis

```
list-audit-modules
[--help]
```

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
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name list-auth-realms – lists the authentication realms

Synopsis
```
list-auth-realms
[--help]
```

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_name
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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
1	error in executing the command

See Also `create-auth-realm(1)`, `delete-auth-realm(1)`

`asadmin(1M)`

Name list-commands – lists available commands

Synopsis

```
list-commands [--help] [--localonly={false|true}] [ --remoteonly ={false|true}]
```

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.

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.

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
```

EXAMPLE 1 Listing the Local Subcommands *(Continued)*

```
stop-domain
unset
verify-domain-xml
version
Command list-commands executed successfully.
```

EXAMPLE 2 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 **********
__locations                           enable
add-resources                         enable-monitoring
configure-jruby-container             flush-connection-pool
configure-ldap-for-admin              flush-jmsdest
create-admin-object                   freeze-transaction-service
create-audit-module                   generate-jvm-report
create-auth-realm                     get
create-connector-connection-pool      get-client-stubs
create-connector-resource             get-host-and-port
create-connector-security-map         jms-ping
create-connector-work-security-map    list
create-custom-resource                list-admin-objects
```

EXAMPLE 2 Listing All Subcommands *(Continued)*

```
create-file-user                        list-app-refs
create-http                             list-applications
create-http-listener                    list-audit-modules
create-iiop-listener                    list-auth-realms
create-javamail-resource                list-components
create-jdbc-connection-pool             list-connector-connection-pools
create-jdbc-resource                    list-connector-resources
create-jms-host                         list-connector-security-maps
create-jms-resource                     list-connector-work-security-maps
create-jmsdest                          list-containers
create-jndi-resource                    list-custom-resources
create-jvm-options                      list-file-groups
create-lifecycle-module                 list-file-users
create-message-security-provider        list-http-listeners
create-network-listener                 list-iiop-listeners
create-password-alias                   list-javamail-resources
create-profiler                         list-jdbc-connection-pools
create-protocol                         list-jdbc-resources
create-resource-adapter-config          list-jms-hosts
create-resource-ref                     list-jms-resources
create-ssl                              list-jmsdest
create-system-properties                list-jndi-entries
create-threadpool                       list-jndi-resources
create-transport                        list-jvm-options
create-virtual-server                   list-lifecycle-modules
delete-admin-object                     list-logger-levels
delete-audit-module                     list-message-security-providers
...
```

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)`

Name list-components – lists deployed components

Synopsis `list-components [--help] [--type` *type*`] [`*target*`]`

Description 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.

This subcommand is supported in remote mode only.

Options

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

`--type`
Specifies the type of the components that are to be listed. The options are as follows:

- `application`
- `appclient`
- `connector`
- `ejb`
- `jruby`
- `web`
- `webservice`

If no type is specified, all components are listed.

Operands

target
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Sun GlassFish Enterprise Server v3 Application Deployment Guide

Name list-connector-connection-pools – lists the existing connector connection pools

Synopsis

```
list-connector-connection-pools [--help]
```

Description The `list-connector-connection-pools` subcommand list 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	error in executing the subcommand

See Also `create-connector-connection-pool(1)`, `delete-connector-connection-pool(1)`, `ping-connection-pool(1)`

`asadmin(1M)`

Name list-connector-resources – lists all connector resources

Synopsis

```
list-connector-resources [--help]
```

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.

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
1	error in executing the subcommand

See Also `create-connector-resource(1)`, `delete-connector-resource(1)`

`asadmin(1M)`

Name 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}] 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`.

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)`

`asadmin(1M)`

Name 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-2, mapped principal=server-principal-2
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
1	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)`

Name 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 : jruby
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)`

Name list-custom-resources – gets all custom resources

Synopsis `list-custom-resources [--help] [--target` *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
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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.
```

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

See Also `create-custom-resource(1)`,`delete-custom-resource(1)`

`asadmin(1M)`

Name 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 that 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 default *as-install*`/domains` directory are listed.

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)`

Name list-file-groups – lists file groups

Synopsis

```
list-file-groups
[--help]
 [ --name  username]
 [--authrealmname auth_realm_name]
```

Description Use this subcommand to view 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.

Operands *target*
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Examples **EXAMPLE 1** Listing Groups in all File Realms

```
asadmin>list-file-groups
staff
manager
Command list-file-groups executed successfully
```

Exit Status

0	command executed successfully
1	error in executing the command

See Also `delete-file-user(1)`, `update-file-user(1)`, `create-file-user(1)`, `list-file-users(1)`

`asadmin(1M)`

Name list-file-users – lists the file users

Synopsis

```
list-file-users
[--help]
 [ --authrealmname  auth_realm_name]
```

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
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Examples **EXAMPLE 1** Listing Users in a Specific File Realm

```
asadmin> list-file-users sample_file_realm
sample_user05
sample_user08
sample_user12
Command list-file-users executed successfully
```

Exit Status

0	command executed successfully
1	error in executing the command

See Also `create-file-user(1)`, `delete-file-user(1)`, `update-file-user(1)`, `list-file-groups(1)`

`asadmin(1M)`

Name list-http-listeners – lists the existing HTTP network listeners

Synopsis

```
list-http-listeners
[--help]
[target]
```

Description The `list-http-listeners` subcommand lists the existing HTTP network listeners. This subcommand is supported in remote mode only.

Options

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

Operands

target
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Examples **EXAMPLE 1** Using the list-http-listeners subcommand

The following command lists all the HTTP 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)`

Name 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
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name list-javamail-resources – lists the existing JavaMail session resources

Synopsis

```
list-javamail-resources [--help [--target 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.

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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
1	error in executing the subcommand

See Also `create-javamail-resource(1)`, `delete-javamail-resource(1)`

`asadmin(1M)`

Name list-jdbc-connection-pools – lists all JDBC connection pools

Synopsis

```
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.

```
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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Examples **EXAMPLE 1** Listing the JDBC Resources

This example lists the current JDBC resources.

```
asadmin> list-jdbc-resources
jdbc/DerbyPool
Command list-jdbc-resources executed successfully.
```

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

See Also `create-jdbc-resource(1)`, `delete-jdbc-resource(1)`

`asadmin(1M)`

Name 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.

Options

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

`--desttype`
`-T`
The type of JMS destination to be listed. Valid values are `topic` and `queue`.

Operands

target
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name list-jms-hosts – lists the existing JMS hosts

Synopsis
```
list-jms-hosts [--help]
[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.

Options

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

Operands

target
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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

1 error in executing the subcommand

See Also `create-jms-host(1)`, `delete-jms-host(1)`, `jms-ping(1)`

`asadmin(1M)`

Name 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.

Options

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

`--restype`
The JMS resource type can be `javax.jms.Topic`, `javax.jms.Queue`, `javax.jms.ConnectionFactory`, `javax.jms.TopicConnectionFactory`, or `javax.jms.QueueConnectionFactory`.

Operands

target
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name 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.

This 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
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Examples **EXAMPLE 1** Using the list-jndi-entries subcommand

```
asadmin> list-jndi-entries
jndi_entry03
jndi_entry72
jndi_entry76
Command list-jndi-resources executed successfully
```

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

See Also `create-jndi-resource(1)`, `delete-jndi-resource(1)`

`asadmin(1M)`

Name list-jndi-resources – lists all existing JNDI resources

Synopsis `list-jndi-resources [--help] [--target` *target*`]`

Description Use the `list-jndi-resources` subcommand to identify all the existing JNDI resources.

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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Examples **EXAMPLE 1** Listing JNDI Resources

This example lists the JNDI in the default domain.

```
asadmin> list-jndi-resources
jndi_resource1
jndi_resource2
jndi_resource3
Command list-jndi-resources executed successfully
```

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

See Also `create-jndi-resource(1)`, `delete-jndi-resource(1)`

`asadmin(1M)`

Name list-jvm-options – lists options for the Java application launcher

Synopsis

```
list-jvm-options [--help]
```

Description The `list-jvm-options` subcommand displays a list of command-line options that are passed to the Java application launcher when Enterprise 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.

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
-client
-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}${com.sun.aas.derbyRoot}/lib
-Xmx512m
-XX:MaxPermSize=192m
-Djava.endorsed.dirs=${com.sun.aas.installRoot}/lib/endorsed
-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:

- Solaris Operating System (Solaris OS) 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`)

Name 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
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Examples **EXAMPLE 1** Using list-lifecycle-modules:

```
asadmin> list-lifecycle-modulesWSTCPConnectorLCModule
Command list-lifecycle-modules executed successfully
```

Where `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)`

Name list-logger-levels – lists the loggers and their log levels

Synopsis
```
list-logger-levels [--help]
```

Description The `list-logger-levels` subcommand lists the current Enterprise 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.

Examples **EXAMPLE 1** Listing the Log Levels

This example lists the existing loggers and indicates how their log levels are set.

```
asadmin> list-logger-levels
javax.enterprise.system.container.cmp: INFO
javax.enterprise.system.tools.admin: INFO
java.util.logging.ConsoleHandler: FINEST
javax.enterprise.system.container.web: INFO
javax.enterprise.system.util: INFO
javax.enterprise.resource.webcontainer.jsf.timing: INFO
javax: INFO
javax.enterprise.resource.corba: INFO
javax.enterprise.system.core.naming: INFO
javax.enterprise.system.core.selfmanagement: INFO
javax.enterprise.system.container.ejb: INFO
javax.enterprise.resource.webcontainer.jsf.config: INFO
javax.enterprise.resource.javamail: INFO
org.apache.catalina: INFO
javax.enterprise.system.core.config: INFO
javax.enterprise.system.webservices.rpc: INFO
javax.enterprise.system.webservices.registry: INFO
javax.enterprise.system.tools.deployment: INFO
javax.enterprise.resource.jms: INFO
javax.enterprise.system: INFO
javax.enterprise.system.webservices.saaj: INFO
org.apache.jasper: INFO
javax.enterprise.resource.webcontainer.jsf.lifecycle: INFO
javax.enterprise.resource.jta: INFO
javax.enterprise.resource.jdo: INFO
javax.enterprise.resource.resourceadapter: INFO
javax.enterprise.system.core.transaction: INFO
javax.enterprise.resource.webcontainer.jsf.resource: INFO
javax.enterprise.system.core.security: INFO
```

EXAMPLE 1 Listing the Log Levels *(Continued)*

```
javax.enterprise.resource.webcontainer.jsf.application: INFO
javax.enterprise.system.core.classloading: INFO
org.apache.coyote: INFO
javax.enterprise.resource.webcontainer.jsf.managedbean: INFO
javax.enterprise.system.container.ejb.mdb: INFO
javax.enterprise.resource.webcontainer.jsf.context: INFO
javax.enterprise.resource.webcontainer.jsf.renderkit: INFO
javax.enterprise.resource.webcontainer.jsf.facelets: INFO
javax.enterprise.resource.webcontainer.jsf.taglib: INFO

Command list-logger-levels executed successfully.
```

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

See Also `rotate-log`(1), `set-log-level`(1)

`asadmin`(1M)

Chapter 7, "Administering the Logging Service," in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name list-message-security-providers – enables administrators to list all security message providers (`provider-config` sub-elements) for the given message layer (`message-security-config` element of `domain.xml`)

Synopsis

```
list-message-security-providers
[--help]
  --layer    message_layer
```

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 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.

`--layer`
The message-layer for which the provider has to be listed. The default value is `HttpServlet`.

Operands *target*
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name list-modules – lists Enterprise Server modules

Synopsis

```
list-modules [--help]
```

Description The `list-modules` subcommand displays a list of modules that are accessible to the Enterprise 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 the Enterprise Server Modules

This example provides a partial listing of modules that are accessible to the Enterprise 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/dixiep/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 Characteristics : List of Jars implementing the module
        Jar : file:/home/dixiep/GlassFish/glassfishv3/glassfish/modules/kernel.jar
Module : org.glassfish.common.util:3.0.0.b66
    Module Characteristics : List of Jars implementing the module
        Jar : file:/home/dixiep/GlassFish/glassfishv3/glassfish/modules/common-util.jar
    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)`

Name list-network-listeners – lists the existing network listeners

Synopsis

```
list-network-listeners
[--help]
```

Description The `list-network-listeners` command lists the existing network listeners. This command is supported in remote mode only.

Options

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

Examples **EXAMPLE 1** Using the list-network-listeners command

The following command lists all the network listeners for the server instance:

```
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)`

Name list-password-aliases – lists all password aliases

Synopsis

```
list-password-aliases
[--help]
```

Description Thissubcommand 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)`

Name list-protocols – lists the existing protocols

Synopsis

```
list-protocols
[--help]
```

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.

Examples **EXAMPLE 1** Using the list-protocols subcommand

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)`

Name 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)`

Name list-sub-components – lists EJBs or Servlets in deployed module or module of deployed application

Synopsis

```
list-sub-components [--help] [--type type]
[--appname appname] modulename
```

Description The `list-sub-commands` subcommand lists EJBs or servlets 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 standalone. 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.

Operands

modulename
Specifies the name of the module containing the subcomponent.

Examples **EXAMPLE 1** Listing Subcomponents

This example lists the subcomponents of the `MEjbApp` application within the `mejb.jar` module.

```
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)`

Sun GlassFish Enterprise Server v3 Application Deployment Guide

Name list-system-properties – lists the system properties of the domain

Synopsis `list-system-properties [--help] [--target` *target*`]`

Description The `list-system-properties` subcommand lists the system properties of a domain.

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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Name list-threadpools – lists all the thread pools

Synopsis

```
list-threadpools [--help]
```

Description Lists the Enterprise Server thread pools.

This subcommand is supported in remote mode only.

Options

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

Examples **EXAMPLE 1** Listing Thread Pools

This example lists the current thread pools.

```
asadmin> list-threadpools
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)`

Name 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. This command is supported in remote mode only.

Options

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

Operands

target
The name of the standalone server instance, typically `server`.

Examples

EXAMPLE 1 Listing the Current Timers in a Server Instance

This example lists the 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 `asadmin(1M)`

"Using the Timer Service" in *The Java EE 6 Tutorial, Volume I*

"EJB Timer Service" in *Sun GlassFish Enterprise Server v3 Application Development Guide*

Name list-transports – lists the existing transports

Synopsis

```
list-transports
[--help]
```

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.

Examples **EXAMPLE 1** Using the list-transports subcommand

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
1	error in executing the command

See Also `create-transport(1)`, `delete-transport(1)`

`asadmin(1M)`

Name 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
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Examples **EXAMPLE 1** Using the list-virtual-servers subcommand

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)`

Name 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:

```
<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)`

Name 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:

```
<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.

```
asadmin> list-web-env-entry hello
Hello User (java.lang.String) = techscribe ignoreDescriptorItem=false
 //User authentication for Hello appplication
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)`

Name 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 Enterprise 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 admin 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 admin 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 login 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
```

EXAMPLE 1 Logging Into a Domain on a Remote Machine *(Continued)*

```
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 `mylhost` 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)`

Name 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 Enterprise 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.

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.

`pt`
The total amount of time (in milliseconds) that the HTTP listener service has spent in processing HTTP requests.

`rc`
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):[2]

`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.

[2] 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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently 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 other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently 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.
```

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)

Chapter 8, "Administering the Monitoring Service," in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name 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` *filename*
: Reads the subcommands as specified in *filename*.

`--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)`

Name ping-connection-pool – tests if a connection pool is usable

Synopsis

```
ping-connection-pool [--help] pool_name
```

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.

Options

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

Operands

pool_name
The name of the pool to test.

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)`

Name recover transactions – manually recovers pending transactions

Synopsis

```
recover-transactions
[--help]
 [ --txlogdir  transaction_log_dir ] [--destination destination_server_name ]
 server_name
```

Description The `recover-transactions` subcommand manually recovers pending transactions. This subcommand is used in remote mode only.

Options

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

`--transactionlogdir`
The location of the transaction logs for a particular server. When a server fails it writes the location in its transaction log.

If the failed server's transaction logs are copied to some other location to make it available to the surrogate recovery server, use this option to specify the new location of the transaction logs. If the failed server's transaction service `tx-log-dir` property is modified to reflect a new location, then this option is not required.

`--destination`
The name of the destination server on which the pending transactions need to be recovered.

Operands

server_name
The name of the server that failed. The in-flight transactions on this server will be recovered. For stand-alone servers the value of this operand is typically `server`.

Examples **EXAMPLE 1** Using recover-transactions

```
% asadmin recover-transactions server
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)`

Chapter 15, "Using the Transaction Service," in *Sun GlassFish Enterprise Server v3 Application Development Guide*

Chapter 26, "Transactions," in *The Java EE 6 Tutorial, Volume I*

Name 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]
[--libraries jar_file[,jar_file]*]
[--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 Center, the syntax and semantics of the deployment descriptor is verified. Default is `false`.

`--name`
Name of the deployable component.

`--upload`
Uploads the deployable file to the administration server. The deployable file must be accessible from the client. If the file is accessible to both server and client, set the `--upload` option to `false`. The default value depends on whether the server you are deploying to is local or remote. If the server is local, the option defaults to false. If the server is remote, the option defaults to true. Explicitly specifying true or false overrides the default.

`--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`, `pointbase`, and `sybase`, case-insensitive. If not specified, the value of the `database-vendor-name` attribute in `sun-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 `sun-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 `sun-ejb-jar.xml` file determines whether or not tables are created.

`--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 `sun-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 `sun-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 containing Sun-specific descriptors. This should be passed along when deploying a pure EAR file. A pure EAR file is an EAR without Sun-specific descriptors.

`--enabled`
Allows users to access the application. If set to `false`, users will not be able to access the application. 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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 *instance-root*`/lib/applibs`. The libraries are made available to the application in the order specified.

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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.

Note – For properties that contain . (dot) separators in their names, using the `set` subcommand to change these properties requires a server restart. A better approach is to use the `redeploy` subcommand with the changed properties. If you do use the `set` subcommand, the . (dot) separators in these properties names must be escaped.

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, Enterprise 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 Enterprise 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.

`jruby.home`
Specifies the directory where JRuby itself (not the Enterprise Server JRuby container) is installed. Default is *as-install*`/jruby`.

`jruby.runtime`
Specifies the initial number of JRuby runtimes to start. Must be greater than zero, greater than or equal to `jruby.runtime.min`, and less than or equal to `jruby.runtime.max`. Default is 1. Overrides JRuby container runtime pool settings. For more information, see the `configure-jruby-container(1)` help page.

`jruby.runtime.min`
Specifies the minimum number of JRuby runtimes in the pool. Must be greater than zero, less than or equal to `jruby.runtime` and `jruby.runtime.max`. Default is 1. Overrides JRuby container runtime pool settings. For more information, see the `configure-jruby-container(1)` help page.

`jruby.runtime.max`
Specifies the maximum number of JRuby runtimes in the pool. Must be greater than zero, greater than or equal to `jruby.runtime` and `jruby.runtime.min`. Overrides JRuby container runtime pool settings. Default is 1. For more information, see the `configure-jruby-container(1)` help page.

`jruby.rackEnv`
Specifies the environment in which a JRuby application such as Rails or Merb runs. Allowed values are development, production, or test. Default is development.

`jruby.applicationType`
Specifies the name of a supported framework or the path to a script that initializes the user's framework. Allowed values corresponding to supported frameworks are Rails, Merb, or Sinatra. Setting this property bypasses the normal, and potentially lengthy,

auto-detection process and forces deployment on the specified framework. If the deployed application is not written for the specified framework, errors result. Default is computed through auto-detection.

`jruby.MTSafe`
If `true`, specifies that a framework being started using `jruby.applicationType` is thread-safe and therefore does not need a pool created for it. This property affects applications started using an auto-detected user-provided startup script. If `jruby.applicationType` is set and `jruby.MTsafe` is not set or is set to `false`, the application starts with a pool of application instances, and each instance of the application is accessed by one thread at a time. This property only affects frameworks being launched where the thread safety cannot be automatically determined. Setting `jruby.MTsafe` to `true` does not cause an auto-detected Rails 2.1.x application to be launched in thread-safe mode, nor can it be used to force a thread-safe framework to start in pooled mode. Default is computed through auto-detection.

`compatibility`
Specifies the Enterprise 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 GlassFish version 2 or Enterprise 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}`
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, Enterprise 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.

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`
- A directory that contains the exploded format of the deployable archive

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.

  If this operand is omitted, the path is retrieved from the `domain.xml` file. Otherwise, the operand can specify a directory or an archive file.

Examples **EXAMPLE 1** Redeploying a Web Application From a File

This example redeploys 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 redeploys the web application `hellodir`. The application was originally deployed from a directory. The path is retrieved from the `domain.xml` file.

```
asadmin> redeploy --name hellodir
Application deployed successfully with name hellodir.
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)`, `configure-jruby-container(1)`

`asadmin(1M)`

Sun GlassFish Enterprise Server v3 Application Deployment Guide

Name restart-domain – restarts the DAS of the specified domain

Synopsis

```
restart-domain [--help] [--domaindir domaindir] [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 there are two or more domains, the *domain_name* operand must be specified. If the server is not already running, the subcommand attempts to restart 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.

`--domaindir`
The directory of the domain that is to be restarted. If specified, the path must be accessible in the file system. If not specified, the domain in the default *as-install*`/glassfish/domains` directory is restarted.

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)`

Name 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. This subcommand is supported in remote mode only.

Options

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

`--target`
The target server instance on which the subcommand is run.

Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

target
The target server instance on which you are rolling back the transactions.

Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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 succeessfully
```

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)`

Chapter 15, "Using the Transaction Service," in *Sun GlassFish Enterprise Server v3 Application Development Guide*

Chapter 26, "Transactions," in *The Java EE 6 Tutorial, Volume I*

Name 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.

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

Examples **EXAMPLE 1** Rotating the Server Log

This example rotates the server log.

```
asadmin> rotate-log
Command rotate-log executed successfully.
```

See Also `list-logger-levels(1)`, `set-log-level(1)`

`asadmin(1M)`

Chapter 7, "Administering the Logging Service," in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name 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 Enterprise 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.

EXAMPLE 2 Enabling the Monitoring Service for a Monitorable Object *(Continued)*

```
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)

Sun GlassFish Enterprise Server v3 Administration Guide

Name set-log-level – sets the log level for one or more loggers

Synopsis `set-log-level [--help]` *logger-name=logger-level*`[:`*logger-name=logger-level*`]*`

Description The `set-log-level` subcommand sets the log level for one or more loggers. Changes take effect dynamically, that is, server restart is not required.

This subcommand is supported in remote mode only.

Options

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

Operands

logger-name
The name of the logger. The `list-logger-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-level 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`:

```
asadmin set-log-level javax.enterprise.system.container.web=FINE:
javax.enterprise.system.container.ejb=SEVERE
Command set-log-level executed successfully.
```

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

See Also `list-logger-levels(1)`, `rotate-log(1)`

`asadmin(1M)`

Chapter 7, "Administering the Logging Service," in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name 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)`

Name 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:

```
<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 appplication" 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)`

Name 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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

component-name
The name of the component whose status is to be listed.

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-components(1)`, `list-sub-components(1)`

`asadmin(1M)`

Sun GlassFish Enterprise Server v3 Application Deployment Guide

Name start-database – starts the Java DB

Synopsis
```
start-database [--help] [--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 Enterprise 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.

`--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.
```

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)

Chapter 14, "Administering Database Connectivity," in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name start-domain – starts a domain

Synopsis
```
start-domain [--help] [--debug ={true|false}] [--domaindir domaindir]
 [--upgrade={true|false}] [--verbose={true|false}] [domain_name]
```

Description The `start-domain` subcommand starts the specified domain. If the domain directory is not specified, the domain in the default domains directory is started. If there are 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.

Options

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

`--debug`
If set to true, the server is started in debug mode and prints the JPDA port on the console. Default is false.

`--domaindir`
Specifies the directory where the domain to be started is located. If specified, the path must be accessible in the file system. If not specified, the domain in the default *install-dir*`/glassfish/domains` directory is started.

`--verbose`
`-v`
If set to true, a console window is opened in which detailed server startup messages and log messages are displayed until the server is started. If the domain is later restarted by using the `restart-domain` subcommand, issued from a different console window, messages continue to be displayed in the original console window. You can kill the server by typing `CTRL-C`, or by getting a thread dump for the server by typing `CTRL-\` on UNIX® systems, or `CTRL-Break` on Windows systems. Default is false.

`--upgrade`
If set to true, the server is upgraded from a previous release. The server is started, the configuration is modified to be compatible with this release of Enterprise Server, and the server process stops. Normally, if the `start-domain` subcommand detects that the configuration is from an older release of Enterprise Server, the domain is upgraded automatically before being started. You should not need to use this option explicitly. Default is false.

Operands

domain_name The unique name of the domain you want to start. Default is the name specified during installation, usually `domain1`.

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
Log file: /myhome/glassfishv3/glassfish/domains/mydomain4/logs/server.log
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)`

Name stop-database – stops the Java DB

Synopsis `stop-database [--help] [--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 Enterprise 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.

`--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)`

Chapter 14, "Administering Database Connectivity ," in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name stop-domain – stops the Domain Administration Server of the specified domain

Synopsis

```
stop-domain [--help] [--domaindir domaindir] [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*`/glassfish/domains` directory is stopped.

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)`

Name 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 `sun-ejb-jar.xml` file determines whether or not tables are dropped. Default value is true.

`--target`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

name	Name of the deployed component.

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.

EXAMPLE 2 Undeploying an Enterprise Bean With Container-Managed Persistence (CMP) *(Continued)*

```
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)`

Sun GlassFish Enterprise Server v3 Application Deployment Guide

Name 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 suspended 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`
The target server instance on which the subcommand is run.

Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Operands

target
The name of the target server instance, typically `server`.

Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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)`

Chapter 15, "Using the Transaction Service," in *Sun GlassFish Enterprise Server v3 Application Development Guide*

Chapter 26, "Transactions," in *The Java EE 6 Tutorial, Volume I*

Name unset – removes one or more variables from the multimode environment

Synopsis

```
unset [--help] [[variable-name]*]
```

Description The `unset` subcommand removes one or more environment variables that you set for the multimode environment. After removal, the variables and their associated values will no longer apply to the multimode environment.

This subcommand is supported in local mode only.

Options

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

Operands

variable-name Environment variable to be removed. To list the environment variables that are set, use the `export` subcommand without options. If no environment variables are listed, then you have not set any.

Examples

EXAMPLE 1 Listing the Environment Variables That Are Set

This example uses the `export` subcommand to view the environment variables that have been set.

```
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.

```
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)`

Name 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:

```
<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)`

Name 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 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 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:

```
<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)`

Name 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_name1]*
|--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)`

Name 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** Updating a Connector Work Security Map

This example updates `workSecurityMap2` by removing `eis-group-2`.

```
asadmin> update-connector-work-security-map
--raname 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)`

Name update-file-user – updates a current file user as specified

Synopsis

```
update-file-user
[--help]
[ --groups  user_groups[:user_groups]*]
[--authrealmname authrealm_name]
username
```

Description This 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 (:).

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`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

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

```
asadmin> update-file-user
 --groups staff:manager:engineer sample_user
Command update-file-user executed successfully
```

Where *sample_user* is the file user for whom the groups and the user name are updated.

Exit Status

0	command executed successfully
1	error in executing the command

See Also `delete-file-user(1)`, `list-file-users(1)`, `create-file-user(1)`, `list-file-groups(1)`

`asadmin(1M)`

Name 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)`

Name 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)`

Name verify-domain-xml – verifies the content of the domain.xml file

Synopsis

```
verify-domain-xml[--help]
[--domaindir domain-dir] [domain-name]
```

Description Verifies the content of the `domain.xml` file. This subcommand is supported in local mode only.

Options

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

`--domaindir`
Specifies the directory where the domains are located. The path must be accessible in the file system. The default is the value of the `$AS_DEF_DOMAINS_PATH` environment variable. This variable is defined in the file `asenv.bat` or `asenv.conf`. The default value of this variable 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 --verbose=true
All Tests Passed.
domain.xml is valid
```

Exit Status

0	command executed successfully
1	error in executing the command

See Also `asadmin(1M)`

Sun GlassFish Enterprise Server v3 Domain File Format Reference

Name version – displays the Enterprise Server version information

Synopsis

```
version [--help] [--verbose={false|true}]
```

Description Use the `version` subcommand to display the version information for Enterprise Server. If the subcommand cannot communicate with Enterprise Server by using the given user/password and host/port, then the subcommand will retrieve the version locally and display a warning message.

This subcommand is supported in remote mode and local mode.

Options

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

`--verbose`
If this flag is set to true, the version of the Java Runtime Environment (JRE) that the server runs is provided. Default is false.

Examples **EXAMPLE 1** Displaying Enterprise Server Version Information

```
asadmin> version
GlassFish v3  (build b59)
Command version executed successfully.
```

If the server cannot be reached, you might receive information similar to the following.

```
Version string could not be obtained from Server [localhost:4848] for some reason.
(Turn debugging on to see details).
Locally retrieved version string from version class
(com.sun.appserv.server.util.Version) is [GlassFish v3.0-b59  (build 59)]
Command version executed successfully.
```

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

See Also `list-modules(1)`

`asadmin(1M)`

REFERENCE

Sun GlassFish v3 Preview Enterprise Server Section 1M: Utility Commands

Name 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-name*`Client.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, Solaris Version (`http://java.sun.com/javase/6/docs/technotes/tools/solaris/classpath.html`) or Setting the Class Path, Windows Version (`http://java.sun.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.sun.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 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 user name of the authorized administrative user of the domain administration server.

`-passwordfile`
optional; specifies the name, including the full path, of a file that contains the 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 valid entries in the file are as follows:

`AS_ADMIN_PASSWORD=`*administration-password*
`AS_ADMIN_MAPPEDPASSWORD=`*mapped-password*
`AS_ADMIN_USERPASSWORD=`*user-password*
`AS_ADMIN_MASTERPASSWORD=`*master-password*
`AS_ADMIN_ALIASPASSWORD=`*alias-password*

The password can be specified by one of the following means:

- Through the `-passwordfile` option
- Interactively at the command prompt

For security reasons, a password that is specified as an environment variable is not read by the `appclient` utility.

If the `AS_ADMIN_PASSWORD` environment variable has been exported to the global environment, specifying the `-passwordfile` option produces a warning about using the `-password` option. To avoid this warning, upset the `AS_ADMIN_PASSWORD` environment variable.

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`.

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 class name of the main client application `main()` method to be invoked by the Application Client Container. For example, `com.sun.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, Solaris Version (`http://java.sun.com/javase/6/docs/technotes/tools/solaris/classpath.html`) or Setting the Class Path, Windows Version (`http://java.sun.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 -splash welcome.jpg -xml sun-acc.xml -jar myclientapp.jar scott sample
```

Where: `welcome.jpg` is a splash screen specified by the `-splash` JVM option, `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 `welcome.jpg`, `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:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Unstable

See Also

`get-client-stubs(1)`

`asadmin(1M)`, `package-appclient(1M)`

Name asadmin – utility for performing administrative tasks for Sun GlassFish Enterprise 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 Sun GlassFish Enterprise 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 '*'
    *
    ```

- **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)[1]

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.

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.

When used without single quote (') characters, the escape character disables the delimiter in the command-line interface. 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
  ```

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" in *Sun GlassFish Enterprise Server v3 Domain File Format Reference* element must be set to `true`.

  To set this attribute, use the `set(1)` subcommand. The `http-listener` element is stored in the `domain.xml` configuration file.

[1] The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform.

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 of a file that contains the 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_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)

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`.

`--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.

```
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`.

```
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.

EXAMPLE 3 Specifying an `asadmin` Utility Option and a Subcommand Option *(Continued)*

```
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:\tools\jruby`.

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 jruby.home='c\:\\tools\\jruby' bookstore
Application deployed successfully with name hello.


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 jruby.home='c:\tools\jruby' bookstore
Application deployed successfully with name hello.


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:

EXAMPLE 5 Specifying a UNIX Operating System Meta Character in an Option *(Continued)*

```
asadmin --user admin --passwordfile gfpass create-jdbc-connection-pool
--datasourceclassname sampleClassName
--description Test\Escape\Character
sampleJDBCConnectionPool
```

EXAMPLE 6 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(1)` 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 single quote (') characters:

```
asadmin --user admin --passwordfile gfpass create-jdbc-connection-pool
--datasourceclassname com.derby.jdbc.jdbcDataSource
--property 'user=dbnuser:passwordfile=dbpasswordfile:
DatabaseName="jdbc:derby":server="http://locahost:9092"'
```

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:
```

EXAMPLE 6 Specifying a Meta Character and an Option Delimiter Character in a Property *(Continued)*

```
DatabaseName=\"jdbc:derby\":server=\"http://localhost:9092\" javadb-pool
```

EXAMPLE 7 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:\sun\appserver` 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 8 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 (`\`):

EXAMPLE 8 Specifying an Option That Contains an Escape Character *(Continued)*

```
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.

Environment Variable	`asadmin` Utility Option
AS_ADMIN_TERSE	--terse
AS_ADMIN_ECHO	--echo
AS_ADMIN_INTERACTIVE	--interactive
AS_ADMIN_HOST	--host
AS_ADMIN_PORT	--port
AS_ADMIN_SECURE	--secure
AS_ADMIN_USER	--user
AS_ADMIN_PASSWORDFILE	--passwordfile
AS_ADMIN_HELP	--help

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Unstable

See Also

create-connector-security-map(1), create-file-user(1), create-jdbc-connection-pool(1), create-jvm-options(1), create-password-alias(1),deploy(1), list-applications(1), list-commands(1), login(1), restart-domain(1), set(1), set-web-env-entry(1), start-domain(1), update-file-user(1)

attributes(5)

"http-listener" in *Sun GlassFish Enterprise Server v3 Domain File Format Reference*

Name 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 current working directory. 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

After unjarring on the client machine, modify *appclient_install_dir*`/config/asenv.conf` (`asenv.bat` for Windows) as follows:

- set `AS_WEBSERVICES_LIB` to *appclient_install_dir*`/lib`
- set `AS_NSS` to *appclient_install_dir*`/lib` (*appclient_install_dir*`\bin` for Windows)
- set `AS_IMQ_LIB` to *appclient_install_dir*`/imq/lib`
- set `AS_INSTALL` to *appclient_install_dir*
- set `AS_JAVA` to your JDK 1.6 home directory
- set `AS_ACC_CONFIG` to *appclient_install_dir*`/config/sun-acc.xml`

Modify *appclient_install_dir*`/config/sun-acc.xml` as follows:

- Ensure the `DOCTYPE` file references *appclient_install_dir*`/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 stubs for your target application, for example, *yourClientStub.jar*.
- Execute the `appclient` utility: `appclient` *-client yourClientStub.jar*

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Unstable

See Also appclient(1M)

REFERENCE

Sun GlassFish v3 Preview Enterprise Server Section 5ASC: Enterprise Server Concepts

Name 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 Enterprise 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.

Name configuration – the data set that determines how Enterprise Server operates

Description The *configuration* of Enterprise Server is the data set that determines how it operates. Parts of this configuration determine the operation of specific parts of Enterprise Server, such as the following:

- Services, such as the transaction service
- Resources, such as databases
- Deployed applications or modules, such as web applications

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.

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 Enterprise 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.

Most of the Enterprise Server configuration is stored in the `domain.xml` file. For full details about the structure of this file, see the *Sun GlassFish Enterprise Server v3 Domain File Format Reference*.

See Also `get(1)`, `list(1)`, `set(1)`

`asadmin(1M)`

`dotted-names(5ASC)`

"Configuration Tasks" in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name 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)

Name dotted-names – syntax for using periods to separate name elements

Description A *dotted name* is an identifier for a particular Enterprise 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.

  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 `--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
    ```

Examples

EXAMPLE 1 Listing All Configurable Elements

This example lists all the configurable elements.

```
asadmin> list *
```

Output similar to the following is displayed:

```
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.jmx-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.iiop-service
```

EXAMPLE 1 Listing All Configurable Elements *(Continued)*

```
configs.config.server-config.iiop-service.iiop-listener.SSL
configs.config.server-config.iiop-service.iiop-listener.SSL.ssl
configs.config.server-config.iiop-service.iiop-listener.SSL_MUTUALAUTH
configs.config.server-config.iiop-service.iiop-listener.SSL_MUTUALAUTH.ssl
configs.config.server-config.iiop-service.iiop-listener.orb-listener-1
configs.config.server-config.iiop-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
configs.config.server-config.mdb-container
configs.config.server-config.monitoring-service
configs.config.server-config.monitoring-service.module-monitoring-levels
...
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
servers.server.server.resource-ref.jdbc/__default
system-applications
Command list executed successfully.
```

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
```

EXAMPLE 2 Listing All the Monitorable Objects *(Continued)*

```
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)

Name instance – an Enterprise Server instance has its own Java EE configuration, Java EE resources, application deployment areas, and server configuration settings.

Description The Enterprise Server creates one application 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 application server instance meets their needs. However, depending upon your environment, you might want to create additional application server instances. For example, in a development environment you can use different application server instances to test different Enterprise Server configurations, or to compare and test different application deployments. Because you can easily add or delete an application server instance, you can use them to create temporary "sandbox" areas to experiment with while developing.

Name logging – capturing information on Enterprise Server runtime events

Description *Logging* is the process by which Enterprise Server captures data about events that occur during Enterprise Server operation. Enterprise 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 Enterprise 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-level` 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-logger-levels(1)`, rotate-log(1), `set-log-level(1)`

`asadmin(1M)`

Chapter 7, "Administering the Logging Service," in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name monitoring – reviewing the runtime state of components and services deployed in Enterprise 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 Enterprise 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 Enterprise 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 Enterprise 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)`

Chapter 8, "Administering the Monitoring Service," in *Sun GlassFish Enterprise Server v3 Administration Guide*

Name passwords – securing and managing application server

Description An Enterprise Server™ administrator 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)`

Name resources – Provide connectivity to various types of EIS .

Description Enterprise Server provides support for JDBC, JMS, and JNDI resources.

See Also asadmin(1M)

Name security – secure and administer application server applications

Description Security is about protecting data: how to prevent unauthorized access or damage to it in storage or transit. The Enterprise 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. The Enterprise 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 `change-admin-password(1)`, `change-master-password(1)`, `create-auth-realm(1)`, `create-file-user(1)`, `create-message-security-provider(1)`, `create-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)`

Index

D

E

F

G

I

J

L

M

P

R

S

T

U

V