User Guide

/ Amster 5.5.2

Latest update: 5.5.2

ForgeRock AS
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Guide to the ForgeRock® Access Management command-line interface, Amster.

Table of Contents

Preface

This guide shows you how to install Amster, and how to integrate with ForgeRock Access Management. Read the Release Notes before you get started.

This guide is written for anyone installing Amster to interface with supported ForgeRock Access Management deployments.

About ForgeRock Identity Platform™ Software

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

Chapter 1

Introducing the Amster Command-line Interface

Amster is a command-line interface built upon the ForgeRock Access Management REST interface. Use Amster in DevOps processes, such as continuous integration, command-line installations, and scripted cloud deployments.

Amster provides the following features:

- **Remote, Scripted Deployments**. Script AM deployments by using the Groovy scripting support within Amster.

  For more information, see "Scripting" and "Installing Access Management with Amster".

- **AM Configuration Import and Export**. Amster can export all the configuration related to an AM instance, and import it back to the same, or a different instance.

  Note that Amster only manages configuration data. User information in data stores is not imported or exported, or modified in any way.

  For more information, see "Exporting Configuration Data" and "Importing Configuration Data".

- **Configuration Stored in JSON**. Amster exports configuration to a hierarchy of JSON format text files on the local filesystem.

  Global defaults and configuration are exported to the `global` folder, and the configuration for realms is exported into subfolders of the `realms` folder.

  The following is a simplified example of an exported hierarchy, including the top-level `root` realm:

```
|-- global
|   |-- ActiveDirectoryModule.json
|   |-- GlobalScripts
|   |   |-- 157298c0-7d31-4059-a95b-eeb08473b7e5.json
|   |   `-- 36863ffb-40ec-48b9-94b1-9a99f71cc3b5.json
|   |-- HotpModule.json
|   |-- Realms
|   |   `-- root.json
|   |-- Servers
|   |   `-- 01
|   |       |-- CtsDataStoreProperties.json
|   |       |-- SessionProperties.json
|   |       `-- 01.json
|   `-- Session.json
`-- realms
    `-- root
        |-- AmsterModule
        |   `-- amster.json
        |-- AuthenticationChains
        |   |-- amsterService.json
        |   `-- myScriptedChain.json
        |-- DataStoreModule
        |   `-- datastore.json
        |-- ScriptedModule
        |   `-- myScriptedAuthModule.json
        `-- Scripts
            |-- 9de3eb62-f131-4fac-a294-7bd170fd4acb.json
            `-- c827d2b4-3608-4693-868e-bbcf86bd87c7.json
```

Store these files in a version control system to manage and maintain AM configurations.

For a list of the available entities, see the Entity Reference.

- **Encryption of Sensitive Data**. Amster can encrypt exported password and sensitive data in the configuration files that are stored on disk. Only a correctly configured AM instance with the required transport key installed is able to decrypt and import the values.

  For more information, see "Creating Transport Keys".

Chapter 2

Getting Started with the Amster Command-line Interface

This chapter covers how to download and install Amster, and how to prepare your environment to connect to ForgeRock Access Management instances.

2.1. Prerequisites

Amster is a standalone client that does not require any other component from the ForgeRock Identity Platform to run. See the following list of prerequisites for installation:

- Amster requires a Java developer environment. Check the output of the **java -version** command to make sure your version is supported according to "Java Requirements" in the *Release Notes*.
- The `JAVA_HOME` environment variable must be set.

2.2. Installing Amster

The *ForgeRock BackStage* website hosts downloadable versions of Amster. For each release of AM you can download Amster as a `.zip` file.

After you download the `.zip` file, create a new directory for Amster and unzip the `.zip` file. For example:

```
$ mkdir /path/to/amster_5.5.2
$ unzip ~/Downloads/amster-5.5.2.zip -d amster_5.5.2
```

The following files and directories are extracted:

`bcprov-jdk15on-1.55.jar`

Third-party cryptography library, by Bouncy Castle.

`bcpkix-jdk15on-1.55.jar`

Third-party cryptography library, by Bouncy Castle.

`amster`

The **amster** command.

`README.md`

Amster readme file, with quick-start information.

`LICENSE`

ForgeRock's Amster terms of license.

`amster-5.5.2.jar`

The main Amster Java library.

`/legal-notices`

Directory containing legal notices relating to the Amster distribution.

`/samples`

Directory containing sample scripts for export, import, and others. For more information about this files, see "Amster Sample Scripts".

2.3. First Steps

Once Amster is extracted, run the **amster** command to start the client:

```
$ cd /path/to/amster
$ ./amster
Amster OpenAM Shell (5.5.2 build 24b5258daa, JVM: 1.8.0_131)
Type ':help' or ':h' for help
--------------------------------------------------------------------------------
am>
```

The version of Amster is included in the first line of output, as well as the version of the running JDK.

Note

If the **amster** command fails to load, make sure the `JAVA_HOME` environment variable is set, and that your JDK version is supported as per "Java Requirements" in the *Release Notes*.

To exit the client, run the **:exit** or **:q** commands:

```
am> :exit
$
```

To get a list of the commands available to the client, run the **:help** command:

```
am> :help
For information about Groovy, visit:
   http://groovy-lang.org

Available commands:
   connect         (am  ) Connect to an OpenAM instance
   create          (c   ) Create an OpenAM entity
   read            (r   ) Read an OpenAM entity
   update          (u   ) Update an OpenAM entity
   delete          (d   ) Delete an OpenAM entity
   query           (q   ) Query an OpenAM entity
   action          (a   ) Perform action an OpenAM entity
   import-config   (i   ) Import configuration into OpenAM
   export-config   (x   ) Export configuration from OpenAM
   replace         (rep ) Replace all matching text
   install-openam  (inst) Install OpenAM
   :help           (:h  ) Display this help message
   ?               (:?  ) Alias to: :help
   :exit           (:x  ) Exit the shell
   :quit           (:q  ) Alias to: :exit
   :load           (:l  ) Load a file or URL into the buffer
   .               (:.  ) Alias to: :load

For help on a specific command type:
   :help command
```

To show help information available for a particular command, run **:help *command***. For example:

```
am> :help connect
usage: connect [options] <baseurl>
Options:

  -i, --interactive
        If specified you will be prompted for credentials. Defaults to private
        key authentication.

  -k, --private-key
        Path to a private key file or directory containing one of amster_rsa,
        id_rsa or id_ecdsa. Defaults to {USER_HOME}/.ssh.

  -t, --connection-timeout
        The default timeout is 10 seconds. If specified, this parameter sets
        the timeout in seconds.

Connect to the OpenAM instance at the given URL.
Example:

  connect -i https://am.example.com/openam

  connect -i -t 30 https://am.example.com/openam
```

Tip

When a command does not proceed as expected, it can sometimes be helpful to start the **amster** command in debug mode and try again. To activate debug mode, start the **amster** command using the `-d` flag. For example:

```
$ ./amster -d
Listening for transport dt_socket at address: 6006
DEBUG [org.codehaus.groovy.tools.shell.BufferManager] Created new buffer with index: 0
DEBUG [org.codehaus.groovy.tools.shell.BufferManager] Buffers reset
DEBUG [org.codehaus.groovy.tools.shell.Parser] Using parser flavor:
 rigid
...
```

While in debug mode, the **amster** command output shows additional information, such as connection handshakes and Groovy calls.

2.4. Connecting to ForgeRock Access Management

Amster can connect to an AM instance using interactive login or using RSA or ECDSA key files, either over HTTP or HTTPS protocols. If you use self-signed certificates for AM, you must either:

- Import the certificates into the JVM's `cacerts` keystore on the Amster client.
- Run the **amster** command specifying the truststore containing the certificates and its type. For example:

```
$ ./amster -D javax.net.ssl.trustStore=/path/to/keystore.jceks -D javax.net.ssl.trustStoreType=jceks
```

2.4.1. Interactive Login Connections

To establish an interactive connection with AM, Amster makes use of the default authentication chain for administrator users configured in the AM instance. To locate this property, log in to AM and navigate to Realms > Top Level Realm > Authentication > Settings > Core.

The `ldapService` authentication chain, configured by default after AM installation, requires a valid user in AM. Log in as an administrative user, for example `amadmin`, to perform operations such as export and import of the configuration.

To Connect with Interactive Login

This procedure assumes the use of the `ldapService` chain. Perform the following steps to connect to a local or remote AM instance using interactive login:

1. Start the Amster command-line interface.
2. Run the **connect** command with the `--interactive` or the `-i` options:

```
am> connect --interactive https://openam.example.com:8443/openam
```

Tip

When using the **amster** command to import or export a significant amount of data, the default timeout of 10 seconds may be insufficient.

To increase the default timeout, add the **--connection-timeout <*seconds*>** option. For example:

```
am> connect --connection-timeout 45 --interactive https://openam.example.com:8443/openam
```

3. Specify username and password to authenticate to AM:

```
Sign in to OpenAM
User Name: amadmin
Password: *********
amster openam.example.com:8443>
```

2.4.2. Private Key Connections

Amster can connect to an AM instance by using a private key pair and an authentication module and a chain in AM. The private key must be available to the Amster client, and the AM instance must trust the client IP address and have the public key in its `authorized_keys` file. Successful connections create an `amAdmin` session in AM.

An install or an upgrade of AM creates the following infrastructure for Amster:

- The `Forgerock Amster` authentication module in the Top Level Realm. The module is enabled by default in new AM installations and disabled by default when upgrading an existing AM. For more information on how to configure this module, see "Amster Authentication Module Properties".
- The `amsterService` authentication chain in the Top Level Realm. Changing or removing this chain may result into not being able to connect with Amster.
- The following RSA key pair files, in SSH format:

Default Private Keypair Files

File Name	Description
`/path/to/openam/authorized_keys`	Holds the public keys of trusted Amster clients. AM check incoming Amster connections against these trusted keys. By default, contains a copy of the public key of a generated key pair that Amster can use. If this file exists in the configuration directory before a new install is performed, the file is not overwritten; the contents of the newly-created `amster_rsa.pub` file are appended to it instead.
`/path/to/openam/amster_rsa`	Contains the private key of a generated key pair that Amster can use.
`/path/to/openam/amster_rsa.pub`	Contains the public key of a generated key pair that Amster can use.

2.4.2.1. Connecting Locally with Default Private Key Files

An Amster installation local to a new AM instance can connect without further configuration.

To Connect Locally With the Default Key Pair

Perform the following steps to connect to a local AM instance using the default key pair:

1. Start the Amster command-line interface.
2. Run the **connect** command with the `--private-key` or the `-k` options:

```
am> connect --private-key /path/to/openam/amster_rsa https://openam.example.com:8443/openam
amster openam.example.com:8443>
```

2.4.2.2. Connecting to a Remote AM Instance

To connect to a remote AM instance, create a private key pair for Amster and append the contents of the public key to the `authorized_keys` file of the instance.

To Create and Configure a Private Key Pair

To create a new key pair and append the public key to the AM instance perform the following steps:

1. Login to the Amster server.
2. Create a directory for the keys, for example, `$HOME/.ssh`.
3. Run the **ssh-keygen** command to generate a key pair without passphrase. You can create RSA or ECDSA key pairs:

   - To create an RSA key pair, run the **ssh-keygen** command with the `-t rsa`option:

```
$ ssh-keygen -t rsa -N "" -f $HOME/.ssh/id_rsa -b 2048
Generating public/private rsa key pair.
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
78:ca:43:bc:0a:84:b0:ab:ac:40:96:49:48:84:80:63 root@amster_server
```

   - To create a ECDSA keypair, run the **ssh-keygen** command with the `-t ecdsa` option. You can create key pairs of 256, 384, or 521 curve sizes. For example:

```
$ ssh-keygen -t ecdsa -N "" -f $HOME/.ssh/id_ecdsa -b 521
Generating public/private ecdsa key pair.
Your identification has been saved in id_ecdsa.
Your public key has been saved in id_ecdsa.pub.
The key fingerprint is:
6b:b9:75:cb:42:07:91:25:a7:bf:d6:d0:bc:6f:5a:d7 root@amster_server
```

   The command generates two files, `id_rsa.pub` or `id_ecdsa.pub` containing the public key, and `id_rsa` or `id_ecdsa` containing the private key.

4. Append the contents of the `id_rsa.pub` or `id_ecdsa.pub` files into the `authorized_keys` file in your AM instance(s), for example, into `/path/to/openam/authorized_keys`.

5. Start the Amster command-line interface.

6. To connect to AM using a specific private key file, run the **connect** command with the `--private-key` or the `-k` options, specifying the path to the private key file. For example:

```
am> connect --private-key /amster/.ssh/id_rsa https://openam.example.com:8443/openam
amster openam.example.com:8443>
```

Chapter 3

Using the Amster Command-line Interface

This chapter covers how to use Amster to import and export configurations, and how to install a stand-alone AM instance.

Also covered is how to export and encrypt password values, how to script Amster usage, and how to use variables to customize imported configuration data.

3.1. Creating Transport Keys

To import and export encrypted password values in the configuration files you must generate a *transport key*, and install it in the keystore of each AM instance that you will be transporting passwords between.

The transport key must be stored in the default AM keystore, located at `/path/to/openam/openam/keystore.jceks`, and should have a key alias of `sms.transport.key`.

The presence of the transport key causes additional fields containing encrypted password values to appear in the exported configuration files. These additional fields have a `-encrypted` suffix, as shown below:

```
"authenticatorPushDeviceSettingsEncryptionKeystorePassword" : null,
"authenticatorPushDeviceSettingsEncryptionKeystorePassword-encrypted" :
"AAAAA0FFUwIQ1WDDMsxGoZMiRHhDQ+ywUfTMdGtYqEsvZZLV9W8ygfHi/5kBWjMpyg=="
```

Note

Encrypted password fields will only be added to REST calls made by administrative users, such as `amAdmin`.

Without a transport key present, all password fields are set to `null` in the exported configuration files, regardless of their actual value in the configuration.

To Generate and Install a New Transport Key

Use the **keytool** command to generate the transport key by performing the following steps:

1. Run the **keytool** command, specifying the location of the `.storepass` file as the password to use for the keystore and the location of the `.keypass` file as the password to use for the key aliases:

```
$ keytool -genseckey -alias "sms.transport.key" -keyalg AES -keysize 128 \
  -storetype jceks -keystore "/path/to/openam/openam/keystore.jceks" \
  -storepass:file "/path/to/openam/openam/.storepass" \
  -keypass:file "/path/to/openam/openam/.keypass"
```

2. You must restart AM for the transport key change to take effect.

   The instance will now be able to include encrypted passwords in the exported configuration files.

To decrypt and import configuration files that contain encrypted passwords, you must install the same transport key used to encrypt it into the keystore of the target AM instance.

To Duplicate and Install a Transport Key

Use the **keytool** command to export the transport key from the source instance keystore, and then install the result on the target instance keystore, by performing the following steps:

1. On the source instance, export the transport key into a keystore that can be transported to another instance by executing the following **keytool** command:

   ```
   $ keytool -importkeystore -srcstoretype jceks -srcalias "sms.transport.key" \
     -deststoretype jceks -destalias "sms.transport.key" \
     -srckeystore "/path/to/openam/openam/keystore.jceks" \
     -destkeystore "/path/to/openam/openam/transport_keystore.jceks" \
     -srckeypass:file "/path/to/openam/openam/.keypass" \
     -srcstorepass:file "/path/to/openam/openam/.storepass" \
     -destkeypass "myTransp0rtK3yP4ssword" \
     -deststorepass "myTransp0rtK3yP4ssword"
   ```

   This command exports the transport key to a temporary keystore file `/path/to/openam/openam/transport_keystore.jceks`, and set a store and key password of `myTransp0rtK3yP4ssword`. You need to use these temporary passwords when importing to the target instance.

2. Move the keystore file created in the previous step, in this example `transport_keystore.jceks`, to the filesystem of the target server.

3. On the target server, import the transport key into the AM keystore by executing the following **keytool** command:

   ```
   $ keytool -importkeystore -srcstoretype jceks -srcalias "sms.transport.key" \
     -deststoretype jceks -destalias "sms.transport.key" \
     -srckeystore "/path/to/openam/openam/transport_keystore.jceks" \
     -destkeystore "/path/to/openam/openam/keystore.jceks" \
     -srckeypass "myTransp0rtK3yP4ssword" \
     -srcstorepass "myTransp0rtK3yP4ssword" \
     -destkeypass:file "/path/to/openam/openam/.keypass" \
     -deststorepass:file "/path/to/openam/openam/.storepass"
   ```

   This command imports the transport key from the temporary keystore file `/path/to/openam/openam/transport_keystore.jceks` into the AM keystore, and set the transport key password to match the password used by the target keystore.

4. You must restart the target AM instance for the transport key change to take effect.

The target instance will now be able to correctly decrypt passwords stored in the imported configuration files.

Warning

Although the presence of the transport key only includes encrypted passwords in requests made by an administrative user, it is recommended to remove the transport key when imports and exports have been completed.

To Delete a Transport Key

1. Run the following **keytool** command:

```
$ keytool -delete -alias "sms.transport.key" -storetype jceks \
  -storepass:file "/path/to/openam/openam/.storepass" \
  -keystore "/path/to/openam/openam/keystore.jceks"
```

2. You must restart the target AM instance for the transport key change to take effect.

   The target instance will no longer include encrypted passwords nor be able to correctly decrypt passwords stored in configuration files.

3.2. Exporting Configuration Data

Amster can export configuration data from an AM instance. Export configuration data by using the **export-config** command.

The exported configuration data is written to a number of JSON-formatted files. The files are arranged in a hierarchy of global and realm configuration data.

To export encrypted password values in the configuration files you must generate and install a *transport key*. See "Creating Transport Keys".

Usage:

```
am> export-config --path Path [options]
```

`--path Path`

The path into which exported configuration files are placed.

Existing files will be overwritten if they exist. The path is created if it does not exist.

Options:

`--realms Realm [...]`

Space-separated list of realms from which to export. Specify the full path of each realm to export. Use a single forward-slash (/) to represent the top-level root realm.

Example: `/ /subRealm/subSubRealm`

Default: all

`--realmEntities *Entity [...]*`

Space-separated list of realm-based entities to export.

Use a space character in single-quotes (`' '`) to specify that no realm-based entities should be exported.

For a list of the available entities, see the Entity Reference.

Default: all

`--globalEntities *Entity [...]*`

Space-separated list of global entities to export.

Use a space character in single-quotes (`' '`) to specify that no global entities should be exported.

For a list of the available entities, see the Entity Reference.

Default: all

`--failOnError [true|false]`

If specified, the export process halts if an error occurs.

Default: `false`

`--listPasswords [true|false]`

If specified, the export process creates a listing of entities that contain password data. The listing is stored in a file in the root of the specified export directory.

Default: `false`

3.2.1. Exporting Configuration Data Examples

Before trying the following examples, start the Amster command-line interface, and connect to the AM instance from which to export data.

For information on connecting to instances, see "Connecting to ForgeRock Access Management".

Example 1

This example exports all configuration data, and will fail immediately if an error occurs.

```
am> export-config --path /tmp/myExportedConfigFiles --failOnError true
Export completed successfully
```

Example 2

This example exports the configuration for the `DataStoreModule`, `Scripts`, and `OAuth2Provider` entities in a subrealm of the root realm named `mySubRealm`.

Configuration data for global entities is not exported.

```
am> export-config --path /tmp/myExportedConfigFiles --realms '/mySubRealm' --realmEntities
 'DataStoreModule Scripts OAuth2Provider' --globalEntities ' '
Export completed successfully
```

3.3. Importing Configuration Data

Amster can import configuration data to an AM instance. Import configuration data by using the **import-config** command.

Caution

Imports overwrite any configuration that already exists in the target AM instance.

Before importing configuration data to an AM instance, consider the following points:

- You must connect to the AM instance where you will import the configuration data after starting the Amster command-line interface. For information, see "Connecting to ForgeRock Access Management".
- You must ensure that the configuration data you are trying to import is compatible with the version of AM you have deployed.

  For example, do not try to import configuration data exported from an AM 5 instance into an AM 6.5 instance.
- AM validates that external data stores are configured and available when creating connections to them, including when importing using Amster.

  You must ensure that each external datastore configured in the source instance has an equivalent datastore available and running before importing Amster configuration into the destination AM instance.
- When importing a full set of configuration data from an instance of AM, specify the `--clean` option to remove configuration settings from the target instance.

  The `--clean` option removes the following items from the target AM instance:

  - Realms, other than the ${tlr}.

- Authentication chains and modules.
- Server and site settings, other than the current server.
- Services.
- Secret ID Mappings and secret definitions.
- Scripts.
- Audit settings.
- Policies, policy sets and resource types.
- Identity store configuration.
- Agents, and agent groups.

Important

Do not use the `--clean` option if you are only importing a partial Amster export.

The target AM instance may not have the settings required to start up and operate if you do not replace the deleted settings by importing a complete set of configuration.

- To import encrypted password values in the configuration files you must install the *transport key* used to encrypt the data. For more information, see "Creating Transport Keys".
- You must ensure that any special characters in names and passwords in Amster shell variables are escaped as required by the Groovy language.

  For example, the dollar **$** character is a special character in Groovy. The following are two possible ways of escaping the **$** character:

  ```
  variable.name="/pa$$word/"
  variable.name='pa\\$\\$word'
  ```

  Note that you cannot use variables, such as `${varname}`, or configuration expressions, such as `&{varname}` if you convert a double-quoted string into a single-quoted string.

  Refer to the Groovy documentation for more information on escaping special characters in strings.

Usage:

```
am> import-config --path Path [options]
```

`--path` *`Path`*

The path containing configuration files to import.

Specify a directory to import from all correctly-formatted JSON files within that directory and recurse through each sub-directory, or specify an individual JSON file.

Options:

`--failOnError [true|false]`

If specified, the import process halts if an error occurs.

Default: `false`

`--clean [true|false]`

If specified, all configuration data is removed from the target AM instance before the import is performed.

Only set this option to `true` when importing a **full** set of configuration files into a new AM instance. Otherwise, the target instance may not function correctly.

Default: `false`

3.3.1. Importing Configuration Data Examples

Before trying the following examples, start the Amster command-line interface, and connect to the AM instance where you will import the configuration data.

For information on connecting to instances, see "Connecting to ForgeRock Access Management".

Example 1

This example imports configuration data, and will halt the import if an error occurs.

```
am> import-config --path /tmp/myExportedConfigFiles --failOnError true
Importing directory /tmp/
myExportedConfigFiles
...
Import completed successfully
```

Example 2

This example cleans all configuration from the target AM instance before importing a full set of configuration data, but will not halt the import if an error occurs.

```
am> import-config --path /tmp/myExportedConfigFiles --clean true --failOnError false
Cleaning global settings
Deleting JSON: Global JSON Handler
Deleting Scripting: 9de3eb62-f131-4fac-a294-7bd170fd4acb
Deleting Scripting: 7e3d7067-d50f-4674-8c76-a3e13a810c33
Deleting Scripting: c827d2b4-3608-4693-868e-bbcf86bd87c7
Global settings cleaned
Importing directory /tmp/
myExportedConfigFiles
...
Import completed successfully
```

3.4. Using Variables in Exported Configuration Files

Amster supports the use of variables in the configuration files. The values for the variables can be set from within Amster.

The syntax for a variable in the configuration files is `AMSTER{myVariableName}`.

The following example shows an excerpt from the configuration file for the email service, with variables inserted for a number of properties:

```
{
  "data" : {
    "_id" : "",
    "emailAddressAttribute" : "mail",
    "message" : null,
    "port" : AMSTER{emailServicePort},
    "password" : null,
    "from" : null,
    "hostname" : "AMSTER{emailServiceHostname}",
    "username" : "AMSTER{emailServiceUsername}",
    "sslState" : "SSL",
    "subject" : "AMSTER{emailServiceSubject}",
    "emailImplClassName" : "org.forgerock.openam.services.email.MailServerImpl",
    "_type" : {
      "_id" : "email",
      "name" : "Email Service",
      "collection" : false
    }
  }
}
```

Note that the JSON file must still be valid once variables have been replaced with their values. For example, ensure properties that require a string surround the variable with double quotes (").

Set the value for a property in Amster as follows:

```
am> myVariableName="Value"
===> Value
```

You can also set the values for variables in Groovy script files that are executed from within Amster. For more information, see "Scripting".

Attempting to import configuration files that contain variables that are not defined causes and error message similar to the following:

```
am> import-config --path /tmp/myExportedConfigFiles/realms/root/EmailService.json
Importing file /tmp/myExportedConfigFiles/realms/root/EmailService
.json
---------------------------------------------------------------------
IMPORT ERRORS
---------------------------------------------------------------------
Failed to import /tmp/myExportedConfigFiles/realms/root/EmailService.json  :
  Found template variable "emailServicePort" in a file with no associated variable in the binding
```

3.5. Scripting

You can create script files containing a series of commands and variable declarations, which can be loaded and executed within Amster.

Start each separate command or variable declaration on a new line. Use the backslash (\) character to represent line continuations.

For example, the following script installs an AM instance, and then exits the Amster command-line interface:

```
install-openam \
--serverUrl https://openam.example.com:8443/openam \
--authorizedKey /var/amster/authorized_keys \
--cookieDomain .example.com --adminPwd forgerock \
--cfgStoreDirMgrPwd password  --cfgStoreAdminPort 389  \
--acceptLicense  --cfgStore dirServer \
--cfgDir /root/openam
:exit
```

The Amster shell supports an `eval(String)` function, which evaluates any Amster command expressed as a string. For example, the function is required within looping structures:

```
for (i = 0; i < 4; i++) {
    eval("create DataStoreModule --realm / --body '{\"_id\":\"myDataStore$i\"}'")
 }
```

You must also use the `eval(String)` function when using Amster commands in conditional structures:

```
dbStatus = databaseName ? 'Found' : eval("create DataStoreModule --realm / --body '{\"_id\":\"myDataStore
\"}'")
```

To load and execute the commands within a script, use the `:load` command, as follows:

```
am> :load myScript.amster
```

You can specify more than one script to load. Scripts are loaded and executed in the order they are specified. If a command in a script fails, execution continues with the next command.

You can also invoke scripts by passing them in directly to Amster:

```
$ vi samples/myScript.amster
  connect http://openam.example.com:8080/openam -k /home/forgerock/am/amster_rsa
  :exit

$ ./amster samples/myScript.amster
Amster OpenAM Shell (5.5.2 build 95de0e129b, JVM: 1.8.0_151)
Type ':help' or ':h' for help
.
--------------------------------------------------------------------------------
  am>:load samples/myScript.amster
===> true
```

You can also inject Bash shell variables into an Amster script by using the `-D` parameter. For example, pass your shell variables into the Amster script, `export-config.amster`:

```
amUrl="http://openam.example.com:8080/openam"
amsterKey="/root/openam/amster_rsa"
configPath="/root/am-config"
.amster export-config.amster -D AM_URL=${amUrl} -D AMSTER_KEYS=${amsterKey} \
   -D AM_CONFIG_PATH=${configPath}
```

Next, use the variables in the Amster script, `export-config.amster`:

```
amUrl = System.getProperty("AM_URL")
amsterKey = System.getProperty("AMSTER_KEY")
exportPath = System.getProperty("AM_CONFIG_PATH")

connect amUrl -k amsterKey
export-config --path exportPath --failOnError
:exit
```

Note

Amster includes a number of example scripts in the `/path/to/amster/samples` directory. For more information, see "Amster Sample Scripts".

3.6. Installing Access Management with Amster

Amster can configure a deployed AM as a single, stand-alone instance, or as an instance that is part of a site.

By default, Amster configures AM to use an embedded DS server as the configuration and user stores, but you can specify an external configuration store. Configuring AM to use an external configuration store also requires an external user store, which defaults to the external configuration store unless otherwise specified.

Install AM configuration with Amster by using the **install-openam** command:

Usage:

```
am> install-openam \
   --serverUrl protocol://FQDN:port/URI \
   --adminPwd amAdmin_password \
   [options]
```

`--adminPwd amAdmin_password`

Specifies the password of the `amAdmin` user. If the `--cfgStoreDirMgrPwd` option is not specified, this value is also the password of the configuration store's directory manager user.

The password must be at least 8 characters in length.

`--serverUrl protocol://FQDN:port/URI`

Specifies the protocol, URL, port, and deployment URI of the AM instance. For example, `https://openam.example.com:8443/openam`.

Options:

`[options]`

Specifies optional parameters to configure properties such as the cookie domain, ports and passwords for the configuration store, and others.

For more information about the possible options, run the **:help install-openam** command or see the install-openam reference section.

3.6.1. Installing Access Management With Amster Examples

Before trying the following examples, make sure the AM instance is deployed and running but not yet configured. For more information, see the *ForgeRock Access Management Installation Guide*.

For more information about options available to the **install-openam** command, see the install-openam reference section.

Tip

Amster also supports scripting the installation process. For more information, see "Scripting".

You can find the following examples in this section:

- Installing AM With an Embedded Configuration Store
- Installing AM With an External Configuration Store

Installing AM With an Embedded Configuration Store

Example 1

This example installs AM with the default values:

```
am> install-openam \
> --serverUrl https://openam.example.com:8443/openam \
> --adminPwd forgerock \
> --acceptLicense

02/22/2017 05:16:20:932 AM GMT: Checking license acceptance...
02/22/2017 05:16:20:934 AM GMT: License terms accepted.
02/22/2017 05:16:20:936 AM GMT: Checking configuration directory /tomcat/openam.
02/22/2017 05:16:20:936 AM GMT: ...Success.
02/22/2017 05:16:20:936 AM GMT: Extracting OpenDJ, please wait...
02/22/2017 05:16:21:265 AM GMT: Complete
02/22/2017 05:16:21:265 AM GMT: Running OpenDJ setup
02/22/2017 05:16:21:265 AM GMT: Setup command: --cli --adminConnectorPort
 4444
--baseDN dc=openam,dc=forgerock,dc=org --rootUserDN cn=Directory
 Manager
--ldapPort 50389 --skipPortCheck --rootUserPassword xxxxxxx --jmxPort
 1689
--no-prompt --doNotStart --hostname openam.example.com --
noPropertiesFile
--backendType je
%0AConfiguring+Directory+Server+.....+Done.
0A%0ATo+see+basic+server+configuration+status+and+configuration%2C+you+can+launch%
0A%2Ftomcat_b%2Fopenam%2Fopends%2Fbin%2Fstatus%0A%0A02/22/2017 05:16:24:531 AM GMT: ...Success.
02/22/2017 05:16:24:531 AM GMT: ...Success
02/22/2017 05:16:24:531 AM GMT: Installing OpenAM configuration store in /tomcat/openam/
opends
...
02/22/2017 05:16:53:123 AM GMT: Configuring server instance.
02/22/2017 05:16:53:176 AM GMT: ...Done
02/22/2017 05:16:55:918 AM GMT: Creating demo user.
02/22/2017 05:16:55:942 AM GMT: ...Done
02/22/2017 05:16:55:943 AM GMT: Setting up monitoring authentication file.
Configuration complete!
```

Notes:

- If only the required parameters are supplied, Amster installs AM in a similar way the web configurator does when using the defaults.
- This example installs AM with an embedded configuration store.
- When installing AM locally to Amster, Amster stores AM's configuration in the home directory of the user that is running the **amster** command. For example, for the `tomcat` user, the configuration is stored in `/path/to/tomcat_home/openam`.

  To modify this behavior, use the `--cfgDir` option.
- If the default ports for the configuration store are already in use, the installer uses the next available free ports.
- The `demo` user is created in the embedded user store.

Example 2

This example installs AM specifying the configuration directory:

```
am> install-openam \
> --serverUrl https://openam.example.com:8443/openam \
> --adminPwd forgerock \
> --acceptLicense \
> --cfgDir /tomcat/openam2

02/22/2017 05:34:32:007 AM GMT: Checking license acceptance...
02/22/2017 05:34:32:007 AM GMT: License terms accepted.
02/22/2017 05:34:32:009 AM GMT: Checking configuration directory /tomcat/openam2.
02/22/2017 05:34:32:010 AM GMT: ...Success.
02/22/2017 05:34:32:010 AM GMT: Extracting OpenDJ, please wait...
02/22/2017 05:34:32:280 AM GMT: Complete
02/22/2017 05:34:32:280 AM GMT: Running OpenDJ setup
02/22/2017 05:34:32:265 AM GMT: Setup command: --cli --adminConnectorPort
 4444
--baseDN dc=openam,dc=forgerock,dc=org --rootUserDN cn=Directory
 Manager
--ldapPort 389 --skipPortCheck --rootUserPassword xxxxxxx --jmxPort
 1689
--no-prompt --doNotStart --hostname openam.example.com --
noPropertiesFile
--backendType je
%0AConfiguring+Directory+Server+.....+Done
.
...
02/22/2017 05:35:03:509 AM GMT: ...Done
02/22/2017 05:35:03:509 AM GMT: Setting up monitoring authentication file.
Configuration complete!
```

Notes:

- This example installs AM with an embedded configuration store.
- Amster will create the directory specified in `--cfgDir`
- The `demo` user is created in the embedded user store.

Example 3

This example installs two AM instances within a site that use embedded configuration stores.

First instance:

```
am> install-openam \
> --serverUrl https://openam1.example.com:8443/openam \
> --adminPwd forgerock \
> --acceptLicense \
> --cookieDomain example.com \
> --cfgDir /tomcat/openam1 \
> --lbSiteName TestSite01 \
> --lbPrimaryUrl http://site.example.com:80/openam

11/01/2018 14:20:20:932 PM GMT: Checking license acceptance...
11/01/2018 14:20:10:440 PM GMT: License terms accepted.
11/01/2018 14:20:10:440 PM GMT: Checking configuration directory /tomcat/openam1.
11/01/2018 14:20:10:440 PM GMT: ...Success.
11/01/2018 14:20:10:440 PM GMT: Extracting OpenDJ, please wait...
```

```
11/01/2018 14:20:11:631 PM GMT: Complete
11/01/2018 14:20:11:631 PM GMT: Running OpenDJ setup
11/01/2018 14:20:11:631 PM GMT: Setup command: --cli --adminConnectorPort
 4444
--baseDN dc=openam,dc=forgerock,dc=org --rootUserDN cn=Directory
 Manager
--ldapPort 50389 --skipPortCheck --rootUserPassword xxxxxxx --jmxPort
 1689
--no-prompt --doNotStart --hostname openam.example.com --
noPropertiesFile
--backendType je
%0AConfiguring+Directory+Server+.....+Done.
0A%0ATo+see+basic+server+configuration+status+and+configuration%2C+you+can+launch%
0A%2Ftomcat_b%2Fopenam%2Fopends%2Fbin%2Fstatus%0A%0A02/22/2017 05:16:24:531 AM GMT: ...Success.
11/01/2018 14:20:14:323 PM GMT: ...Success
11/01/2018 14:20:14:333 PM GMT: Installing OpenAM configuration store in /tomcat/openam1/
opends
...
11/01/2018 14:20:43:217 PM GMT: Configuring server instance.
11/01/2018 14:20:43:705 PM GMT: ...Done
11/01/2018 14:20:45:118 PM GMT: Creating demo user.
11/01/2018 14:20:45:242 PM GMT: ...Done
11/01/2018 14:20:45:243 PM GMT: Setting up monitoring authentication file.
Configuration complete!
```

Notes:

- Amster will create the directory specified in the `--cfgDir` option.
- The `demo` user is created in the embedded user store.
- Amster will create a site with the name specified in the `--lbSiteName` option, which can be accessed using the URL specified in the `--lbPrimaryUrl` option.
- The cookie domain is specified in the `--cookieDomain`. If not specified, Amster sets the cookie domain to the URL of the AM instance, which is not optimal when having multiple instances in a site.
- The example does not specify the configuration store's ports. To specify them, see the install-openam reference section.

Second instance:

```
am> install-openam \
> --serverUrl https://openam2.example.com:8443/openam \
> --adminPwd forgerock \
> --acceptLicense \
> --cfgDir /tomcat/openam2 \
> --cookieDomain example.com \
> --lbSiteName TestSite01 \
> --lbPrimaryUrl http://site.example.com:80/openam \
> --existingServerId https://openam1.example.com:8443/openam \
> --dsEmbReplReplPort1 50989 \
> --dsEmbReplReplPort2 50989 \
> --dsEmbReplFlag embReplFlag \
> --dsEmbReplHost2 https://openam1.example.com:8443/openam \
> --dsEmbReplAdminPort2 4444 \
```

```
> --pwdEncKey MneLwkkOokJx58znp7QyvGmiawmc2vl4

11/01/2018 14:24:20:932 PM GMT: Checking license acceptance...
11/01/2018 14:24:23:440 PM GMT: License terms accepted.
11/01/2018 14:24:23:440 PM GMT: Checking configuration directory /tomcat/openam2.
11/01/2018 14:24:23:440 PM GMT: ...Success.
11/01/2018 14:24:23:440 PM GMT: Extracting OpenDJ, please wait...
11/01/2018 14:24:24:631 PM GMT: Complete
11/01/2018 14:24:24:631 PM GMT: Running OpenDJ
 setup
...
11/01/2018 14:24:26:385 PM GMT: Creating embedded DS configuration store
 replica
...
11/01/2018 14:24:27:400 PM GMT: Updating remote references on server openam1.example.com:4444 .....
 Done
11/01/2018 14:24:27:401 PM GMT: Configuring Replication port on server openam2.example.com:4444 ....
.Done
Updating replication configuration for baseDN dc=openam,dc=forgerock,dc=org
 on
...
11/01/2018 14:24:28:206 PM GMT: Reinitializing system properties.
11/01/2018 14:24:28:394 PM GMT: ...Done
11/01/2018 14:24:28:406 PM GMT: Configuring server instance.
11/01/2018 14:24:28:460 PM GMT: ...Done
11/01/2018 14:24:28:770 PM GMT: Installing new plugins...
11/01/2018 14:24:28:846 PM GMT: Plugin installation complete.
11/01/2018 14:24:28:890 PM GMT: Setting up monitoring authentication file.
Configuration complete!
```

Notes:

- This example installs am AM instance with an embedded configuration store as part of the `TestSite01` site.

  As part of the site installation with embedded configuration store, Amster configures the new configuration store to replicate the data from the configuration store defined in the `dsEmbReplHost2` option.

- The password specified in the `--adminPwd` option must be the same password used across the site.
- Amster will create the directory specified in the `--cfgDir` option.
- The `demo` user is created in the embedded user store.
- The cookie domain is specified in the `--cookieDomain` option. When unspecified, Amster sets the cookie domain to the URL of the AM instance, which will cause issues when having multiple instances in a site.
- The example does not specify the configuration store's ports. To specify them, see the install-openam reference section.
- The `--pwdEncKey` specifies the encryption key used by the servers already in the site. To locate the encryption key value, navigate to Deployment > Servers > Server Name > Security > Encryption.

Failure to set this option to the appropriate value will cause the original encryption key to be overwritten, which will render the site unable to read the configuration and the user stores.

Installing AM With an External Configuration Store

Installing AM with an external configuration store requires manual configuration of the directory server. This is also true when specifying an external user store.

Note that you cannot install AM with an external configuration store that already contains configuration data.

Example 1

This example installs AM with an external configuration store. The install process will use the configuration store options for the external user store. Before running the **amster** command:

- Prepare the external configuration data store as detailed in the ForgeRock Access Management Installation Guide.

```
am> install-openam \
> --serverUrl https://openam.example.com:8443/openam \
> --adminPwd forgerock \
> --acceptLicense \
> --cfgStoreDirMgrPwd mypassword \
> --cfgStore dirServer \
> --cfgStoreHost opendj.example.com \
> --cfgStoreAdminPort 4444 \
> --cfgStorePort 1389 \
> --cfgStoreRootSuffix dc=example,dc=com

09/27/2017 03:37:29:345 PM BST: Checking license acceptance...
09/27/2017 03:37:29:989 PM BST: License terms accepted.
09/27/2017 03:37:29:991 PM BST: Checking configuration directory /Users/forgerock/openam.
09/27/2017 03:37:29:991 PM BST: ...Success.
09/27/2017 03:37:30:997 PM BST: Tag swapping schema files.
09/27/2017 03:37:30:002 PM BST: ...Success.
09/27/2017 03:37:30:004 PM BST: Loading Schema odsee_config_schema.ldif
09/27/2017 03:37:30:040 PM BST: ...Success
.
...
09/27/2017 03:37:38:520 PM BST: Loading Schema /Users/forgerock/openam/opendj_pushdevices.ldif
09/27/2017 03:37:38:703 PM BST: ...Success.
09/27/2017 03:37:38:811 PM BST: Installing new plugins...
09/27/2017 03:37:39:401 PM BST: Plugin installation complete.
09/27/2017 03:37:43:252 PM BST: Setting up monitoring authentication file.
Configuration complete!
```

Notes:

This example installs the configuration store and the user store in the `opendj.example.com` host. Both configuration and user data store use the same DS instance.

- When installing AM locally to Amster, Amster stores AM's configuration in the home directory of the user that is running the **amster** command. For example, for the `tomcat` user, the configuration is stored in `/path/to/tomcat_home/openam`.

To modify this behavior, use the `--cfgDir` option.

- If there is any problem setting up the configuration store, the installation process will exit with an error, and navigating to the AM will open the configuration page.
- The `demo` user is not created in the user store.

Example 2

This example installs AM with an external configuration and user stores. Before running the **amster** command:

- Prepare the external configuration data store as detailed in the ForgeRock Access Management Installation Guide.
- Ensure your user store does not have user data. If the user store has user data, you must prepare it as detailed in the ForgeRock Access Management Installation Guide.

```
am> install-openam \
> --serverUrl https://openam.example.com:8443/openam \
> --adminPwd forgerock \
> --acceptLicense \
> --cfgStoreDirMgrPwd mypassword \
> --cfgStore dirServer \
> --cfgStoreHost opendj.example.com \
> --cfgStoreAdminPort 4444 \
> --cfgStorePort 1389 \
> --cfgStoreRootSuffix dc=example,dc=com \
> --userStoreDirMgrPwd mypassword2 \
> --userStoreHost ldap.example.com \
> --userStoreType LDAPv3ForOpenDS \
> --userStorePort 1390 \
> --userStoreRootSuffix dc=example,dc=com

09/27/2017 03:33:47:989 PM BST: Checking license acceptance...
09/27/2017 03:33:47:989 PM BST: License terms accepted.
09/27/2017 03:33:47:991 PM BST: Checking configuration directory /Users/forgerock/openam.
09/27/2017 03:33:47:991 PM BST: ...Success.
09/27/2017 03:33:47:994 PM BST: Tag swapping schema files.
09/27/2017 03:33:48:006 PM BST: ...Success.
09/27/2017 03:33:48:006 PM BST: Loading Schema odsee_config_schema.ldif
09/27/2017 03:33:48:050 PM BST: ...Success
.
...
09/27/2017 03:33:54:691 PM BST: Loading Schema /Users/forgerock/openam/opendj_pushdevices.ldif
09/27/2017 03:33:54:800 PM BST: ...Success.
09/27/2017 03:33:54:847 PM BST: Installing new plugins...
09/27/2017 03:33:55:535 PM BST: Plugin installation complete.
09/27/2017 03:33:56:330 PM BST: Setting up monitoring authentication file.
Configuration complete!
```

Notes:

This example installs the configuration store in the `opendj.example.com` host and the user store in the `ldap.example.com` host.

- When installing AM locally to Amster, Amster stores AM's configuration in the home directory of the user that is running the **amster** command. For example, for the `tomcat` user, the configuration is stored in `/path/to/tomcat_home/openam`.

  To modify this behavior, use the `--cfgDir` option.

- If there is any problem setting up the configuration store, the installation process will exit with an error, and navigating to the AM will open the configuration page.

- The `demo` user is not created in the user store.

Example 3

This example installs two AM instances within a site that use an external configuration store. Before running the **amster** command:

- Prepare the external configuration data store as detailed in the ForgeRock Access Management Installation Guide.

First instance:

```
am> install-openam \
> --serverUrl https://openam1.example.com:8443/openam \
> --adminPwd forgerock \
> --acceptLicense \
> --cookieDomain example.com \
> --lbSiteName TestSite01 \
> --cfgDir /tomcat/openam1 \
> --lbPrimaryUrl http://site.example.com:80/openam \
> --cfgStore dirServer \
> --cfgStoreHost opendj.example.com \
> --cfgStoreAdminPort 3444 \
> --cfgStoreJmxPort 3689 \
> --cfgStorePort 3389 \
> --cfgStoreRootSuffix dc=examplecfg1,dc=com \
> --cfgStoreDirMgr "cn=Directory Manager" \
> --cfgStoreDirMgrPwd mySecretPassword

11/01/2018 14:37:40:221 AM GMT: Checking license acceptance...
11/01/2018 14:37:40:221 AM GMT: License terms accepted.
11/01/2018 14:37:40:221 AM GMT: Checking configuration directory /tomcat/openam1.
11/01/2018 14:37:40:221 AM GMT: ...Success
.
...
11/01/2018 14:38:46:070 PM BST: ...Success.
11/01/2018 14:38:46:070 PM BST: Loading Schema odsee_config_schema.ldif
11/01/2018 14:38:46:079 PM BST: ...Success
.
...
11/01/2018 14:41:36:387 PM BST: ...Success.
11/01/2018 14:41:36:388 PM BST: Installing new plugins...
11/01/2018 14:41:38:390 PM BST: Plugin installation complete.
11/01/2018 14:41:43:392 PM BST: Setting up monitoring authentication file.
Configuration complete!
```

Notes:

- Amster will create the directory specified in the `--cfgDir` option.
- Since the user store is not specified, Amster configures the configuration store as the user store.
- Amster will create a site with the name specified in the `--lbSiteName` option, which can be accessed using the URL specified in the `--lbPrimaryUrl` option.
- The cookie domain is specified in the `--cookieDomain`. If not specified, Amster sets the cookie domain to the URL of the AM instance, which is not optimal when having multiple instances in a site.

Second instance:

```
am> install-openam \
> --serverUrl https://openam2.example.com:8443/openam \
> --adminPwd forgerock \
> --acceptLicense \
> --cookieDomain example.com \
> --lbSiteName TestSite01 \
> --cfgDir /tomcat/openam2 \
> --lbPrimaryUrl http://site.example.com:80/openam \
> --existingServerId https://openam1.exaple.com:8443/openam \
> --cfgStore dirServer\
> --cfgStoreHost opendj.example.com \
> --cfgStoreAdminPort 3444 \
> --cfgStoreJmxPort 3689\
> --cfgStorePort 3389\
> --cfgStoreRootSuffix dc=examplecfg1,dc=com \
> --cfgStoreDirMgr "cn=Directory Manager" \
> --cfgStoreDirMgrPwd mySecretPassword \
> --pwdEncKey MneLwkkOokJx58znp7QyvGmiawmc2vl4

11/01/2018 14:53:22:773 PM: Checking license acceptance...
11/01/2018 14:53:22:773 PM GMT: License terms accepted.
11/01/2018 14:53:22:773 PM GMT: Checking configuration directory /tomcat/openam2.
11/01/2018 14:53:22:773 PM GMT: ...Success.
11/01/2018 14:53:22:773 PM GMT:Reinitializing system properties.
11/01/2018 14:53:22:773 PM GMT:...Done
11/01/2018 14:54:04:773 PM GMT:Configuring server instance.
11/01/2018 14:54:04:773 PM GMT: ...Done
11/01/2018 14:54:04:123 PM BST: Installing new plugins...
11/01/2018 14:54:04:124 PM BST: Plugin installation complete.
11/01/2018 14:54:10:322 PM BST: Setting up monitoring authentication file.
Configuration complete!
```

Notes:

- This example installs am AM instance with as part of the `TestSite01` site. Note that the configuration store details are the same as those used for the first server, since they are sharing the same DS instance.
- The password specified in the `--adminPwd` option must be the same password used across the site.
- Since the user store is not specified, Amster configures the configuration store as the user store.

- Amster will create the directory specified in the `--cfgDir` option.
- The cookie domain is specified in the `--cookieDomain` option. The cookie domain must be the same as the one used when installing the first instance, in this case, `example.com`. When unspecified, Amster sets the cookie domain to the URL of the AM instance, which is not optimal when having multiple instances in a site.

  Failure to set this option correctly may result in login failure to the new instance.
- The `--pwdEncKey` specifies the encryption key used by the servers already in the site. To locate the encryption key value, navigate to Deployment > Servers > Server Name > Security > Encryption.

  Failure to set this option to the appropriate value will cause the original encryption key to be overwritten, which will render the site unable to read the configuration and the user stores.

3.6.2. Troubleshooting Access Management Installations

The following table describes possible errors when installing AM with the `install-openam` command:

Error	Solution
`Invalid Suffix.`	Review that the suffix you are trying to use exists in DS.
`Cannot connect to Directory Server. Invalid Credentials.`	Review the credentials to connect to DS.
`Cannot connect to Directory Server. Connect Error: Connection refused.`	Review DS's host and/or connection port.
`Unexpected LDAP exception occurred.`	Review DS logs. DS may be stopped, or may have become unreachable during install.
`Connection to the server could not be established: Connection to the server could not be established, refer to install.log...`	Review that the container where AM will be installed trusts DS's SSL certificates.

3.7. Amster Usage Examples

In this section, you can find examples of tasks you can do with Amster.

Tip

For Amster examples in Docker and Kubernetes deployments, see the *ForgeRock DevOps Guide*.

3.7.1. Cloning an Access Management Instance

This example shows the high-level steps required to clone an AM instance, from exporting the configuration of the original instance, to installing the new instance and importing the configuration into it.

To Clone an AM Instance

Perform the following steps to clone an AM instance using Amster:

1. Create a transport key in the original AM instance, if one does not exist already. For more information, see "Creating Transport Keys".

2. Keep the transport key safe by exporting it to another keystore. The key is required to import the configuration into the new AM instance. For more information, see "To Duplicate and Install a Transport Key".

3. Connect to the original AM instance using the **amster** command. For more information, see "Connecting to ForgeRock Access Management".

4. Export all the configuration of the original AM instance using the **export-config** command. For more information, see "Exporting Configuration Data".

5. Take note of the value of the Password Encryption Key field on the original AM, for example, `O6QWwHPO4os+zEz3Nqn/2daAYWyiFE32`.

   To locate it, log in to the original AM instance and navigate to Deployment > Servers > *Server Name* >Security > Encryption.

6. In the new server, deploy the AM `.war` file in a web container, but do not configure it.

7. Install the new AM instance using the **install-openam** command, specifying the original AM password encryption key with the `--pwdEncKey` option. For example:

   ```
   am> install-openam \
   > --serverUrl https://openam.example.com:8443/openam \
   > --adminPwd forgerock \
   > --pwdEncKey O6QWwHPO4os+zEz3Nqn/2daAYWyiFE32 \
   > --acceptLicense
   ```

   For more information, see "Installing Access Management with Amster".

8. Import the transport key of the original AM instance into the keystore of the new AM instance. For more information, see "To Duplicate and Install a Transport Key".

9. Connect to the new AM instance using the **amster** command. For more information, see "Connecting to ForgeRock Access Management".

10. Import the configuration of the original AM instance using the **import-config** command. For more information, see "Importing Configuration Data".

3.7.2. Amster Sample Scripts

This section covers sample scripts and files found in the `/path/to/amster/samples` directory:

`transport-key.sh`

Shell script to help managing transport keys. You can use it as a template for your own scripts to create, delete, and export the key to another keystore.

Invoke the script's help for a list of possible actions:

```
$ ./transport-key.sh help
```

For more information about the transport key, see "Creating Transport Keys".

`realm.amster`

Amster script containing an example of different operations that can be done at realm level, such as creating a data store, displaying its configuration, modifying it, an deleting it.

For more information about writing scripts for Amster, see "Scripting".

`import-example.amster`

Amster script containing an example of the `import-config` command.

For more information about writing scripts for Amster, see "Scripting".

`export-example.amster`

Amster script containing an example of the `export-config` command.

Chapter 4
Reference

This chapter contains Amster reference.

4.1. Amster Authentication Module Properties

amster service name: `iPlanetAMAuthAmsterService`

Authorized Keys

Specifies the location of the `authorized_keys` file that contains the private and public keys used to validate remote **amster** client connections.

The default location for the `authorized_keys` file is the `/path/to/openam/` path. Its content is similar to an OpenSSH `authorized_keys` file.

amster attribute: `forgerock-am-auth-amster-authorized-keys`

Enabled

When enabled, allows **amster** clients to authenticate using PKI. When disabled, allows **amster** clients to authenticate using interactive login only.

amster attribute: `forgerock-am-auth-amster-enabled`

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

amster attribute: `forgerock-am-auth-amster-auth-level`

4.2. Command-Line Reference

Name

install-openam — Install Access Management

Synopsis

`install-openam` [options]

Description

Command to install and setup an AM instance.

The following parameters are required:

`--adminPwd` ***`amAdmin_password`***

Specifies the password of the `amAdmin` user. If the `--cfgStoreDirMgrPwd` option is not specified, this value is also the password of the configuration store's directory manager user.

The password must be at least 8 characters in length.

`--serverUrl` ***`protocol://FQDN:port/URI`***

Specifies the protocol, URL, port, and deployment URI of the AM instance. For example, `https://openam.example.com:8443/openam`.

The following options are available:

`--acceptLicense`

Specifies that the user accepts Amster usage terms and conditions.

`--authorizedKey` ***`path`***

Specifies the path to an SSH public key file. The content of this file is appended to the `authorized_keys` file of the newly-installed AM instance, allowing users to connect to it with Amster after the install completes.

For more information about connecting to AM with Amster, see "Connecting to ForgeRock Access Management".

`--cfgDir` ***`path`***

Specifies the configuration directory where AM stores files. It also stores the embedded directory server, when applicable.

Default: `$HOME/openam`

`--cfgStore [embedded|dirServer]`

Specifies the type of the configuration data store. Possible values are:

- `embedded`: Amster installs AM with an embedded DS server to act as the configuration, user, and CTS stores.
- `dirServer`: Amster installs AM on an external DS server to act as the configuration store.

  When you install AM with an external configuration store, you must also use an external user store. By default, the external user store is the same directory server instance as the external configuration store.

Default: `embedded`

`--cfgStoreAdminPort` ***port***

Specifies the administration port number for the configuration store.

Default: `4444`

`--cfgStoreDirMgr` ***username***

Specifies the distinguished name of the directory manager user for the configuration store.

Default: `cn=Directory Manager`

`--cfgStoreDirMgrPwd` ***password***

Specifies the password of the directory manager user for the configuration store.

Default: If not set, it takes the password defined for the `--adminPwd` option.

`--cfgStoreHost` ***FQDN***

Specifies the FQDN of the configuration store, for example, `opendj.example.com`

Default: `localhost`

`--cfgStoreJmxPort` ***port***

Specifies the Java Management eXtension port number for the configuration store.

Default: `1689`

`--cfgStorePort` ***port***

Specifies the LDAP or LDAPS port number for the configuration store.

Default: `50389`

`--cfgStoreRootSuffix` ***DN***

Specifies the root suffix DN for the configuration store.

Default: `dc=openam,dc=forgerock,dc=org`

`--cfgStoreSsl [SIMPLE|SSL]`

Specifies whether AM should connect to the configuration store over SSL. Possible values are `SIMPLE`, for non-secure connections, and `SSL`, for secure connections.

Default:`SIMPLE`

`--cookieDomain` ***domain***

Specifies the name of the trusted DNS domain AM returns to a browser when it grants a session ID to a user.

Default: FQDN used in the `--serverUrl` option

`--dsEmbReplAdminPort2` ***port***

Specifies the administration port number for the embedded configuration store defined in the `--dsEmbReplHost2` parameter. For example, `4444`.

Use this parameter when the `--cfgStore` parameter is set to `embedded`, and only when adding a new server to a site.

`--dsEmbReplFlag` **embReplFlag**

Specifies that the new server should install an embedded DS server as the configuration store. Setting this parameter causes Amster to configure the new embedded DS instance for data replication.

Use this parameter when the `--cfgStore` parameter is set to `embedded`, and only when adding a new server to a site.

`--dsEmbReplHost2` ***URI***

Specifies the URI of the embedded configuration store used by one of the instances of the AM site. For example, `https://openam1.example.com:8443/openam`.

You can choose any of the servers configured in your site.

The new AM server will replicate the configuration from this DS instance.

Use this parameter when the `--cfgStore` parameter is set to `embedded`, and only when adding a new server to a site.

`--dsEmbReplReplPort1` ***port***

Specifies the replication port number for the embedded configuration store of the new AM instance. For example, `50899`.

Use this parameter when the `--cfgStore` parameter is set to `embedded`, and only when adding a new server to a site.

`--dsEmbReplReplPort2` ***port***

Specifies the replication port number for the embedded configuration store defined in the `--dsEmbReplHost2` parameter. For example, `50899`.

Use this parameter when the `--cfgStore` parameter is set to `embedded`, and only when adding a new server to a site.

`--existingServerId` ***URI***

Specifies the URI of an existing AM instance in the site. This URI can match the one defined in the `--dsEmbReplHost2` parameter, but it is not mandatory. For example, `https://openam1.example.com:8443/openam`.

Use this parameter when the `--cfgStore` parameter is set to `embedded`, and only when adding a new server to a site.

`--installLocale` ***locale***

Specifies the locale to use during the install process.

Default: `en_US`

`--lbPrimaryUrl` ***URL***

Specifies the load balancer URL of the site, such as `https://lb.example.com:443/openam`

`--lbSiteName` ***name***

Specifies the name of the site to create, if any.

`--platformLocale` ***locale***

Specifies the default locale for the AM installation.

Default: `en_US`

`--pwdEncKey` ***key***

Specifies the encryption key value that is used to encrypt passwords in the AM instance. For example `O6QWwHPO4os+zEz3Nqn/2daAYWyiFE32`.

If you are installing an AM instance that will be making use of exiting data in a data store, you must provide the same encryption key value originally used to encrypt the passwords in those stores.

To locate the encryption key value in an AM instance, navigate to Deployment > Servers > *Server Name* > Security > Encryption.

If you are installing a new AM instance that will not be using existing data in a data store, you can leave this property empty. AM will generate a random encryption key during installation to encrypt the data that will be added to the data store.

This option is *required* when configuring an AM instance into a site, and must be set to the encryption key configured for the rest of the servers in the site. Failure to set this option to the appropriate value will cause the original encryption key to be overwritten, which will render the site unable to read the configuration and the user stores.

Default: No value; a random encryption key is generated during installation.

`--userStoreDirMgr` ***username***

Specifies the distinguished name of the directory manager user for the user store, for example, `cn=Directory Manager`.

Default: If not set, it takes the user defined for the `--cfgStoreDirManager` option.

`--userStoreDirMgrPwd` ***password***

Specifies the password of the directory manager user for the user store.

Default: If not set, it takes the password defined for the `--adminPwd` option.

`--userStoreDomainName` ***FQDN***

Specifies the Active Directory Domain Name, such as `ad.example.com`, when the `--userStoreType` option is set to `LDAPv3ForADDC`.

Default: Not set

`--userStoreHost` ***FQDN***

Specifies the FQDN of the configuration store, for example, `opendj.example.com`

Default: If not set, it takes the value defined for the `--cfgStoreHost` option.

`--userStorePort` ***port***

Specifies the LDAP or LDAPS port number for the configuration store.

Default: If not set, it takes the value defined for the `--cfgStorePort` option.

`userStoreRootSuffix` ***DN***

Specifies the root suffix DN for the user store.

Default: If not set, it takes the value defined for the `--cfgStoreRootSuffix` option.

`userStoreSsl [SIMPLE|SSL]`

Specifies whether AM should connect to the user store over SSL. Possible values are `SIMPLE`, for non-secure connections, and `SSL`, for secure connections.

Default: If not set, it takes the value defined for the `--cfgStoreSsl` option.

`--userStoreType \ [LDAPv3ForOpenDS|LDAPv3ForAD|LDAPv3ForADDC|LDAPv3ForADAM|LDAPv3ForODSEE|LDAPv3ForTivoli]`

Specifies the type of user store to use when installing AM with an external configuration store. Possible values are:

- `LDAPv3ForOpenDS`, for DS stores.
- `LDAPv3ForAD`, for Active Directoy with host and port settings.
- `LDAPv3ForADDC`, for Active Directory with domain name setting.
- `LDAPv3ForADAM`, for Active Directory Application Mode.
- `LDAPv3ForODSEE`, for Sun/Oracle DSEE.
- `LDAPv3ForTivoli`, for IBM Tivoli Directory Server.

When using the `LDAPv3ForADDC` store type, set up the `--userStoreDomainName` option to the Active Directory Domain Name, for example `ad.example.com`

Default: `LDAPv3ForOpenDS`

Appendix A. Getting Support

For more information or resources about OpenAM and ForgeRock Support, see the following sections:

A.1. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

  While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

- ForgeRock product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

A.2. Using the ForgeRock.org Site

The ForgeRock.org site has links to source code for ForgeRock open source software, as well as links to the ForgeRock forums and technical blogs.

If you are a *ForgeRock customer*, raise a support ticket instead of using the forums. ForgeRock support professionals will get in touch to help you.

A.3. Getting Support and Contacting ForgeRock

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, see https://www.forgerock.com.

ForgeRock has staff members around the globe who support our international customers and partners. For details on ForgeRock's support offering, including support plans and service level agreements (SLAs), visit https://www.forgerock.com/support.