Composed February 22, 2011

◆ ◆ ◆ CHAPTER 5

Managing Administrative Security

This chapter describes how to manage administrative security by using the secure administration feature.

This chapter assumes that you are familiar with security features such as authentication, authorization, and certificates. If you are not, first see Chapter 1, "Administering System Security."

Instructions for accomplishing the tasks specific to GlassFish Server by using the Administration Console are contained in the Administration Console online help.

Secure Administration Overview

The secure administration feature allows an administrator to secure all administrative communication between the domain administration server (DAS), any remote instances, and administration clients such as the `asadmin` utility, the administration console, and REST clients.

In addition, secure administration helps to prevent DAS-to-DAS and instance-to-instance traffic, and carefully restricts administration-client-to-instance traffic.

The secure administration feature, which is henceforth referred to as *secure admin*, provides a secure environment, in which you can be confident that rogue users or processes cannot intercept or corrupt administration traffic or impersonate legitimate GlassFish Server components.

When you install GlassFish Server or create a new domain, secure admin is disabled by default. When secure admin is disabled, GlassFish Server does not encrypt administrative communication among the system components and does not accept administrative connections from remote hosts.

This release adds two new subcommands to control secure administration settings:

- enable-secure-admin–The `enable-secure-admin` subcommand turns on secure admin. GlassFish Server uses SSL encryption to protect subsequent administrative traffic and will accept remote administrative connections. Enabling secure admin affects the entire domain, including the DAS and all instances. The DAS must be running, and not any instances, when you run `enable-secure-admin`. You must restart the DAS immediately after enabling secure admin, and then start any instances you want to run.
- disable-secure-admin–`disable-secure-admin` subcommand turns off secure admin. GlassFish Server no longer encrypts administrative messages and will no longer accept remote administration connections. Disabling secure admin affects the entire domain, including the DAS and all instances. The DAS must be running , and not any instances, when you run `disable-secure-admin`. You must restart the DAS immediately after disabling secure admin, and then start any instances you want to run.

  If secure admin is not enabled, this subcommand has no effect.

This section describes how to use these commands to run secure admin, and the implications of doing so.

How Secure Admin Works: The Big Picture

Secure admin is a domain-wide setting. It affects the DAS and all instances and all administration clients. This section describes the following topics:

Functions Performed by Secure Admin

The `enable-secure-admin` subcommand performs the following functions. Subsequent sections describe these functions in more detail.

- Enables the secure admin behavior, optionally setting which aliases are to be used for identifying the DAS and instance certificates.

- Adjusts all configurations in the domain, including default-config.
- Adjusts Grizzly settings:
  - SSL/TLS is enabled in the DAS's admin listener and the instances' admin listeners.
  - Port unification (that is, HTTP and HTTPS are handled by the same port), http—to—https redirection, and client authentication (client-auth=want) are enabled.
  - Configures SSL to use the administration truststore.
  - Configures SSL to use the administration keystore and the correct alias (for the self-signed cert) for authenticating itself. (You can use your own certificate instead, as described in "Using Your Own Certificates" on page 90.

    The Grizzly configuration on the DAS and each instance is identical, with the exception that the DAS uses the `s1as` alias for SSL/TLS authentication and the instances use the `glassfish-instance` alias. (These alias names are the default, and you can change them.)

  A server restart is required to change the Grizzly adapter behavior.

  The restart also synchronizes the restarted instances. When you start the instances, the DAS delivers the updated configuration (in domain.xml) to the instances.

Which Administration Account is Used?

If only one administration account exists in the realm, GlassFish Server treats that account as the current default administration account. In this case, when you run an `asadmin` command, you do not need to specify the username. If a password for that username is required, you need to specify it, typically by using the `--passwordfile` option or by letting `asadmin` prompt you for it.

By default, GlassFish Server includes a single account for user "admin" and an empty password. Therefore, if you make no other changes before you enable secure admin, "admin" is the initial default username and no password is required. You need to decide whether enabling secure admin without also requiring a password makes sense in your environment.

If multiple admin accounts exist, then GlassFish Server does not recognize any admin username as the default. You must then specify a valid username via the `-–user` option when you use the `asadmin` command (or by or defining the `AS_ASDMIN_USER` environment variable), and its associated password (if the associated password is not empty).

The username and password used for a login attempt must match the username and password (if required) for an account defined in the realm, and you must have set up the account as a member of the admin group.

What Authentication Methods Are Used for Secure Administration?

The secure admin feature enforces security via the following authentication methods:

- The DAS and instances authenticate to each other via mutual (two-way) SSL/TLS certificate authentication. The DAS authenticates to clients via one-way SSL/TLS certificate authentication.

  The domain creation process creates a default keystore and truststore, plus a default private key for the DAS. Secure admin uses this initial configuration to set up the truststore so that the DAS and instances always trust each other.
- Remote administration clients (`asadmin`, administration console, browsers, and IDEs) must accept the public certificate presented by the DAS. If accepted, remote administration clients then send a user name and password (HTTP Basic authentication) in the HTTP Authorization header. The receiving DAS or instance makes sure those credentials are valid in its realm, and authenticates and authorizes the user.
- A locally-running `asadmin` (that is, connecting to an instance on the same host) authenticates and authorizes to the co-located instance using a locally-provisioned password.
- Credentials or other sensitive information sent over the network are always encrypted if secure admin is enabled. No credentials are sent in the clear if secure admin is enabled. (If secure admin is disabled, credentials *are* sent in the clear.) Messages between administration clients and the DAS, between the DAS and remote instances, and between local administration clients and instances are encrypted using SSL/TLS. This is true even if you explicitly set the `asadmin --secure` option to false.

Table 5–1 shows which authentication methods are employed when secure admin is enabled or disabled.

TABLE 5–1 Authentication Methods Employed

Access Method	When Secure Admin is Disabled	When Secure Admin is Enabled
Remote administration access to the DAS	Rejected.	Username/password authentication. (Client must also accept server certificate.)
Communication between DAS and instances	Cleartext messages. No mutual authentication.	SSL-encrypted messages. SSL mutual authentication using certificates.
Communication between administration clients and DAS	Cleartext messages. No DAS authentication.	SSL-encrypted messages. DAS uses SSL certificate server authentication.
Local asadmin client to instance on same node	Cleartext messages. Locally-provisioned password mechanism is used.	SSL-encrypted messages. Locally-provisioned password mechanism is used.

Understanding How Certificate Authentication is Performed

The domain creation process creates a primary (private) key and a self-signed certificate for the DAS, and a separate private key and self-signed certificate for remote instances.

Then, when you enable secure admin, the following actions are performed:

- Both private keys are stored in the domain-wide DAS keystore file, keystore.jks.
- Both public certificates are stored in the domain-wide DAS truststore file, cacerts.jks.

When the DAS sends a message to an instance:

1. SSL on the instance asks the DAS to provide an SSL/TLS certificate.
2. The DAS sends the certificate with the alias you specified using the `--adminalias` option when you ran the `enable-secure-admin` subcommand.
3. SSL on the instance makes sure the certificate is valid and GlassFish Server makes sure that the security Principal associated with the incoming request (provided automatically by Grizzly and the SSL/TLS Java implementation) matches the Principal associated with the adminalias from the instance's truststore.

What Certificates Are Used?

When you enable secure admin, you can optionally set the `--adminalias` and `--instancealias` options that tell secure admin which aliases to use for the DAS and instance certificates.

The DAS uses the alias associated with the `--instancealias` option to check incoming requests that use SSL/TLS cert authentication. Conversely, instances use the alias associated with the `--adminalias` option to check incoming requests with certificate authentication.

By default, `--adminalias` of the `enable-secure-admin` subcommand uses the `s1as` alias, and the `--instancealias` option uses the `glassfish-instance` alias, both of which identify the default self-signed certificates.

You can use your tool of choice, such as keytool, to list the default self-signed certificates in the keystore, similar to the following:

Note – You can list the contents of the keystore without supplying a password. However, for a request that affects the private key, such as the keytool.exe `--certreq` option, the keystore password is required. This is the master password and has a default value of *changeit* unless you change it with the `change-master-password` subcommand.

```
keytool.exe -list -keystore keystore.jks
```

```
Enter keystore password:

*****************  WARNING WARNING WARNING  *****************
* The integrity of the information stored in your keystore  *
* has NOT been verified!  In order to verify its integrity, *
* you must provide your keystore password.                  *
*****************  WARNING WARNING WARNING  *****************

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries

glassfish-instance, Jan 3, 2011, PrivateKeyEntry,
Certificate fingerprint (MD5): 06:A4:83:84:57:52:9C:2F:E1:FD:08:68:BB:2D:ED:E8
s1as, Jan 3, 2011, PrivateKeyEntry,
Certificate fingerprint (MD5): 8B:7D:5A:4A:32:36:1B:5D:6A:29:66:01:B0:A3:CB:85
```

The `--adminalias` and `--instancealias` values are maintained in the domain.xml file. Because of this design, normal instance creation operations (create-instance over SSH and create-local-instance) apply the up-to-date keystore, truststore, and configuration to each instance.

Self-Signed Certificates and Trust

The self-signed certificates that GlassFish Server uses might not be trusted by clients by default because a certificate authority does not vouch for the authenticity of the certificate. If you enable secure admin and then contact the DAS using an administration client, that client will detect whether the certificate is automatically trusted.

Browsers will warn you, let you view the certificate, and ask you to reject the certificate, accept it once, or accept it indefinitely, as shown in Figure 5–1.

FIGURE 5–1 Sample Browser Response to Untrusted Certificate

Similarly, the first time `asadmin` receives an untrusted certificate, it displays the certificate and lets you accept it or reject it, as follows: (If you accept it, `asadmin` also accepts that certificate in the future.)

```
D:\glassfish3\glassfish\bin>asadmin enable-secure-admin
Command enable-secure-admin executed successfully.


D:\glassfish3\glassfish\bin>asadmin stop-domain domain1
Waiting for the domain to stop .......
Command stop-domain executed successfully.

D:\glassfish3\glassfish\bin>asadmin start-domain domain1
Waiting for domain1 to start ............................
Successfully started the domain : domain1
domain  Location: D:\glassfish3\glassfish\domains\domain1
Log File: D:\glassfish3\glassfish\domains\domain1\logs\server.log
Admin Port: 4848
Command start-domain executed successfully.

D:\glassfish3\glassfish\bin>asadmin list-domains
[
[
  Version: V3
  Subject: CN=machine.oracle.com, OU=GlassFish, O=Oracle Corporation, L=San
ta Clara, ST=California, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus: 91604359507378444963235875637429733088161806229854910107270225245885674079656358328568800001548507219262910864311924824938195045822088563459253216383211006608196572047575238964156068334714995640712267224780564071023188627967974656245090519956376357288295037519504394674686082145398885236913866246525691704749
  public exponent: 65537
```

```
  Validity: [From: Tue Jan 04 14:30:08 EST 2011,
              To: Fri Jan 01 14:30:08 EST 2021]
  Issuer: CN=machine.oracle.com, OU=GlassFish, O=Oracle Corporation, L=Sant
a Clara, ST=California, C=US
  SerialNumber: [    4d237540]

Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: AF 8B 90 1E 51 9A 80 1B   EB A4 D9 C6 01 8A A0 FD  ....Q...........
0010: DE EC 83 8A                                        ....
]
]

]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 3F 2B 30 CE 97 0B 5E F3   72 0E 60 18 8D 3B 04 DC  ?+0...^.r.`..;..
0010: 26 E6 7A 6F D0 19 CC 26   1D 90 C0 DE 33 4E 53 FB  &.zo...&....3NS.
0020: DC E7 AE 78 9E BA EF 14   86 57 36 D4 3E 9B C9 FB  ...x.....W6.>...
0030: C0 B4 EF 72 27 D9 4F 79   1F 89 91 B8 96 26 33 64  ...r'.Oy.....&3d
0040: 9F 4B 04 4B 83 B9 BF 4D   54 B4 8F 75 17 1A 51 BD  .K.K...MT..u..Q.
0050: F3 69 94 CE 90 95 08 55   2C 07 D2 23 AC AE EC 6D  .i.....U,..#...m
0060: 84 B6 3D 00 FB FE 92 50   37 1A 2D 00 F1 21 5C E6  ..=....P7.-..!\.
0070: 1F 39 26 B2 5D C1 FD C8   B1 4F CC EE 26 84 B8 B5  .9&.]....O..&...

]
Do you trust the above certificate [y|N] -->
```

`asadmin` saves certificates you accept in the file `.asadmintruststore` in your log-in default directory. You do not generally need to work with the file directly, but if you delete or move the file, `asadmin` will prompt you again when it receives untrusted certificates.

Some `asadmin` commands such as run-script can contact an instance directly to retrieve information (but not to make configuration changes). The instances do not use the same certificate as the DAS, so in these cases `asadmin` then prompts you to accept or reject the instance certificate.

Using Your Own Certificates

By default, `--adminalias` of the `enable-secure-admin` subcommand uses the `s1as` alias, and the `--instancealias` option uses the `glassfish-instance` alias, both of which identify the default self-signed certificates.

You can instead have GlassFish Server use your own certificates for this purpose by first adding your certificates to the keystore and truststore, and then running `enable-secure-admin` and specifying the aliases for your certificates.

It is also possible to use `s1as` and `glassfish-instance` as the alias names for your own certificates. A benefit of doing so is that you would not have to specify alias names with the `enable-secure-admin` subcommand.

In addition, your own certificate identified by the `s1as` alias would be used in all other cases within the domain where the `s1as` alias is used (by default), such as in the SSL configuration of the IIOP and http-listener-2 listeners, and as the `encryption.key.alias` and `signature.key.alias` used for provider configuration in the SOAP authentication layer for Message Security configuration.

You may find the wide-reaching effect of using the `s1as` alias with your own certificate to be either a useful feature or an unintended consequence. Therefore, you should understand the implications of using the `s1as` alias before doing so.

If you decide to use the `s1as` and `glassfish-instance` aliases with your own certificates, you will first need to disable secure admin (if enabled) and then change or delete the exiting `s1as` alias from both the `keystore.jks` keystore and `cacerts.jks` truststore for the DAS. You can use the `--changealias` or`--delete` option of `keytool` to accomplish this. Then, import your own certificates.

When you enable secure admin, the DAS and the instances then have copies of the same keystore and truststore

Guarding Against Unwanted Connections

Secure admin guards against unwanted connections in several ways:

- DAS-to-DAS, instance-to-instance:
  - The DAS and the instances have copies of the same truststore, which contains the public certificate of the DAS and the separate public certificate that is used by all instances.
  - DAS-to-other-DAS communication is not authenticated because each different DAS will have its own self-signed certificate that is not in the truststore of the other DAS.
  - DAS-to-itself communication is unlikely unless you were to misconfigure the admin listener port for an instance on the same host so it is the same as for the DAS. Similarly, instance-to-instance traffic is unlikely unless you were to misconfigure listener ports for instances on the same host.

    To prevent both of these situations, both cases are handled by making sure that the connecting Principal (alias) is not the running Principal. secure admin ensures that if the client has authenticated using SSL/TLS client authentication that the Principal associated with the remote client is not the same as the current process. That is, the DAS makes sure that the Principal is not itself. Similarly, each instance ensures that the client is not an instance. (The instances share the same self-signed certificate and therefore are mapped to the same Principal.)
- Remote client-to-instance:

Remote `asadmin` clients are unable to connect directly to instances. If the user on host "test1" runs a local command but specifies a remote instance on host "test2," `asadmin` on test1 will read and send that locally-provisioned password. The instance on "test2" will have a different locally-provisioned password and so the authentication attempt will fail.

Therefore, a user on "test1" will not be able to run a remote command targeting an instance on "test2."

Considerations When Running GlassFish Server With Default Security

In GlassFish Server, the default admin account is username "admin" with an empty password. Admin clients provide empty credentials or none at all, and all are authenticated and authorized as that default admin user. None of the participants (clients, DAS, or instances) encrypts network messages.

If this level of security is acceptable in your environment, no changes are needed and you do not need to enable secure administration. Imposing a heightened level of security is optional.

However, consider Table 5–2, which shows which operations are accepted and rejected when secure admin is disabled.

Note – When secure admin is disabled, GlassFish Server does allow remote monitoring (read-only) access via the REST interface.

TABLE 5–2 Accepted and Rejected Operations if Secure Admin is Disabled

Operation	Run From Same System as DAS	Run From Remote System
`start-local-instance`	Functions as expected	Cannot sync with DAS. The instance starts but cannot communicate with the DAS. DAS will not see the instance.
Any other `asadmin` subcommand	Functions as expected	Rejected. A user sees the username/password prompt, but even correct entries are rejected.
Commands that use SSH. For example, create-instance.	Functions as expected; requires prior SSH configuration.	Functions as expected; requires prior SSH configuration.

Running Secure Admin

This section describes how to run secure admin. The section begins with prerequisites for running secure admin.

Prerequisites for Running Secure Admin

Before running GlassFish Server with secure admin enabled, you must make sure that:

1. The DAS is installed, initialized, and running.
2. If one or more remote instances are installed and initialized, they must *not* be running.
3. Any administration clients you require are installed.
4. The DAS communicates on the `--adminport` you configure when you create the domain, and defaults to 4848. An instance communicates on the `ASADMIN_LISTENER_PORT` system property you specify for the instance.
5. The user name and password sent by remote administration clients (`asadmin`, administration console, browsers, and IDEs) must exist in the realm and be in the admin group.
6. The keystore and truststore for the domain exist. (They are created by default when you create the domain or install GlassFish Server.)

   If you are not using the default self-signed certificates, you must add your own valid certificates and CA root in the keystore and truststore, respectively.
7. If you are not using the default self-signed certificates, create two aliases corresponding to certificates in the keystore and truststore: one that the DAS will use for authenticating itself in administration traffic, and one that the instances will use for authenticating itself in administration traffic.

Example of Running enable-secure-admin

The following example shows how to enable secure admin for a domain using the default admin alias and the default instance alias. You must restart the DAS immediately after enabling secure admin.

Note – The only indicator that secure admin is enabled is the successful status from the `enable-secure-admin` subcommand. When secure admin is running, the DAS and instances do not report the secure admin status.

```
asadmin> enable-secure-admin

Command enable-secure-admin executed successfully.
```

The following example shows how to enable secure admin for a domain using an admin alias `adtest` and an instance alias `intest`. You can also use this command to modify an existing secure admin configuration to use different aliases.

```
asadmin> enable-secure-admin --adminalias adtest --instancealias intest
```

The following example shows how to disable secure admin:

```
asadmin> disable-secure-admin

Command disable-secure-admin executed successfully.
```

You can use the following command to see the current state of secure admin in a domain:

```
asadmin> get secure-admin.enabled

secure-admin.enabled=false

Command get executed successfully.
```

Additional Considerations When Creating Local Instances

If you use `xxx-local-instance` commands to set up local instances, either leave secure admin disabled, or enable it before you create or start the instances and leave it that way.

However, if you use `xxx-instance` commands over SSH to manage remote instances, you can enable and disable secure admin, although this is not recommended because it can result in an inconsistent security model.

Secure Admin Use Case

This section describes a simple secure admin use case.

In the **`asadmin --secure=false --user me --passwordfile myFile.txt cmd ...`** use case, the user submits a command with `--secure` set to false, and supplies password credentials.

The important concept to note is that `asadmin` uses HTTPS because of the DAS redirection, even though the command sets `--secure` to false. `asadmin` sends the HTTP Authorization header along with the redirected request.

In addition to the flow described here, certificate authentication is also performed as described in Table 5–3. Also, the credentials that the user supplies are assumed to be valid administrator credentials for the DAS.

TABLE 5–3 asadmin --secure=false, With Username and Password

asadmin	Grizzly	AdminAdapter
Sends HTTP request, no authorization header (because the transport is not secure).		
	Returns 3xx status and redirects HTTP to HTTPS.	
Follows redirection, this time adding the Authorization header (because transport is now HTTPS).		
		Authenticates admin user and password from HTTP Authorization header in the realm Executes command, and responds with success status.

Upgrading an SSL-Enabled Secure GlassFish Installation to Secure Admin

If you enable secure admin on an SSL-enabled GlassFish Server installation, secure admin uses the existing <ssl cert-nickname> value as the DAS adminalias for secure admin.

Composed February 22, 2011