Resynchronizing GlassFish Server Instances and the DAS

Configuration data for a GlassFish Server instance is stored as follows:

- In the repository of the domain administration server (DAS)
- In a cache on the host that is local to the instance

The configuration data in these locations must be synchronized. The cache is synchronized only when a user uses the administration tools to start or restart an instance.

Default Synchronization for Files and Directories

The `--sync` option of the subcommands for starting an instance controls the type of synchronization between the DAS and the instance's files when the instance is started. You can use this option to override the default synchronization behavior for the files and directories of an instance. For more information, see the following subcommand help pages:

- `start-instance(1)`
- `start-local-instance(1)`

The default synchronization behavior for the files and directories of an instance is as follows:

`applications`
: This directory contains a subdirectory for each application that is deployed to the instance.

  By default, only a change to a top-level application subdirectory causes the DAS to synchronize the `applications` directory. When the DAS resynchronizes the `applications` directory, all the application's files and all generated content that is related to the application are copied to the instance.

  If a file below a top-level subdirectory is changed without a change to a file in the top-level subdirectory, full synchronization is required. In normal operation, files below the top-level subdirectories of these directories are not changed. If an application is deployed and undeployed, full synchronization is not necessary to update the instance with the change.

`config`
: This directory contains configuration files for the entire domain.

  By default, the DAS resynchronizes files that have been modified since the last resynchronization only if the `domain.xml` file in this directory has been modified.

`config/`*config-name*
: This directory contains files that are to be shared by all instances that reference the named configuration *config-name*. A *config-name* directory exists for each named configuration in the configuration of the DAS.

Because the *config-name* directory contains the subdirectories `lib` and `docroot`, this directory might be very large. Therefore, by default, only a change to a file or a top-level subdirectory of *config-name* causes the DAS to resynchronize the *config-name* directory.

`config/domain.xml`
This file contains the DAS configuration for the domain to which the instance belongs.

By default, the DAS resynchronizes this file if it has been modified since the last resynchronization.

Note – A change to the `config/domain.xml` file is required to cause the DAS to resynchronize files in the `config` directory. If the `config/domain.xml` file has not changed since the last resynchronization, no files in the `config` directory are resynchronized, even if some of these files are out of date in the cache.

`docroot`
This directory is the HTTP document root directory. By default, all instances in a domain use the same document root directory. To enable instances to use a different document root directory, a virtual server must be created in which the `docroot` property is set. For more information, see the `create-virtual-server(1)` help page.

The `docroot` directory might be very large. Therefore, by default, only a change to a file or a top-level subdirectory of the `docroot` directory causes the DAS to resynchronize the `docroot` directory. By checking files in the `docroot` directory directory, the DAS can detect changes to the `index.html` file.

When the DAS resynchronizes the `docroot` directory, all modified files and subdirectories at any level are copied to the instance.

If a file below a top-level subdirectory is changed without a change to a file in the top-level subdirectory, full synchronization is required. In normal operation, files below the top-level subdirectories of these directories are not changed. If an application is deployed and undeployed, full synchronization is not necessary to update the instance with the change.

`generated`
This directory contains generated files for Java EE applications and modules, for example, EJB stubs, compiled JSP classes, and security policy files.

This directory is resynchronized when the `applications` directory is resynchronized. Therefore, only directories for applications that are deployed to the instance are resynchronized.

`java-web-start`
Only files and directories for applications that are deployed to the instance are resynchronized.

`lib`
`lib/classes`
: These directories contain common Java class files or JAR archives and ZIP archives for use by applications that are deployed to the entire domain. Typically, these directories contain common JDBC drivers and other utility libraries that are shared by all applications in the domain.

  The contents of these directories are loaded by the common class loader. For more information, see "Using the Common Class Loader" in *GlassFish Server Open Source Edition 3.1 Application Development Guide*. The class loader loads the contents of these directories in the following order:

  1. `lib/classes`
  2. `lib/*.jar`
  3. `lib/*.zip`

  These directories typically contain only a small number of files. Therefore, by default, a change to any file in these directories causes the DAS to resynchronize the directory that contains the file.

`lib/applibs`
: This directory contains application-specific Java class files or JAR archives and ZIP archives for use by applications that are deployed to the entire domain.

`lib/ext`
: This directory contains optional packages in JAR archives and ZIP archives for use by applications that are deployed to the entire domain. These archive files are loaded by using Java extension mechanism. For more information, see *Optional Packages - An Overview*.

Note – Optional packages were formerly known as *standard extensions* or *extensions*.

▼ To Resynchronize an Instance and the DAS Online

Resynchronizing an instance and the DAS updates the instance with changes to the instance's configuration files on the DAS. An instance is resynchronized with the DAS when the instance is started or restarted.

Tip – If default resynchronization is sufficient, you can restart the instance in a single operation as explained in the following sections:

- "To Restart an Individual Instance Centrally" on page 55
- "To Restart an Individual Instance Locally" on page 63

---

1 Ensure that the DAS is running.

2 **Determine whether the instance requires resynchronization with the DAS.**

```
asadmin> list-instances instance-name
```

instance-name
The name of the instance that you are resynchronizing with the DAS.

3 **Stop the instance.**

```
asadmin> stop-instance instance-name
```

instance-name
The name of the instance that you are resynchronizing with the DAS.

4 **Start the instance.**

- **If SSH is set up, start the instance centrally.**

  If you require full synchronization, set the `--sync` option of the start-local-instance command to full. If default synchronization is sufficient, omit this option.

  ```
  asadmin> start-instance [--sync full] instance-name
  ```

  Note – Only the options that are required to complete this task are provided in this step. For information about all the options for controlling the behavior of the instance, see the `start-instance(1)` help page.

  instance-name
  The name of the instance that you are starting.

- **If SSH is not set up, start the instance locally from the host where the instance resides.**

  If you require full synchronization, set the `--sync` option of the start-local-instance command to full. If default synchronization is sufficient, omit this option.

  ```
  $ asadmin start-local-instance [--node node-name] [--sync full] instance-name
  ```

  Note – Only the options that are required to complete this task are provided in this step. For information about all the options for controlling the behavior of the instance, see the `start-local-instance(1)` help page.

  node-name
  The node on which the instance resides. If only one node is defined in the domain, you may omit this option.

  instance-name
  The name of the instance that you are starting.

Example 4–15 Resynchronizing an Instance and the DAS Online

This example determines that the instance `yml-i1` requires resynchronization and fully resynchronizes the instance with the DAS. Because SSH is not set up, the instance is restarted locally on the host where the instance resides.

```
asadmin> list-instances yml-i1
yml-i1   running;  requires restart
Command list-instances executed successfully.
asadmin> stop-instance yml-i1
The instance, yml-i1, is stopped.
Command stop-instance executed successfully.
$ asadmin start-local-instance --node sj01 --sync full yml-i1
Removing all cached state for instance yml-i1.
Waiting for yml-i1 to start ................
Successfully started the instance: yml-i1
instance Location: /export/glassfish3/glassfish/nodes/sj01/yml-i1
Log File: /export/glassfish3/glassfish/nodes/sj01/yml-i1/logs/server.log
Admin Port: 24849
Command start-local-instance executed successfully.
```

See Also

- "To Restart an Individual Instance Centrally" on page 55
- "To Restart an Individual Instance Locally" on page 63
- `list-instances`(1)
- `start-instance`(1)
- `start-local-instance`(1)
- `stop-instance`(1)

You can also view the full syntax and options of the subcommands by typing the following commands at the command line.

```
asadmin help list-instances

asadmin help start-instance

asadmin help start-local-instance

asadmin help stop-instance
```

▼ To Resynchronize Library Files

To ensure that library files are resynchronized correctly, you must ensure that each library file is placed in the correct directory for the type of file. Some types of library file must be added to the class path.

1 Place each library file in the correct location for the type of library file as shown in the following table.

Type of Library Files	Location
Common JAR archives and ZIP archives for all applications in a domain.	*domain-dir*`/lib`
Common Java class files for a domain for all applications in a domain.	*domain-dir*`/lib/classes`
Application-specific libraries.	*domain-dir*`/lib/applibs`
Optional packages for all applications in a domain.	*domain-dir*`/lib/ext`
Library files for all applications that are deployed to a cluster or a standalone instance.	*domain-dir*`/config/`*config-name*`/lib`
Optional packages for all applications that are deployed to a cluster or a standalone instance.	*domain-dir*`/config/`*config-name*`/lib/ext`

domain-dir
The directory in which the domain's configuration is stored.

config-name
For a standalone instance: the named configuration that the instance references.

For a clustered instance: the named configuration that the cluster to which the instance belongs references.

2 For library files for all applications that are deployed to a cluster or a standalone instance, add the files to the class path prefix or class path suffix of the named configuration that the cluster or standalone instance references.

This type of library file is stored in the *domain-dir*`/config/`*config-name*`/lib` directory.

For other types of library file, omit this step. Other types of library file are automatically added to the class path by their class loaders.

- **To add the files to the class path prefix, perform these steps:**

  a. Determine the existing class path prefix.

  ```
  asadmin> get configs.config.config-name.java-config.classpath-prefix
  ```

  config-name The named configuration that the cluster or standalone instance references.

b. Set the class path prefix to the existing class path prefix and the files that you are adding.

```
asadmin> set
configs.config.config-name.java-config.classpath-prefix=library-file-list
```

config-name
The named configuration that the cluster or standalone instance references.

library-file-list
A comma-separated list of the files in the existing class path prefix and the files that you are adding.

- **To add the files to the class path suffix, perform these steps:**

a. Determine the existing class path suffix.

```
asadmin> get configs.config.config-name.java-config.classpath-suffix
```

config-name The named configuration that the cluster or standalone instance references.

b. Set the class path suffix to the existing class path suffix and the files that you are adding.

```
asadmin> set
configs.config.config-name.java-config.classpath-suffix=library-file-list
```

config-name
The named configuration that the cluster or standalone instance references.

library-file-list
A comma-separated list of the files in the existing class path suffix and the files that you are adding.

Next Steps When you deploy an application that depends on these library files, use the `--libraries` option of the deploy subcommand to specify these dependencies. For library files in the *domain-dir*`/lib/applib`, you can specify only the JAR file name, for example:

```
asadmin deploy --libraries commons-coll.jar,X1.jar foo.ear
```

For other types of library file, the full path is required.

See Also

- `get(1)`
- `set(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line.

```
asadmin help get

asadmin help set
```

▼ To Resynchronize Custom Configuration Files for an Instance

Configuration files in the *domain-dir*`/config` that are resynchronized are resynchronized for the entire domain. If you create a custom configuration file for an instance, the custom file is resynchronized only for the instance.

1 Place the custom configuration file in the *domain-dir*`/config/`*config-name* directory.

domain-dir
The directory in which the domain's configuration is stored.

config-name
The named configuration that the instance references.

2 If the instance locates the file through an option of the Java application launcher, update the option.

a. Delete the option.

```
asadmin> delete-jvm-options --target instance-name
option-name=current-value
```

instance-name
The name of the instance for which the custom configuration file is created.

option-name
The name of the option for locating the file.

current-value
The current value of the option for locating the file.

b. Re-create the option that you deleted in the previous step.

```
asadmin> create-jvm-options --target instance-name
option-name=new-value
```

instance-name
The name of the instance for which the custom configuration file is created.

option-name
The name of the option for locating the file.

new-value
The new value of the option for locating the file.

Example 4–16 Updating the Option for Locating a Configuration File

This example updates the option for locating the `server.policy` file to specify a custom file for the instance `pmd`.

```
asadmin> delete-jvm-options --target pmd
-Djava.security.policy=${com.sun.aas.instanceRoot}/config/server.policy
Deleted 1 option(s)
Command delete-jvm-options executed successfully.
asadmin> create-jvm-options --target pmd
-Djava.security.policy=${com.sun.aas.instanceRoot}/config/pmd-config/server.policy
Created 1 option(s)
Command create-jvm-options executed successfully.
```

See Also

- create-jvm-options(1)
- delete-jvm-options(1)

You can also view the full syntax and options of the subcommands by typing the following commands at the command line.

```
asadmin help create-jvm-options

asadmin help delete-jvm-options
```

▼ To Resynchronize Users' Changes to Files

A change to the `config/domain.xml` file is required to cause the DAS to resynchronize files in the `config` directory. If other files in the `config` directory are changed without a change to the `config/domain.xml` file, instances are not resynchronized with these changes.

The following changes are examples of changes to the `config` directory without a change to the `config/domain.xml` file:

- Adding files to `lib` directory
- Adding certificates to the key store by using the `keytool` command

1 Change the last modified time of the `config/domain.xml` file.

Exactly how to change the last modified time depends on the operating system. For example, on UNIX and Linux systems, you can use the touch(1) command.

2 Resynchronize each instance in the domain with the DAS.

For instructions, see "To Resynchronize an Instance and the DAS Online" on page 66.

See Also

- "To Resynchronize an Instance and the DAS Online" on page 66
- touch(1)

▼ To Resynchronize Only Specific Configuration Files

If you do not require instances in a domain to be resynchronized with all configuration files for the domain, you can specify a list of files to resynchronize.

Caution – If you specify a list of files to resynchronize, you must specify all the files that the instances require. Any file in the instance's cache that is not in the list is deleted when the instance is resynchronized with the DAS.

- **In the `config` directory of the domain, create a plain text file that is named `config-files` that lists the files to resynchronize.**

  In the `config-files` file, list each file name on a separate line.

Example 4–17 `config-files` File

This example shows the content of a `config-files` file that specifies that only the following files are to be resynchronized:

- `admin-keyfile`
- `cacerts.jks`
- `default-web.xml`
- `domain.xml`
- `domain-passwords`
- `keyfile`
- `keystore.jks`
- `server.policy`
- `sun-acc.xml`
- `wss-server-config-1.0`
- `xml wss-server-config-2.0.xml`

```
admin-keyfile
cacerts.jks
default-web.xml
domain.xml
domain-passwords
keyfile
keystore.jks
server.policy
sun-acc.xml
wss-server-config-1.0.xml
wss-server-config-2.0.xml
```

▼ To Prevent Deletion of Application-Generated Files

When the DAS resynchronizes an instance's files, the DAS deletes from the instance's cache any files that not listed for resynchronization. If an application creates files in a directory that the DAS resynchronizes, these files are deleted when the DAS resynchronizes an instance with the DAS.

- **Put the files in a subdirectory under the domain directory that is not defined by GlassFish Server, for example, `/export/glassfish3/glassfish/domains/domain1/myapp/myfile`.**

▼ To Resynchronize an Instance and the DAS Offline

Resynchronizing an instance and the DAS offline updates the instance's cache without the need for the instance to be able to communicate with the DAS. Offline resynchronization is typically required for the following reasons:

- To reestablish the instance after an upgrade
- To synchronize the instance manually with the DAS when the instance cannot contact the DAS

1 Ensure that the DAS is running.

2 Export the configuration data that you are resynchronizing to an archive file.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for exporting the configuration data, see the `export-sync-bundle(1)` help page.

---

3 Log in to the host where the instance resides.

4 Import the configuration data that you are resynchronizing from the archive file that you created in the previous step.

Note – Only the options that are required to complete this task are provided in this step. For information about all the options for importing the configuration data, see the `import-sync-bundle(1)` help page.

```
$ asadmin --host das-host [--port admin-port]
import-sync-bundle [--node node-name] --instance instance-name archive-file
```

das-host
The name of the host where the DAS is running.

admin-port
The HTTP or HTTPS port on which the DAS listens for administration requests. If the DAS listens on the default port for administration requests, you may omit this option.

node-name
The node on which the instance resides. If you omit this option, the subcommand determines the node from the DAS configuration in the archive file.

instance-name
The name of the instance that you are resynchronizing.

archive-file
The name of the file, including the path, that contains the archive file to import.

Example 4–18 Resynchronizing an Instance and the DAS Offline

This example resynchronizes the clustered instance `yml-i1` and the DAS offline. The instance is a member of the cluster `ymlcluster`.

```
asadmin> export-sync-bundle --target=ymlcluster
Sync bundle: /export/glassfish3/glassfish/domains/domain1/sync/
ymlcluster-sync-bundle.zip
Command export-sync-bundle executed successfully.
$ asadmin --host dashost.example.com import-sync-bundle
--node sj01 --instance yml-i1
/net/dashost.example.com/export/glassfish3/glassfish/domains/domain1/
sync/ymlcluster-sync-bundle.zip
Command import-sync-bundle executed successfully.
```

See Also

- `export-sync-bundle(1)`
- `import-sync-bundle(1)`

You can also view the full syntax and options of the subcommands by typing the following commands at the command line.

```
asadmin help export-sync-bundle
```

```
asadmin help import-sync-bundle
```

Migrating EJB Timers

▼ To Migrate EJB Timers

If a GlassFish Server server instance stops or fails abnormally, it may be desirable to migrate the EJB timers defined for that stopped server instance to a another running server instance.

The EJB timers can be migrated from the stopped source instance to a specified target instance. If no target instance is specified, the DAS will attempt to find a suitable server instance or multiple server instances. A migration notification will then be sent to the selected target server instances.

Note the following restrictions:

- If the source is a standalone instance, then the target must also be a standalone instance.
- If the source instance is part of a cluster, then the target instance must also be part of that same cluster.
- It is not possible to migrate timers from a standalone instance to a clustered instance, or from one cluster to another cluster.
- All EJB timers defined for a given instance are migrated with this procedure. It is not possible to migrate individual timers.