Centrify for Web Applications

AD FS Configuration Guide

February 2017

Centrify Corporation

Legal notice

This document and the software described in this document are furnished under and are subject to the terms of a license agreement or a non-disclosure agreement. Except as expressly set forth in such license agreement or non-disclosure agreement, Centrify Corporation provides this document and the software described in this document “as is” without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. Some states do not allow disclaimers of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This document and the software described in this document may not be lent, sold, or given away without the prior written permission of Centrify Corporation, except as otherwise permitted by law. Except as expressly set forth in such license agreement or non-disclosure agreement, no part of this document or the software described in this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, or otherwise, without the prior written consent of Centrify Corporation. Some companies, names, and data in this document are used for illustration purposes and may not represent real companies, individuals, or data.

This document could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein. These changes may be incorporated in new editions of this document. Centrify Corporation may make improvements in or changes to the software described in this document at any time.

U.S. Government Restricted Rights: If the software and documentation are being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), in accordance with 48 C.F.R. 227.7202-4 (for Department of Defense (DOD) acquisitions) and 48 C.F.R. 2.101 and 12.212 (for non-DOD acquisitions), the government’s rights in the software and documentation, including its rights to use, modify, reproduce, release, perform, display or disclose the software or documentation, will be subject in all respects to the commercial license rights and restrictions provided in the license agreement.

Centrify, DirectControl, DirectAuthorize, DirectAudit, DirectSecure, DirectControl Express, Centrify User Suite, and Centrify Server Suite are registered trademarks and Centrify for Mobile, Centrify for SaaS, Centrify for Mac, DirectManage, Centrify Express, DirectManage Express, Centrify Identity Platform, Centrify Identity Service, and Centrify Privilege Service are trademarks of Centrify Corporation in the United States and other countries. Microsoft, Active Directory, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

Centrify software is protected by U.S. Patents 7,591,005; 8,024,360; 8,321,523; 9,015,103 B2; 9,112,846; 9,197,670; and 9,378,391.

The names of any other companies and products mentioned in this document may be the trademarks or registered trademarks of their respective owners. Unless otherwise noted, all of the names used as examples of companies, organizations, domain names, people and events herein are fictitious. No association with any real company, organization, domain name, person, or event is intended or should be inferred.

Contents

About this guide

The *Centrify Active Directory Federation Services Configuration Guide* describes how to configure single sign-on to Internet applications through federated identity authentication. This guide explains how to configure Centrify software for Apache, Tomcat, JBoss, WebLogic and WebSphere servers through the use of simple, sample applications. These sample applications enable you to test and verify configuration settings before deploying in a production environment.

Before using this guide you should have the AD FS 1.0 or AD FS 2.0 infrastructure in place and tested. For example, if you are using AD FS 2.0, you should have the claims provider and federated servers already identified and configured. You should also have the appropriate trust relationships in the resource and account partner organizations, and the claim rules already established.

Intended audience

This guide is for administrators and application developers who responsible for installing Centrify software and configuring web-based applications in a production or evaluation environment. Application developers should be familiar with native authentication methods and their development environment. You should also have a working knowledge of Windows, Active Directory, Active Directory Federation Services and the directory structure and configuration files required for the Java or Web application server you use.

Using this guide

Start with Chapter 1, “Authenticating using Active Directory Federation Services” to learn how Centrify extends authentication services provided by AD FS 1.x and AD FS 2.0.

Next proceed to the chapter corresponding to your web application server. Each chapter describes the steps to configure a specific

application server to use AD FS for authentication and how to modify your own applications to use AD FS.

Conventions used in this guide

The following conventions are used in this guide:

- `Fixed-width` font is used for sample code, program names, program output, file names, and commands that you type at the command line. When *`italicized`*, the fixed-width font is used to indicate variables. In addition, in command line reference information, square brackets (`[ ]`) indicate optional arguments.
- **Bold** text is used to emphasize commands, buttons, or user interface text, and to introduce new terms.
- *Italics* are used for book titles and to emphasize specific words or terms.
- Standalone software packages include version and architecture information in the file name.

Finding more information

Centrify provides extensive documentation targeted for specific audiences, functional roles, or topics of interest. If you want to learn more about Centrify and Centrify products and features, start by visiting the Centrify website. From the Centrify website, you can download data sheets and evaluation software, view video demonstrations and technical presentations about Centrify products, and get the latest news about upcoming events and webinars.

For access to documentation for all Centrify products and services, visit the Centrify documentation portal. From the Centrify documentation portal, you can always view or download the most up-to-date version of this guide and all other product documentation.

To get to the documentation portal, go to docs.centrify.com or https://www.centrify.com/support/documentation.

Contacting Centrify

You can contact Centrify by visiting our website, www.centrify.com. On the website, you can find information about Centrify office locations worldwide, email and phone numbers for contacting Centrify sales, and links for following Centrify on social media. If you have questions or comments, we look forward to hearing from you.

Getting additional support

If you have a Centrify account, click Support on the Centrify website to log on and access the Centrify Technical Support Portal. From the support portal, you can to search knowledge base articles, open and view support cases, download software, and access other resources.

To connect with other Centrify users, ask questions, or share information, visit the Centrify Community website to check in on customer forums, read the latest blog posts, view how-to videos, or exchange ideas with members of the community.

Authenticating using Active Directory Federation Services

When an organization uses Active Directory, users can sign on once and be authenticated to resources throughout the organization.

Active Directory Federation Services (AD FS) extends this Single Sign-On (SSO) capability: users can sign on once and be authenticated to Internet-facing applications.Centrify provides an interface to AD FS for Web application servers that are not running on IIS.

This chapter describes how Centrify participates in the AD FS infrastructure and the system requirements.

The following topics are covered:

- Introduction to federated identity management
- Working with AD FS 1.0
- Working with AD FS 2.0
- Centrify application support for AD FS
- AD FS infrastructure requirements
- How to proceed

Introduction to federated identity management

Microsoft provides **federated identity** management through Active Directory Federation Services (AD FS) available for Microsoft Windows Server 2003 R2 (AD FS 1.x) and Microsoft Windows Server 2008 (AD FS 1.x or AD FS 2.0). AD FS provides a Web single-sign-on (SSO) solution to authenticate a user to multiple Web applications, including external, third-party, or portal applications that are beyond the scope of the Active Directory identity store.

AD FS version 1.x and 2.0 are both claims-based, however, they differ dramatically in the federation service terminology, application set up and claims processing. The difference is transparent to the Web server,

however it does affect how you configure the sample applications on the AD FS server to test your installation.

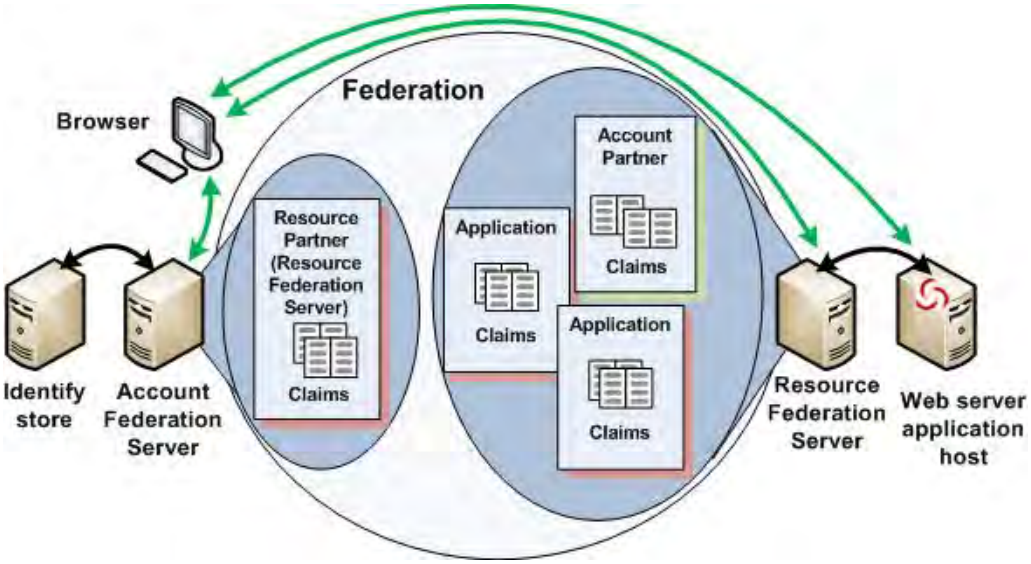
The following AD FS overview illustrates at a very basic level the interactions between the Web application server and federation services for AD FS 1.0 and 2.0 separately. For a comprehensive description of AD FS and 1.x and 2.0 go to the following URL:

http://technet.microsoft.com/en-us/library/cc772128%28WS.10%29.aspx

Working with AD FS 1.0

In this federated trust relationship, there are two account organizations:

- **Account Partner Organization**: Contains the users authorized to access the web-facing applications on a resource partner.
- **Resource Partner Organization:** Issues claims-based, security tokens for each web-facing application available to the account partner members.



Components of the account partner

There are three physical components are associated with the Account Partner Organization:

- Client browser: The computer from which the user launches the application. The user initiates the authentication from the browser. In addition, the browser is the locus for authentication protocol communications between the federation servers. (These communications are, however, transparent to the user.)
- Identity store: The central repository that contains all of the user accounts. For example, an Active Directory domain controller is the likely identity store for Centrify users; however, other types of identity stores are supported.
- Account federation server: Issues security tokens to users based on user authentication. The account federation server is also referred to as a claims provider.

  The account federation server authenticates the user against the identity store, extracts the attribute and group membership information, packages the data into claims, and generates and signs a security token (that includes the claim with other information) to return to the user. The user can use the token either within its own organization or to be sent to a resource partner organization.

Components of the resource partner

There are two physical components are associated with the Resource Partner Organization:

- Web server: The host on which the web-facing application is deployed.
- resource federation server: Issues security tokens to the user based on a security token that was previously issued by an account federation server. The resource federation server is also referred to as a relying partner.

Extending federation to Linux and UNIX

Centrify lets you use Linux- and UNIX-based Web application servers in a standard AD FS environment. For all intents and purposes, the Centrify modules provide the same service to the Java application servers as the Active Directory Federation Services Web SSO agent does to Microsoft IIS.

For more complete information about setting up and managing AD FS 1.0 federated trust relationships or configuring federation services for account or resource business partners consult the Microsoft documentation.

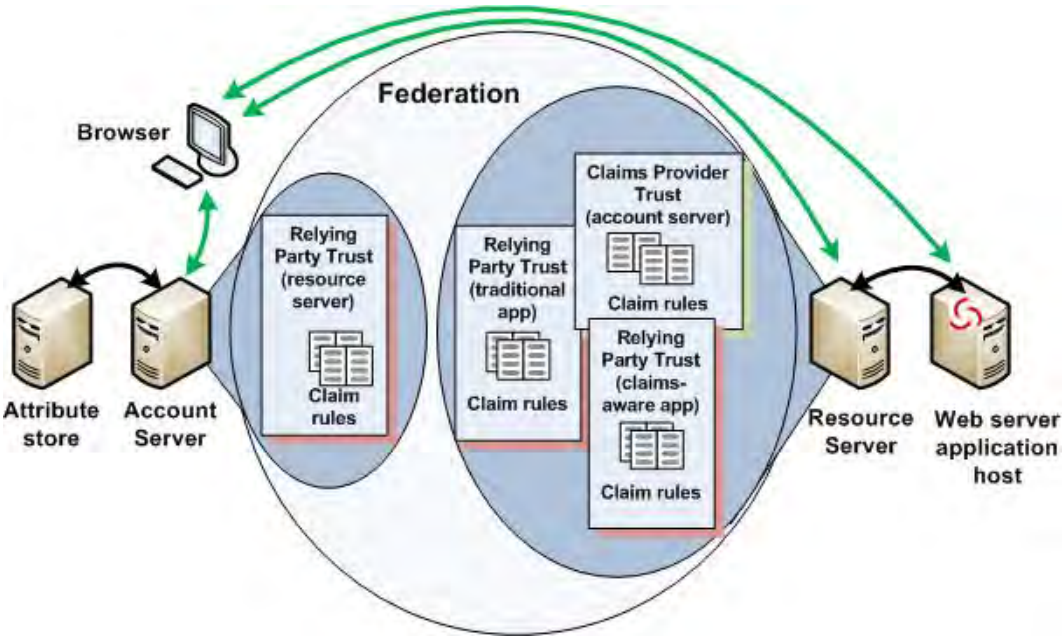
Working with AD FS 2.0

AD FS 2.0 still has the resource and account servers, however, the claims processing is done differently. In this model, you now have a

- claims provider trust: Very broadly, a trust object on a federated server (either resource or account) that maintains the relationship to a federated service that **provides** claims.
- relying party trust: a trust object on a federated server (either resource or account) that maintains the relationship with a federation service or application that **consumes** claims.

It is beyond the scope of this book to describe AD FS 2.0 components and protocols.

For the purposes of Centrify installation and testing, the following figure illustrates the configuration and communications:



Notice that both the account and resource servers have claims provider trusts and relying party trusts. For the purposes of Web server and application integration into AD FS 2.0, the Web server must have a trust relationship with the resource server that has the relying party for the applications it hosts.

Centrify application support for AD FS

The Centrify components for AD FS reside solely on the Web server; the rest of the AD FS infrastructure is unaffected. You do have to make minor changes to the Java applications. Sample configuration files are included in the package and instructions are provided.

The Centrify components support two types of Web applications for AD FS:

- **Claims-aware applications:** These Java applications are specifically designed to use the SAML-based security mechanisms and interfaces and make authorization decisions based on those claims. For these applications, DirectControl validates the claims

through the AD FS infrastructure and ultimately passes the appropriate claim and user information in the token from the resource federation server to the application.

- **Traditional applications:** These Java applications use the standard Java EE authentication and authorization mechanisms inherent to the Web server.

  Note Even though a traditional application uses standard Java EE authentication functions, the Centrify modules use claims to authenticate. To get a traditional application to work in an AD FS environment, you will need to make it look like it is claims-aware to AD FS. The instructions for this are provided in the platform sections.

This guide tells you how to configure the Web servers and applications to use the Centrify libraries to route authentication for traditional and claims-aware applications through AD FS. It also describes how to configure the sample applications and the federation servers to test your AD FS infrastructure. Use the applications first to confirm proper set up; then use the applications' configuration files as examples to update your own Java applications to use AD FS for authentication.

AD FS infrastructure requirements

The following table describes the AD FS federation server and client browser software you need to have in place to use Centrify.

On this computer	You need these infrastructure components
Resource federation server	Windows Server 2003 R2 (AD FS 1.0 only) or Windows Server 2008 or 2008 R2 for AD FS 2.0
	Active Directory domain controller
	DNS Service
	Internet Information Services (IIS) and ASP.NET
	Secure Socket Layer (SSL) certificate
	Be sure to update the c:\windows\systems32\drivers\etc\hosts file to include the IP address of the web application server.

On this computer	You need these infrastructure components
Account federation server	Windows Server 2003 R2 (AD FS 1.0 only) or Windows Server 2008 or 2008 R2 for AD FS 2.0 Identity store (AD FS 1.0) or Attribute Store (AD FS 2.0) in the account organization. Internet Information Services (IIS) and ASP.NET Secure Socket Layer (SSL) certificate
Client browser	A valid account in the account federation domain with a Web browser such as Internet Explorer or Firefox.

Note If you are using AD FS 2.0, the following restrictions apply:

- Only the "WS-Federation Passive protocol" is supported
- The SHA-1 hashing algorithm must be used when the SAML 2.0 profile is selected.

How to proceed

By now you should have your AD FS 1.x or 2.0 infrastructure in place and working (that is, users launching web applications running on IIS-based servers are authenticated using AD FS). See the Microsoft AD FS documentation for the instructions.

You should also have the Centrify agent installed on the Linux or UNIX computer you are using as the web server. If you have not already installed the Centrify web application support software package, see the *Authentication Guide for Java Applications* or the *Authentication Guide for Apache* for information about installing the software on the web server. After you have prepared the environment, see the instructions in the appropriate chapter to configure your web server to authenticate using AD FS.

Configuring an Apache Server for AD FS

At this juncture, you should have deployed and confirmed the proper installation and configuration of the Centrify package for Active Directory authentication. See the *Authentication Guide for Apache Servers* for those instructions.

This chapter has two logical parts:

- Part 1: Installing the software and sample applications: Describes how to complete the configuration of the Apache server and run the sample applications that test your AD FS-enabled configuration. When you have finished these instructions, go to Part 2: Modifying Apache applications.

  The following topics are covered in Part 1:

  - Validating SAML token with AD FS
  - Preparing the sample applications
- Part 2: Modifying Apache applications: Describes how to modify Apache applications to use Centrify for Web Applications for authentication via AD FS. This part has the following sections:
  - Working with claims-aware Apache applications
  - Working with traditional Apache applications
  - Verifying authentication on your own

Part 1: Installing the software and sample applications

The Centrify for Web Applications on Apache software package includes a separate AD FS-compliant module that enables an Apache web server running on Linux and UNIX platforms to authenticate and authorize web browser clients using Microsoft Active Directory Federation Services (AD FS) 1.0 or Active Directory Federation Services (AD FS) 2.0. The module supports two types of applications:

- **Claims-aware applications**: These applications are written to use the industry-standard Security Assertion Markup Language (SAML) Web single sign-on protocol. DirectControl for Web Applications for Apache validates and passes along any verified claims from the Web browser client to the application. Because the application has been designed to understand how to interpret the claims presented in the security token, the application itself decides on the level of service to provide to the client based on these claims presented.
- **Traditional applications**: These applications use standard Java EE authentication methods and do NOT make explicit use of the AD FS claims processing to authenticate users. The DirectControl for Web Applications modules for Apache use custom Apache directives that convert the authentication request into the standard AD FS-aware protocol to control access to the application.

The Centrify software package includes sample applications for both types for testing.

Preparing the Apache server

By now, you should have installed and tested the DirectControl package for Apache software for Active Directory authentication. Use the following instructions to install the corresponding module for AD FS authentication.

1 Enable Secure Socket Layer (SSL) support for the Apache server. (SSL is required if you are using AD FS but optional if you are using Active Directory.)

Note If you have SSL installed already, you do no need to repeat this step. You can verify whether you have configured support for SSL by opening a browser and trying to access the default web page using `https://localhost/` or `https://servername/`. You should always perform this test if you intend to use Centrify with Active Directory Federation Services.

Configuring the Apache server to use SSL varies depending on the version of Apache you use. For example, on Apache 2.0, you start SSL using the `apachectl startssl` command; however, in Apache 2.2, you configure SSL using directives in the main server configuration file.

- For Apache 1.3, add the `mod_ssl` module to the server configuration.
- For Apache 2.0, which includes the `mod_ssl` module, you must enable SSL support; for example, your `configure` command might look like this:

```
./configure --enable-ssl
```

  You can start the Apache 2.0 server with SSL by running the `apachectl startssl` command.

- For Apache 2.x, you can enable and configure SSL settings in directives in the main Apache server configuration file, `httpd.conf`. Once configured, you can start the Apache server with SSL by running the standard `apachectl start` command.

Note In an evaluation or lab environment, you can use a local self-signed certificate for testing purposes. In a production environment, however, you should ensure that the security certificates you accept provide an appropriate level of protection.

2 Edit the Apache server configuration file, `httpd.conf`, to include the DirectControl for Apache authentication module, `mod_adfs_centrifydc_xx` (where xx is the Apache version number; for example, 24 for version 2.4), for the platform. The simplest way to add the authentication module and directives is by using the `Include` directive and specifying the location of the sample application configuration file, `centrify`*xx*`.conf` (where *xx* is the mnemonic for the Apache version) in the Dynamic Shared Object section. The sample application configuration files are located in the `/usr/share/centrifydc/apache/samples/conf` directory.

`Include` directive examples:

- For Apache 2.2 on a 32-bit system, you would add the following in the Dynamic Shared Object section of the `httpd.conf` file:

```
Include /usr/share/centrifydc/apache/samples/conf/centrify22.conf
```

  For the 64-bit version you would add the following instead:

```
Include /usr/share/centrifydc/apache/samples/conf/centrify22_64.conf
```

- For Apache 1.3 on a 32-bit system

```
Include /usr/share/centrifydc/apache/samples/conf/centrify13.conf
```

  OR, on a 64-bit system:

```
Include /usr/share/centrifydc/apache/samples/conf/centrify13_64.conf
```

Alternatively, you can use the LoadModule directive in `httpd.conf` to load the authentication module. In this case, you would edit `httpd.conf` and add the `LoadModule` directive and an `include directive` for the `centrify.conf` sample application configuration file.

Validating SAML token with AD FS

The DirectControl for Web Applications on Apache authentication module for AD FS, `mod_adfs_centrifydc_xx` (where xx is the Apache version number; for example, 24 for version 2.4) plugs into the Apache Web server as a dynamically loaded module. The module does not require you to recompile or relink the Apache server. You simply need to add the module to the Apache server configuration file and restart the Apache server.

This module handles both traditional and claims-aware authentication models. All of the HTTP communication is through `https` using the Secure Socket Layer (SSL) protocol to encrypt communication.

To validate a Security Assertion Markup Language (SAML) token, `mod_adfs_centrifydc_xx` sends a message to the `adfsagent` daemon. The `adfsagent` process listens for requests to validate SAML tokens and upon success returns the validated token information to `mod_adfs_centrifydc_xx`. The `adfsagent` process also periodically sends an HTTPS request to the AD FS resource federation server to get any updated certificates on the AD FS server as well as any updated login URLs.

Starting and stopping the adfsagent process

The `adfsagent` process is installed but is NOT enabled or started automatically. If you are using AD FS for authentication you must manually enable and start `adfsagent`.

The command(s) you use to start `adfsagent` differ for each platform. In addition, on some platforms you just start adfsagent but on others you need to enable it first. Use following command(s) according to your platform. The commands you use to stop (and disable, if necessary) `adfsagent` are also listed.

- On generic Linux

```
chkconfig --add adfsagent
/etc/init.d/adfsagent start

/etc/init.d/adfsagent stop
chkconfig --del adfsagent
```

- On Debian Linux

```
update-rc.d adfsagent defaults
/etc/init.d/adfsagent start

/etc/init.d/adfsagent stop
update-rc.d -f adfsagent remove
```

- On SuSE Linux 10+

```
chkconfig adfsagent on
chkconfig adfsagent 5
/etc/init.d/adfsagent start

/etc/init.d/adfsagent stop
chkconfig adfsagent off
```

- On Solaris 8

```
ln -s /etc/init.d/adfsagent /etc/rc1.d/K12adfsagent
ln -s /etc/init.d/adfsagent /etc/rc2.d/K12adfsagent
ln -s /etc/init.d/adfsagent /etc/rc2.d/S72adfsagent
/etc/init.d/adfsagent start

/etc/init.d/adfsagent stop
rm /etc/rc1.d/K12adfsagent
rm /etc/rc2.d/K12adfsagent
rm /etc/rc2.d/S72adfsagent
```

- On Solaris 8

```
ln -s /etc/init.d/adfsagent /etc/rc1.d/K12adfsagent
ln -s /etc/init.d/adfsagent /etc/rc2.d/K12adfsagent
ln -s /etc/init.d/adfsagent /etc/rc2.d/S72adfsagent
/etc/init.d/adfsagent start

/etc/init.d/adfsagent stop
rm /etc/rc1.d/K12adfsagent
rm /etc/rc2.d/K12adfsagent
rm /etc/rc2.d/S72adfsagent
```

- On Solaris 10

```
svccfg import /var/svc/manifest/application/security/
adfsagent.xml
svcadm restart adfsagent
```

```
/usr/sbin/svcadm disable adfsagent
/usr/sbin/svccfg delete adfsagent
```

- On HPUX

```
/sbin/init.d/adfsagent start

/sbin/init.d/adfsagent stop
```

- On AIX:

```
mkssys -s adfsagent -p /usr/sbin/adfsagent -u 0 -S -n 2 -f 9 -R
/usr/bin/startsrc -s adfsagent

/usr/bin/stopsrc -s adfsagent
rmssys -s adfsagent
```

If a proxy server is required for the `adfsagent` process to reach the AD FS server, set the `HTTPS_PROXY` environment variable to the proxy host and port before starting the `adfsagent` process, as follows:

On Linux, Solaris, and HPUX systems:

1. Edit `/etc/init.d/adfsagent` or `/sbin/init.d/adfsagent` on HPUX.
2. Locate the line or lines that start with the following on the various systems:
   - Redhat EnterpriseLinux:

     `"daemon $adfsagent $OPTIONS"`
   - SuSE Linux:

     `"startproc $adfsagent_BIN"`
   - Debian Linux:

     `"start-stop-daemon --start --quiet --exec $binpath"`
   - Solaris and HPUX:

     `"[ -x "$EXEC" ] && $EXEC $OPTIONS"`
3. Add the following environment variable definition before the line in the previous step. This definition works for any of the specified systems.

   ```
   HTTPS_PROXY=proxyhost[:proxyport]
   export HTTPS_PROXY
   ```

where *proxyhost* is the proxy server host name and *proxyport* is the proxy server port number.

4 Restart the `adfsagent` process:

```
/etc/init.d/adfsagent restart # Linux & Solaris
/sbin/init.d/adfsagent restart # HPUX
```

On AIX, send the proxy information on the command line to start the `adfsagent` process, as follows:

- Stop `adfsagent`:

```
stopsrc -s adfsagent
```

- Start `adfsagent` and pass the proxy information:

```
startsrc -e "HTTPS_PROXY=proxyhost[:proxyport]" -s adfsagent >> /var/log/
centrifydc-install.log
```

where *proxyhost* is the proxy server host name and *proxyport* is the proxy server port number.

Note If you made modifications to `/etc/init.d/adfsagent`, or `/sbin/init.d/adfsagent`, be sure to save a copy before uninstalling or upgrading the DirectControl for Web Applications for Apache package. When you uninstall the DirectControl for Web Applications Apache package, it removes `/.../init.d/adfsagent`. When you install the package to upgrade an existing installation, it overwrites the file.

Configuring adfsagent log level and timeout setting

You can configure the `adfsagent` log level and timeout setting.

Log information for `adfsagent` is written to the `/var/log/centrifydc.log` file. The log level is set in the configuration file `/etc/centrifydc/centrifydc.conf` by the `log` parameter. The default the log level is `INFO`:

```
log: INFO
```

Use the `addebug` command to check or change the log level. To check the debug level:

```
# /usr/share/centrifydc/bin/addebug
```

DirectControl for Web Applications debug logging is off. To enable debug logging, execute the following command:

```
/usr/share/centrifydc/bin/addebug on
```

When you execute this command, the log level in the configuration file is changed to `DEBUG`:

```
log: DEBUG
```

To turn debug logging off, execute the following command:

```
/usr/share/centrifydc/bin/addebug off
```

Note For performance and security reasons, you should only enable DirectControl for Web Applications debugging when necessary. See the `addebug` man page for more information.

You can also change amount of time `adfsagent` waits for a message from `mod_adfs_centrifydc_xx` before timing out. The default is 60 seconds and is controlled by the parameter, `adfsagent.read.data.timeout`. If the load on your server is high you might set the timeout higher by editing `/etc/centrify/adfsagent.conf` and setting `adfsagent.read.data.timeout` to a greater number of seconds; for example:

```
adfsagent.read.data.timeout: 120
```

Preparing the sample applications

The install command you ran when you installed the Centrify for Web Applications modules on the Apache server also installed sample applications and a sample application configuration file. The files are in the `/usr/share/centrifydc/apache/samples` directory. To use these sample applications, you need to modify the application configuration file, `centrify.conf`, and include this file in your Apache server configuration.

To prepare the sample application's configuration file for Apache:

1. Log on to the Apache server and change to the `/usr/share/centrifydc/apache/samples/conf` directory.
2. Use a text editor to modify the `centrify.conf` file:

- Replace the `FEDERATION_SERVER_HOST_NAME` placeholder with the fully-qualified domain name for the resource server.
- Replace the `LOCAL_HOST_NAME` placeholder with the fully-qualified domain name for the Apache Web server. Make this change for each of the AD FS sample applications: `adfs-traditional`, `adfs-claims-aware` and `adfs-ordering`.

The URLs you specify in this file for the sample applications should be exactly the same as the URLs you specify when you add the DirectControl for Web Applications sample applications to the resource server for AD FS 1.0 or the relying party trust for AD FS 2.0.

3 Save your changes and close the file.

4 Include the sample application configuration file, `/usr/share/centrifydc/apache/samples/conf/centrify.conf`, in the main Apache server configuration file, `httpd.conf`, or copy the file to a configuration directory that is included in the main Apache server configuration file.

If you choose to include the sample application configuration file in the main Apache server configuration file, add a line similar to the following in the `httpd.conf` file:

```
include /usr/share/centrifydc/apache/samples/conf/
centrify.conf
```

5 Restart the Apache server. For example:

```
apachectl restart
```

Before you can run the sample application, you need to configure the AD FS account and resource servers to recognize the sample applications. For example, if you are using AD FS 1.0 you need to add the sample application to the resource server and create identity claims on the account server. If you are using AD FS 2.0, it's similar but different: you add identity claims and claim rules in the Claims Provider Trust and on the account server, add claim rules in the Relying Party Trust on the resource server, and add the sample application as a Relying Party Trust on the resource server. See Chapter 7, "Add sample applications and verify configuration" to complete the sample application installation and the Centrify for Web Applications testing.

Part 2: Modifying Apache applications

The following sections describe how to modify your claims-aware and traditional applications to use the Centrify for Web Applications authentication modules.

Working with claims-aware Apache applications

Claims-aware applications are applications that comply with the Security Assertion Markup Language (SAML) and WS-Federation standards for authorization messages. Because these applications are specifically written or modified to recognize the content and format of Active Directory Federation Service claims, Centrify for Apache simply passes any verified claims from the client to the application. The application then decides the level of service to provide the client based directly on those claims. If the application needs claims and none are present, it redirects to the Resource Federation Server to get claims.

Centrify for Apache uses the following environment variables to set values for claims-aware applications.

Environment variable	Sets this value
`IDENTITY_TYPE`	The type of the identity claim provided by the IDENTITY variable. The valid identity types are: • UPN • EmailAddress • CommonName For example, if the identity claim is the Universal Principal Name (UPN) of the client requesting service: `IDENTITY_TYPE=UPN`
`IDENTITY`	The identity of the client requesting service. For example, if the type of identity claim is the Universal Principal Name: `IDENTITY=john.doe@acme.com`

Environment variable	Sets this value
`GROUP_`*`name`*	The claim name with a value of `TRUE` for each group claim enabled. For example, if there is a group organization claim of `Purchaser` enabled for an application: `GROUP_Purchaser=TRUE`
`CUSTOM_`*`name`*	The claim name with the custom value defined for the claim. For example, if there is a custom organization claim of `Title` enabled for an application and the value of the custom claim is `Puchasing Agent`: `Group_Title=Purchasing Agent`
`ADFS_FEDERATION_URL`	The URL to which the application can redirect when it wants to retrieve authenticated claims or respond to a "log out" request.
`ADFS_ENTRY_URL`	The URL identifying the application. This value corresponds to the `EntryUrl` directive, and is used by the application when initiating a "log in" or "log out" request.
`ADFS_SAML`	The raw SAML XML for the claim.

The following is an example of the information a claims-aware application might receive from DirectControl for Web Applications for Apache:

```
IDENTITY=john.doe@acme.com
IDENTITY_TYPE=UPN
GROUP_Gold=TRUE
GROUP_Administrator=TRUE
GROUP_Purchaser=TRUE
CUSTOM_Title=Purchasing Agent
CUSTOM_DisplayName=John Doe
ADFS_FEDERATION_URL=https://dc1.acme.com/ADFS/fs/
federationserverservice.asmx
ADFS_ENTRY_URL=https://unix1.acme.com/orderapp/
mainpage.php
ADFS_SAML=
```

```
<saml:Assertion
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
AssertionID="_d66c7a92-f343-4b0a-a381-289db9c14dba"
      IssueInstant="2005-06-21T23:38:12Z"
Issuer="urn:federation:resource2" MajorVersion="1"
MinorVersion="1">
  <saml:Conditions NotBefore="2005-06-21T23:38:12Z"
NotOnOrAfter="2005-06-22T00:38:12Z">
    <saml:AudienceRestrictionCondition>
      <saml:Audience>https://hatter.wonder.land/test/
ADFS/ADFS.html</saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:Advice>
    <ClaimSource
xmlns="urn:microsoft:federation">urn:federation:account2
</ClaimSource>
  </saml:Advice>
  <saml:AuthenticationStatement
AuthenticationInstant="2005-06-21T23:37:02Z"
AuthenticationMethod="urn:federation:authentication:wind
ows">
    <saml:Subject>
      <saml:NameIdentifier Format="http://fabrikam.com/
federation/v1/upn">john.doe@acme.com</
saml:NameIdentifier>
    </saml:Subject>
  </saml:AuthenticationStatement>
  <saml:AttributeStatement>
    <saml:Subject>
      <saml:NameIdentifier Format="http://fabrikam.com/
federation/v1/upn">john.doe@acme.com</
saml:NameIdentifier>
    </saml:Subject>
    <saml:Attribute AttributeName="group"
AttributeNamespace="http://fabrikam.com/federation/v1/
group">
      <saml:AttributeValue>Gold</saml:AttributeValue>
    </saml:Attribute>
```

```
    <saml:Attribute AttributeName="group"
AttributeNamespace="http://fabrikam.com/federation/v1/
group">
      <saml:AttributeValue>Administrator</
saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute AttributeName="group"
AttributeNamespace="http://fabrikam.com/federation/v1/
group">
      <saml:AttributeValue>Purchaser</
saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute AttributeName="Title"
AttributeNamespace="http://fabrikam.com/federation/v1/
namevalue">
      <saml:AttributeValue>Purcashing Agent</
saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute AttributeName="DisplayName"
AttributeNamespace="http://fabrikam.com/federation/v1/
namevalue">
     <saml:AttributeValue>John Doe</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://
www.w3.org/2001/10/xml-exc-c14n#" />
     <SignatureMethod Algorithm="http://www.w3.org/2000/
09/xmldsig#rsa-sha1" />
      <Reference URI="#_d66c7a92-f343-4b0a-a381-
289db9c14dba">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/
09/xmldsig#enveloped-signature" />
          <Transform Algorithm="http://www.w3.org/2001/
10/xml-exc-c14n#" />
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/
09/xmldsig#sha1" />
```

```
        <DigestValue>Wm3bWWul/6Q4jVazyH/wW+2Buvw=</
DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>ArErm7gMEcfmeZjHQwFjgpCz/
GWljtxPXMjTnzs2tkwomxBnLnxzGJI5X1L9DoxV4leZtN83hwV+88PTe
rx+cX9SNNyaXxAKDRWEe3g8yBnrm7O+4lK4FvfCuobZweqwHkYDsKHbK
G3PC5sDfRU6BWWWqSsF7KFZ+EuGgazoMNk=</SignatureValue>
    <KeyInfo>
      <X509Data>
<X509Certificate>MIIB8jCCAV+gAwIBAgIQHVrew0qibqNL28eiaUB
BwzAJBgUrDgMCHQUAMCcxJTAjBgNVBAMTHHJlc291cmNlMi1kYzEucmV
zb3VyY2UyLnRlc3QwHhcNMDUwNTI3MDA1NzIwWhcNMDYwNTI3MDA1NzI
wWjAnMSUwIwYDVQQDExxyZXNvdXJjZTItZGMxLnJlc291cmNlMi50ZXN
0MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCtKdA0+E80Rg9ovmX
yewJT7B6OwWO5tzWeX0sdhLGJe6rvPZ2ppd7Fgp3CVdxlphHfDU29AEG
WOpDnf2FGpZkJsmJOSZFqaqVLWKiTXyeSpizPPMRTo9l4BhVvx5YyWge
UKaYTQZKhekwAugYdSX73q5HgYOtfo1/
z5fuSDaEvlwIDAQABoycwJTATBgNVHSUEDDAKBggrBgEFBQcDATAOBgN
VHQ8EBwMFALAAAAAwCQYFKw4DAh0FAAOBgQCOLu2RUFkJ9RGKG/
4b1BvrTD8woADI/OtX8zGVN/
cFJC7jSX05HcHGhslK3HE2TlM2AP1pLkusClnPnfgvnFiNujEQwfU0++
VFZ99jHv3SdFDpYdPx/5KTWmI/
+Lbz8U4qmn1m91NRmWDwUHceZzJA75jXXI+rseV7e4Ou5WCNSQ==</
X509Certificate>
      </X509Data>
    </KeyInfo>
  </Signature>
</saml:Assertion>
```

Working with traditional Apache applications

Traditional applications do not take advantage of Active Directory Federation Services claims directly. Instead Apache directives are used to control access to the application. For example, a page can be configured to require a specific group claim.

For traditional applications that do not use SAML tokens, DirectControl for Web Applications for Apache authentication and access control is handled through extensions to the standard Apache directives that appear in the Apache `httpd.conf` or `.htaccess` files.

For background information about configuring Apache authentication and access control, see the Apache documentation on *Authentication*. For more information about how and where to set Apache directives for web pages, directories, virtual web sites, and more, see the Apache documentation on *Directives*.

After the Centrify for Apache module is loaded into the Apache server, it provides the following additional directives:

Set this directive	To specify
`AuthType`	The authorization type to use. This directive must be set to `CENTRIFY_ADFS`, in all uppercase letters.
`FederationServerUrl`	The URL to use for the resource federation server.
`EntryUrl`	The URL to use as the starting page of the application This URL identifies the application to the Active Directory Federation Services. It must match the URL specified in the Resource Federation Service Application URL.

Set this directive	To specify
`VerifyFederationServer`	Whether to verify the Resource Federation Server's SSL certificate when retrieving the Federation Server's configuration information. This directive must be set to `true` or `false`. The default for the directive is `true`. If set to `true`, the SSL certificate used by the Resource Federation Server must be signed by a certificate authority with a certificate listed in the `cacerts.crt` file in the `/usr/share/centrifydc/apache/certs` directory. The `cacerts.crt` file is initialized with a list of commonly trusted certificate authorities. If you have your own certificate authority, you must include its certificate in PEM-encoded format in the `cacerts.crt` file. **Note** Self-signed certificates are often used in demonstration or evaluation environments, but self-signed certificates are not considered valid for a production environment. You should set this directive to `true` for any production deployment.
`SignoutUrl`	The URL of the image to display in the Federation Server's logout page to represent the application. This image is typically a small icon representing the application.
`MaxClockSkew`	The maximum acceptable clock skew in minutes. The acceptable clock skew is used in determining whether a claim is within its valid lifetime.

Set this directive	To specify
`XmlClaimValidation`	How the server should respond to claims that don't strictly adhere to the Microsoft standard for claims. • Set to `error` if you want the server to reject claims that don't conform to the standard. • Set to `warning` if you want the server to log a warning but accept claims that don't strictly adhere to the claims standard.
`XmlFederatedInfoValidation`	How the server should respond to information received from the Resource Federation Server that doesn't strictly adhere to the Microsoft standard. • Set to `error` if you want the server to reject information that doesn't conform to the standard. • Set to `warning` if you want the server to log a warning but accept information that doesn't strictly adhere to the standard.
`TrustInfoUpdateInterval`	The maximum number of minutes information received from the Resource Federation Server can be considered valid.
`CookiePath`	The path under which AD FS cookies are stored in the browser.
`MaxCookieSize`	The maximum size, in bytes, of cookies stored on the browser.
`Require` *option*	The group claim to which you are granting access for traditional applications. You can also use the Require directive to specify any authenticated user or no authentication required.

Specifying the Require directive

You can use the following command syntax to specify a group claim for the `Require` directive:

```
Require claimname [claimname ...]
```

You can specify multiple group claims by separating each group name with a space. If the name contains a space, you should enclose the entire name in double quotation marks.

The syntax for specify the group claim varies slightly depending on the version of the Apache server you are using. The following example shows how to specify group claims for the `HR Staff` and `Development` groups on Apache 2.0 and 2.2 servers:

```
Require "HR Staff" Development
```

The following example shows how to specify group claims for the `HR Staff` and `Development` groups on Apache 2.4 servers:

```
Require centrify-adfs-claim-group "HR Staff" Development
```

If you specify multiple group claims, the user will be able to log on if she has any group claim included in the list.

To specify “any authenticated user” or “no authentication is required” you can use the following syntax on Apache 2.0 and 2.2 servers:

```
Require valid-user
Require none
```

To specify “any authenticated user” or “no authentication is required” you can use the following syntax on Apache 2.4 servers:

```
Require centrify-adfs-valid-user
Require centrify-adfs-none
```

File location for directives

You can place the Centrify for Apache directives in either the `httpd.conf` or `.htaccess` file, depending on your needs. For example, if you centrally manage the configuration for different directories in the main configuration file, `httpd.conf`, you can add these directives where needed in a single file and maintain them in a

single location to avoid the per-request processing overhead of using individual `.htaccess` files.

Alternatively, you can provide these directives in separate `.htaccess` files so that different administrators can set their own directives for the directories they manage without making changes to the main configuration file or if you want to change the configuration without restarting the Apache server. If you decide to place the directives in individual `.htaccess` files, however, you must include the `AllowOverride` directive in the `httpd.conf` file, and be sure that this directive is set to `All` or, at a minimum, set to allow `AuthConfig` directives.

Sample directive settings

The following is an example of the directives set for a specific directory in the main `httpd.conf` file:

```
<Directory “usr/local/apache2/htdocs/ADFS-sample-dir”>
  AuthType                    CENTRIFY_ADFS
  FederationServerUrl       https://dc.ace.com/ADFS/fs/resource.asmx
  EntryUrl                    https://linux.ace.com/order/order.php
  SignoutUrl                  https://linux.ace.com/order/order.ico
  MaxClockSkew                5
  XmlClaimValidation          warning
  XmlFederatedInfoValidation warning
  TrustInfoUpdateInterval     5
  CookiePath                  /
  MaxCookieSize               2000
  Require                     purchaser
</Directory>
```

Verifying authentication on your own

To verify that accounts are authenticated using Active Directory, you may want to create a test directory within your Apache server’s root directory with a local copy of the authentication directives you plan to place in the main server configuration file (`httpd.conf`) or in individual access control files (`.htaccess`).

To verify authentication:

1. Check that the `AllowOverride` directive in the main server configuration file allows authentication directives to be set. You can

temporarily change this setting, if needed, for testing purposes. For example:

```
AllowOverride AuthConfig
```

2 Create your test directory and an `.htaccess` file with the directives to use.

For the `Require` directive, you can specify an existing Active Directory user or group or use `valid-user`.

3 Open your Web browser and attempt to access the test directory using a valid Active Directory logon name and password.

If authentication is successful, you will be logged on and able to access files in the test directory.

You can view information about every successful and failed authentication or authorization attempt in the Apache `error_log` file under the Apache installation directory. For example, the default location for the file in Apache 2.0 is `/usr/local/apache2/logs/error_log`. Any time a user attempts to access a protected Web page, Web directory, virtual Web site, or Web site, details about the success or failure are recorded in the log file. The logging level is controlled by the standard Apache `LogLevel` directive and can include errors, warnings, and informational messages.

Configuring a Tomcat Server for AD FS

At this juncture you should have deployed and confirmed the proper installation and configuration of the Centrify package for Active Directory authentication and completed the optional server configuration procedures (for example, run as a Windows service). See the *Centrify Authentication Guide for Java Applications* for those instructions.

This chapter serves two purposes:

- Finish Tomcat Configuration: This section tells you how to finish up Tomcat configuration to use Centrify and AD FS for user authentication.
- Configuring Tomcat applications to use AD FS: This section describes how to modify Java EE applications running on Tomcat servers to use Centrify and AD FS for authentication.

If you used the configure.pl option 0 (runs all of the configure.pl options) to install and configure the DirectControl for Web Applications package you can skip the first section; all of the Tomcat configuration required to use AD FS was done in the script. Proceed directly to Chapter 7, "Add sample applications and verify configuration" to run the sample applications to confirm your set up.

After you have completed the sample applications testing, return to Configuring Tomcat applications to use AD FS in this chapter to learn how to modify your applications to use AD FS.

Note In addition to Active Directory Federation Services, Tomcat requires that you have a supported version of the Java development environment (JDK) installed on the Web server. The version of the JDK required can vary depending on the version of Tomcat installed. For more information about JDK requirements, see your Tomcat documentation.

Finish Tomcat Configuration

This section is organizes as follows.

- Configure sample applications to work in your AD FS environment
- Configure Centrify AD FS Authenticator
- Configure SSL settings
- Configure your Tomcat server to trust the AD FS server

Note After you have completed these procedures you mus restart the Tomcat server for the changes to take effect

Configure sample applications to work in your AD FS environment

In this step, you customize the centrifydc_fs.xml for each sample application to work with your AD FS configuration.

When you installed the Centrify sample applications, you created a separate directory on the Web server for the following sample applications:

```
centrifydc-samples.war
centrifydc-kerberos.war
centrifydc-ntlm.war
centrifydc-basic.war
centrifydc-form.war
adfs-traditional.war
adfs-claims-aware.war
adfs-ordering.war
```

Before you can run the adfs-traditional.war, adfs-claims-aware.war, and adfs-ordering.war samples you need to do the following:

1. If you have not restarted the Tomcat server since you installed the Centrify package, run the Tomcat startup.sh script.
2. Edit each application's `WEB-INF/centrifydc_fs.xml` and replace the following keywords.
   - Replace `ADFS_SERVER_HOST` with your AD FS resource server host name.

- Replace APP_SERVER_HOST with the fully qualified domain name of your JBoss server computer.
- Replace 443 with the SSL port of your AD FS server (the default is 443)
- Replace 7002 with the SSL port that your Tomcat Server is running on (for example, 8443)

3 Restart the server.

If you used configure.pl option 0 to install and configure DirectControl for Web Applications package, this completes the Tomcat configuration. Go to Chapter 7, "Add sample applications and verify configuration" to run the sample applications to confirm your set up.

If you did DirectControl for Web Applications package installation and configuration manually proceed with the remaining configuration instructions.

Configure Centrify AD FS Authenticator

Confirm that your the `Authenticators.properties` file on the server includes the Centrify AD FS authenticator.

Open the following file:

```
CATALINA_HOME/server/classes/org/apache/catalina/startup/
Authenticators.properties
```

Add the following line to the end of the file:

```
CENTRIFYFS=com.centrify.fs.tomcat.SamlAuthenticator
```

Note In the *Centrify Authentiation Guide for Java Applications* you extracted the `Authenticators.properties` file and added the line `SPNEGO=com.centrify.dc.tomcat.SpnegoAuthenticator` to this file. See the Tomcat **Configure application server** section for the details.The file can contain both lines.

Continue with the next section to ensure that the Web application server and resource server have the proper certificates.

Configure SSL settings

ADFS requires your server to run with SSL. One easy way to do this for testing is to configure the server for SSL by generating a self-signed certificate and enabling the default Tomcat server SSL port 8443 in the `server.xml` file as described below. Do not use this configuration for production. See Tomcat documentation for more information on configuring Tomcat server for SSL.

1 Run the following command to generate a self-signed SSL certificate:

```
JDK_HOME/bin/keytool -genkey -keystore
CATALINA_HOME/conf/keystore.jks -alias ssl-server-cert-key
-keyalg RSA -dname "cn=localhost" -storepass changeit -keypass
changeit
```

where

- `CATALINA_HOME` is the base directory for your Tomcat installation
- `changeit` is the default password. If you have changed it replace that with your own.

2 Configure the Tomcat server to use the self-signed SSL certificate and enable the default SSL port. Edit `CATALINA_HOME/conf/server.xml` file with a text editor and do the following:

- Uncomment the `Connector` element that starts with `<Connector port="8443" …`
- Add the following attributes:

```
keystoreFile="$CATALINA_HOME/conf/keystore.jks"
```

- If the Tomcat server is running on an AIX-based computer, also add the following to server.xml:

```
algorithm="IbmX509"
sslProtocol="SSL"
```

Notes If you are using Centrify for AD FS authentication and are using Sun JDK 6 version 19, IBM JDK 6 refresh 7, or HP JDK 6.0.07 or higher, the TLS/SSL renegotiation option must be enabled for SSL communication with the AD FS server.

Use the following steps to enable the option.

On a Linux or UNIX system

1. Open the file `CATALINA_HOME/bin/setclasspath.sh`
2. Add the following to the end of the file:
   - For Sun's and HP's JDK:

```
JAVA_OPTS="$JAVA_OPTS -Dsun.security.ssl.allowUnsafeRenegotiation=true"
```

   - For IBM's JDK:

```
JAVA_OPTS="$JAVA_OPTS -Dcom.ibm.jsse2.renegotiate=ALL"
```

On a Windows system

1. Open the file `CATALINA_HOME/bin/setclasspath.bat`
2. Add the following definition to the end of the file:
   - For Sun's and HP's JDK:

```
set JAVA_OPTS=$JAVA_OPTS -Dsun.security.ssl.allowUnsafeRenegotiation=true
```

   - For IBM's JDK:

```
JAVA_OPTS=$JAVA_OPTS -Dcom.ibm.jsse2.renegotiate=ALL
```

If you are running the Tomcat server as a Windows service, add the Java options to the Tomcat service as follows

1. Stop the Tomcat service
2. Run the following to add the Java options:
   - For Sun's and HP's JDK:

```
CATALINA_HOME\bin\tomcatn.exe //US//%SERVICE_NAME% ++JvmOptions
"-DJAVA_OPTS=-Dsun.security.ssl.allowUnsafeRenegotiation=true"
```

   - For IBM's JDK:

```
CATALINA_HOME\bin\tomcatn.exe //US//%SERVICE_NAME% ++JvmOptions
"-DJAVA_OPTS=-Dcom.ibm.jsse2.renegotiate=ALL"
```

   where *tomcatn.exe* depends upon your Tomcat version:
   - For Tomcat 5.5: `tomcat5.exe`
   - For Tomcat 6.0: `tomcat6.exe`
   - For Tomcat 7.0: `tomcat7.exe`

See the links below for more information:

http://java.sun.com/javase/javaseforbusiness/docs/TLSReadme.html

http://www-01.ibm.com/support/docview.wss?uid=swg21415499

http://blogs.technet.com/b/srd/archive/2010/08/10/ms10-049-an-inside-look-at-cve-2009-3555-the-tls-renegotiation-vulnerability.aspx

http://docs.hp.com/en/JDKJRE60RN/jdk_rnotes_6.0.07.html#whatsnew

---

Configure your Tomcat server to trust the AD FS server

The Tomcat server must trust the Certificate Authority (CA) that issued the AD FS resource server's certificate used for SSL communications. You do this by loading the AD FS resource server's certificate in your application server's `cacerts` keystore.

Note Before you can perform this step you must export the CA certificate into a binary DER-encoded (`.cer`) file and copy it to your Tomcat server.

Run the following JDK `keytool` command to import the CA certificate into your Tomcat server's `cacerts` trusted keystore. You may need root permission if the `JDK_HOME/jre/lib/security/cacerts` file is owned by root.

```
JAVA_HOME/jre/bin/keytool -import -keystore
JAVA_HOME/jre/lib/security/cacerts -file <your-exported-
CA-certificate-file> -alias <a-unique-name-for-this-ca>
```

The `keytool` command prompts you for a password for the `cacerts` keystore. If you have never changed it, the default keystore password is `changeit`.

Unless you are using an AD FS proxy server this completes the Tomcat server configuration for AD FS authentication. Restart the Tomcat server for these changes to take effect and go to Chapter 7, "Add sample applications and verify configuration" to run the sample applications and confirm proper configuration.

Configure an AD FS Proxy Server

If an AD FS proxy server (also referred to as a Federation server proxy - an intermediary proxy service resides between an Internet client and a federation service that is behind a firewall) is required for the Tomcat server to communicate with the AD FS server, set the `JAVA_OPTS` environment variable to the proxy host and port before starting the Tomcat server.

On a Linux or UNIX system:

1. Edit `CATALINA_HOME/bin/setclasspath.sh`.
2. Add the following definition to `JAVA_OPTS`:

   ```
   JAVA_OPTS="$JAVA_OPTS
   -Dhttps.proxyHost=proxyhost -Dhttps.proxyPort=proxyport"
   ```

   where *`proxyhost`* is the proxy server host name and *`proxyport`* is the proxy server port number.
3. Restart the Tomcat server.

On a Windows system:

1. Edit `CATALINA_HOME/bin/setclasspath.bat`.
2. Add the following definition to `JAVA_OPTS`:

   ```
   set JAVA_OPTS=%JAVA_OPTS%
   -Dhttps.proxyHost=proxyhost -Dhttps.proxyPort=proxyport
   ```

   where *`proxyhost`* is the proxy server host name and *`proxyport`* is the proxy server port number.
3. Restart the Tomcat server.

If you are running the Tomcat server as a Windows service, send the Java options to the command line to launch Tomcat, as follows:

1. Stop the Tomcat service.
2. Add the following Java options to the Tomcat service:

   ```
   CATALINA_HOME\bin\tomcatn.exe //US//%SERVICE_NAME%
   ++JvmOptions
   "-Dhttps.proxyHost=proxyhost; -Dhttps.proxyPort=proxyport;"
   ```

where

- ***tomcatn.exe*** depends upon your Tomcat version:

  For Tomcat 5.5: `tomcat5.exe`

  For Tomcat 6.0: `tomcat6.exe`

  For Tomcat 7.0: `tomcat7.exe`
- *`proxyhost`* is the proxy server host name
- *`proxyport`* is the proxy server port number.

This completes the Tomcat server configuration for AD FS authentication. Restart the Tomcat server for these changes to take effect and go to Chapter 7, "Add sample applications and verify configuration" to run the sample applications and confirm proper configuration.

Configuring Tomcat applications to use AD FS

The sample applications are preconfigured to use Centrify and AD FS for authentication. This section describes how to modify Java EE applications to use AD FS. There are two types of Java applications:

- Traditional: Traditional applications do use Active Directory Federation Services claims directly. Instead they rely on standard Java EE APIs andJava EE security constraints defined in an application's `web.xml` file to authenticate users.

  For traditional applications, you add a context file modify its web.xml file and add the centrifydc-fs.xml file to its `WEB-INF` directory.
- Claims-aware: Claims-aware applications are applications that comply with Security Assertion Markup Language (SAML) and WS-Federation standards for authorization messages. Because these applications are specifically written or modified to recognize the content and format of Active Directory Federation Service claims, Centrify validates and passes along any verified claims from the client to the application. The application then decides the level of service to provide the client based directly on those claims. If the application needs claims and none are present, it redirects to AD FS to get claims.

For claims-aware applications you add a servlet filter to the `web.xml` file and copy the Centrify SAML JSP Tag library, `centrifydc_fs_taglib.jar` to the application's `WEB-INF/lib` directory.

Working with traditional applications

In addition to the `web.xml` file, each traditional application must have a `centrifydc_fs.xml` file in its `WEB-INF` directory. A template version of this file is installed by default in the `/usr/share/centrifydc/java/web/templates` directory on Linux- and UNIX-based computers or in the `C:\Program Files\Centrify\Centrify\java\web\templates` folder on Windows computers. Each Centrify AD FS sample applications also includes a copy of the `centrifydc_fs.xml` file.

The application configuration is composed of the following steps:

- Add the Centrify realm to the application
- Add the SAML filter to web.xml
- Set the authentication method and realm in web.xml
- Configure the security constraints in web.xml
- Modify centrifydc_fs.xml

Add the Centrify realm to the application

For traditional applications to use Active Directory Federation Services, the applications must be configured to use the Centrify SAML realm. You do this by creating a `context.xml` file for individual applications in the application's `WEB-INF` directory.

Use the following steps to use the Centrify SAML realm:

1. Navigate to the applications Web application archive (WAR) directory and create a `context.xml` file if one does not already exist.
2. Open the file and enter the following to specify the Centrify SAML realm:

```
<Context path="/my-app-name">
<Realm className="com.centrify.fs.tomcat.SamlRealm"/>
</Context>
```

3 Save your changes and close the file.

Add the SAML filter to web.xml

For applications that use the standard Java EE APIs, you need to modify the application's `web.xml` file to include the SAML filter. The SAML filter intercepts requests to the application that match the URL pattern you specify and enables the processing of AD FS messages for the application.

To add the SAML filter to Tomcat applications:

1 Open the application's `web.xml` file with a text editing tool. For example:

```
vi $CATALINA_HOME/server/webapps/appName/WEB-INF/web.xml
```

2 Add the following to the file to use the Centrify SAML filter as a servlet filter. The SAML filter intercepts requests and enables the processing of SAML-based AD FS messages for the application.

```
<filter>
<filter-name>saml</filter-name>
<filter-class>com.centrify.fs.SamlFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>saml</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

Save but do not close web.xml yet.

Set the authentication method and realm in web.xml

In this step change the authentication method and realm in your web.xml file to the realm configured in the server.xml file. For example:

```
<!-- Define the Login Configuration for this Application-->
<login-config>
<auth-method>CENTRIFYFS</auth-method>
<realm-name>SamlRealm</realm-name>
</login-config>
```

Save but do not close web.xml yet.

Configure the security constraints in web.xml

For each application, you need to modify the `web.xml` file to define the security constraints for the application.

To modify the security constraints for an application, edit the `<security-constraint>` and `<auth-constraint>` sections as appropriate to your application.

For example:

```
...
<security-constraint>
<web-resource-collection>
<web-resource-name>ProtectedResource</web-resource-name>
<url-pattern>/*</url-pattern>
<http-method>GET</http-method>
<http-method>POST</http-method>
</web-resource-collection>
<auth-constraint>
<role-name>user</role-name>
</auth-constraint>
</security-constraint>
```

Save your changes and close the web.xml file.

Modify centrifydc_fs.xml

To configure a traditional application to use Centrify and Active Directory Federation Services, you need to customize settings in the `centrifydc_fs.xml` file to identify the resource federation and the application URL that's been defined for the application in the resource federation server. By placing this file in an application's `WEB-INF` directory, you can control these custom settings on an application-by-application basis.

To customize the `centrifydc_fs.xml` file for an application:

1. Copy the default version of the `centrifydc_fs.xml` file from:
   - Linux and UNIX:`/usr/share/centrifydc/java/web/templates`
   - Windows: `C:\Program Files\Centrify\Centrify\java\web\templates`

to the application's `WEB-INF` directory.

2. Open the `centrifydc_fs.xml` file that is in the application's `WEB-INF` directory with a text editing tool.

3. Edit the appropriate sections of the template file to configure authentication through Active Directory Federation Services for the application. For example, modify the following elements in this file:

   - Set `federationServerUrl` to the URL of the Active Directory Federation Services resource federation server.
   - Set `entryUrl` to the URL for accessing an application. The entryUrl should be exactly the same as the entry URLs you specify when you add the Centrify sample applications to the resource federation server.
   - Set the attributes in the `RoleMapping` section to map Active Directory groups and users to the role names defined for an application in its `web.xml` file.

   The `centrifydc_fs.xml` template file is used for all Java-based applications. For more information about the `centrifydc_fs.xml` elements and settings defined in this file, see Chapter 9, "Understanding the centrifydc_fs.xml file."

4. Save your changes and close the file.

For further examples of customized `web.xml` and `centrifydc_fs.xml` files, see the Centrify sample applications in the `$CATALINA_HOME/webapps/adfs-*` directories.

Working with claims-aware applications

To handle claims and support Active Directory Federation Services, Centrify includes APIs that enable an application to query for claim information, query SAML information, obtain raw SAML tokens, and control log-on and log-off operations.

Use the following procedure to make a claims-aware application:

1. Copy the Centrify SAML JSP Tag library, `centrifydc_fs_taglib.jar`, from the `server/lib` directory under the Tomcat server to the application's `WEB-INF/lib` directory.

2 Open the application's `web.xml` file and add Centrify SAML as a servlet filter. The SAML filter intercepts requests to the application that match the URL pattern you specify and enables the processing of AD FS messages for the application.

For example:

```
<filter>
<filter-name>saml</filter-name>
<filter-class>com.centrify.fs.SamlFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>saml</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

3 Use the tags and attributes defined in the SAML JSP Tag library to make your application understand and respond to SAML-based claims.

You can then use the tags and attributes defined in the file `centrifydc_fs_taglib.jar` file to make your application understand and respond to SAML-based claims. The `aware.jsp` file in the `adfs-claims-aware` and `adfs-ordering` sample applications illustrate how to configure claims-aware applications. For reference information about the Centrify SAML JSP tags and attributes see Chapter 8, "Developing claims-aware Java EE applications for Centrify."

Configuring a JBoss Server for AD FS

At this juncture you should have deployed and confirmed the proper installation and configuration of the Centrify package for Active Directory authentication and completed the optional server configuration procedures (for example, run as a Windows service). See the *Centrify Authentication Guide for Java Applications* for those instructions.

This chapter serves two purposes:

- Finish JBoss configuration: This section tells you how to finish up JBoss configuration to use Centrify and AD FS for user authentication.
- Configure JBoss applications to use AD FS: This section describes how to modify Java EE applications running on JBoss servers to use Centrify and AD FS for authentication.

If you used the configure.pl option 0 (runs all of the configure.pl options) to install and configure the DirectControl for Web Applications package proceed to Finish JBoss configuration to configure the sample applications to work in your AD FS environment. Once that is complete, proceed to Chapter 7, "Add sample applications and verify configuration" to run the sample applications to confirm your set up.

If you chose the manual configuration option instead of configure.pl option 0, proceed to Finish JBoss configuration and perform all of the procedures in this section. Then, go to Chapter 7, "Add sample applications and verify configuration" to run the sample applications to confirm your set up.

After you have completed the sample applications testing, return to Configure JBoss applications to use AD FS in this chapter to learn how to modify your applications to use AD FS.

Finish JBoss configuration

This section is composed of the following procedures. You may not need to perform them all to complete the configuration:

- Configure sample applications to work in your AD FS environment
- Add Centrify AD FS Authenticator
- Configure SSL
- Configure JBoss server to trust the AD FS server
- Configure an AD FS Proxy Server

Configure sample applications to work in your AD FS environment

In this step, you customize the centrifydc_fs.xml for each sample application to work with your AD FS configuration.

When you installed the Centrify sample applications, you created a separate directory on the Web server for the following sample applications:

```
centrifydc-main.war
centrifydc-kerberos.war
centrifydc-ntlm.war
centrifydc-basic.war
centrifydc-form.war
adfs-traditional.war
adfs-claims-aware.war
adfs-ordering.war
```

For the adfs-traditional.war, adfs-claims-aware.war, and adfs-ordering.war samples only, you need to edit each application's `WEB-INF/centrifydc_fs.xml` and replace the following keywords.

- Replace `ADFS_SERVER_HOST` with your AD FS resource federation server host name.
- Replace 443 with your AD FS resource federation server SSL port.
- Replace `APP_SERVER_HOST` with the fully qualified domain name of your JBoss server computer.

- Replace 7002 with the SSL port that your JBoss server is running on (for example, 8443)

Restart the server.

If you used configure.pl option 0 to install and configure DirectControl for Web Applications package, this completes the JBoss configuration. Go to Chapter 7, "Add sample applications and verify configuration" to run the sample applications to confirm your set up.

If you did DirectControl for Web Applications package installation and configuration manually proceed with the remaining configuration instructions.

Add Centrify AD FS Authenticator

In this step you update the JBoss `Authenticators.properties` file on the server to add the AD FS authenticator.

Note You already extracted the `Authenticators.properties` file from the in the *Authentication Guide for Java Applications* to add the SPNEGO authenticator (see the *Add SPNEGO Authenticator* instructions in the *Configure JBoss application server* section). In these steps you modify the file to add the AD FS authenticator.

Open the file

```
org/apache/catalina/startup/Authenticators.properties
```

and add the following line to the end of the file:

```
CENTRIFYFS=com.centrify.fs.tomcat.SamlAuthenticator
```

Configure SSL

AD FS requires your server to run with SSL. SSL requires the application server to have the default SSL port set and a valid certificate.

Notes

If you are using Centrify for AD FS authentication and are using Sun JDK 6 version 19, IBM JDK 6 refresh 7, or HP JDK 6.0.07 or higher, the TLS/SSL renegotiation option must be enabled for SSL communication with the AD FS server.

Use the following steps to enable the option:

On a Linux or UNIX system

1 Open script.

In this step you edit the script. The script you use depends upon your JBoss version

- JBoss before version 7.0: Open the file `JBOSS_HOME/bin/run.sh.`
- JBoss version 7.0 and later: Open the `standalone.conf` or `domain.conf` file.

2 Add the following lines to enable TLS/SSL renegotiation:

- Find the line: `# Display our environment.`
- Add the following line **just before** that line

For Sun's and HP's JDK:

```
JAVA_OPTS="$JAVA_OPTS -Dsun.security.ssl.aalowUnsafeRenegotation=true"
```

For IBM's JDK:

```
JAVA_OPTS="$JAVA_OPTS -Dcom.ibm.jsse2.renegotiate=ALL”
```

On a Windows server

1 Open script.

In this step you edit the script. The script you use depends upon your JBoss version

- JBoss before version 7.0: Open the file `JBOSS_HOME/bin/run.bat.`
- JBoss version 7.0 and later: Open the `standalone.conf.bat` or `domain.conf.bat` file and add the following line:

2 Add the following lines to enable TLS/SSL renegotiation:

- Find the line: `set JBOSS_ENDORSED_DIRS=%JBOSS_HOME%\lib\endorsed`
- Add the following line **just after** that line
- For Sun's and HP's JDK:

```
set JAVA_OPTS=%JAVA_OPTS% -Dsun.security.ssl.allowUnsafeRenegotation=true
```

- For IBM's JDK:

```
set JAVA_OPTS=%JAVA_OPTS% -Dcom.ibm.jsse2.renegotiate=ALL
```

See the links below for more information:

http://java.sun.com/javase/javaseforbusiness/docs/TLSReadme.html

http://www-01.ibm.com/support/docview.wss?uid=swg21415499

http://blogs.technet.com/b/srd/archive/2010/08/10/ms10-049-an-inside-look-at-cve-2009-3555-the-tls-renegotiation-vulnerability.aspx

http://docs.hp.com/en/JDKJRE60RN/jdk_rnotes_6.0.07.html#whatsnew

---

If your application server does not yet have a valid certificate, an easy way to satisfy this requirement for testing is to generate a self-signed certificate and enable the default Tomcat server SSL port 8443 in the JBoss `server.xml` file. Do not use this configuration for production. See JBoss or Tomcat documentation for more information on configuring JBoss server for SSL.

1 Run the following command to generate a self-signed SSL certificate:

```
JDK_HOME/bin/keytool -genkey -keystore
JBOSS_HOME/server/myserver/conf/keystore.jks -alias
ssl-server-cert-key -keyalg RSA -dname "cn=localhost"
-storepass changeit -keypass changeit
```

where

- `JBOSS_HOME` represents the JBoss home directory
- *`myserver`* is the JBoss server profile, for example, `default` or `all`.
- `changeit` is the default password. If you have changed it replace that with your own.)

2 Configure the JBoss server to use the self-signed SSL certificate and enable the default SSL port. Edit the `server.xml` file corresponding to your Tomcat version and do the following:

- Uncomment the `Connector` element that starts with `<Connector port="8443" …`
- Add the following attributes:

```
keystoreFile="$CATALINA_HOME/conf/keystore.jks"
keystorepass’”changeit”
```

- If the JBoss server is running on an AIX-based computer, also add the following to `server.xml`:

```
algorithm="IbmX509"
sslProtocol="SSL"
```

Configure JBoss server to trust the AD FS server

The JBoss server must trust the Certificate Authority (CA) that issued the AD FS resource server’s certificate used for SSL communications. You do this by loading the AD FS resource server’s certificate in your application server’s `cacerts` keystore.

Note Before you can perform this step you must export the CA certificate into a binary DER-encoded (`.cer`) file and copy it to your JBoss server.

Run the following JDK `keytool` command to import the CA certificate into your JBoss server’s `cacerts` trusted keystore. You may need root permission if the `JDK_HOME/jre/lib/security/cacerts` file is owned by root.

```
JAVA_HOME/jre/bin/keytool -import -keystore
JAVA_HOME/jre/lib/security/cacerts -file <your-exported-
CA-certificate-file> -alias <a-unique-name-for-this-ca>
```

The `keytool` command prompts you for a password for the `cacerts` keystore. If you have never changed it, the default keystore password is `changeit`.

Unless you are using an AD FS proxy server this completes the JBoss server configuration for AD FS authentication. Restart the JBoss server for these changes to take effect and go to Chapter 7, “Add sample

applications and verify configuration" to run the sample applications and confirm proper configuration.

Configure an AD FS Proxy Server

If an AD FS proxy server (also referred to as a Federation server proxy - an intermediary proxy service resides between an Internet client and a federation service that is behind a firewall) is required for the JBoss server to communicate with the AD FS server, set the `JAVA_OPTS` environment variable to the proxy host and port before starting the JBoss server, as follows:

On a Linux or UNIX system:

1. Open the script.

   The script you edit depends upon the JBoss version:

   - JBoss before version 7.0: Edit `JBOSS_HOME/bin/run.sh`.
   - JBoss version 7.0 and later: Edit the `standalone.conf` or `domain.conf` file.

2. Find the line: `# Display our environment.`

3. Add the following definition to `JAVA_OPTS` **just after** that line:

   ```
   JAVA_OPTS="$JAVA_OPTS
   -Dhttps.proxyHost=proxyhost -Dhttps.proxyPort=proxyport"
   ```

   where *`proxyhost`* is the proxy server host name and *`proxyport`* is the proxy server port number.

4. Restart the JBoss server.

On a Windows systems

1. Open the script.

   The script you edit depends upon the JBoss version:

   - JBoss before version 7.0: Edit `JBOSS_HOME/bin/run.bat`.
   - JBoss version 7.0 and later: Edit the `standalone.conf.bat` or `domain.conf.bat` file.

2. Find the line: `set JBOSS_ENDORSED_DIRS=%JBOSS_HOME%\lib\endorsed`

3 Add the following definition to JAVA_OPTS **just before** that line:

```
set JAVA_OPTS=%JAVA_OPTS%
-Dhttps.proxyHost=proxyhost -Dhttps.proxyPort=proxyport
```

where *proxyhost* is the proxy server host name and *proxyport* is the proxy server port number.

4 Restart the JBoss server.

This completes the JBoss server configuration for AD FS authentication. Restart the JBoss server for these changes to take effect and go to Chapter 7, "Add sample applications and verify configuration" to run the sample applications and confirm proper configuration.

Configure JBoss applications to use AD FS

The sample applications are preconfigured to use Centrify and AD FS for authentication. This section describes how to modify Java EE applications to use AD FS. There are two types of Java applications:

- Traditional: Traditional applications do use Active Directory Federation Services claims directly. Instead they rely on standard Java EE APIs andJava EE security constraints defined in an application's `web.xml` file to authenticate users.

  For traditional applications, you add a context file modify its web.xml file and add the centrifydc-fs.xml file to its `WEB-INF` directory.

- Claims-aware: Claims-aware applications are applications that comply with Security Assertion Markup Language (SAML) and WS-Federation standards for authorization messages. Because these applications are specifically written or modified to recognize the content and format of Active Directory Federation Service claims, Centrify validates and passes along any verified claims from the client to the application. The application then decides the level of service to provide the client based directly on those claims. If the application needs claims and none are present, it redirects to AD FS to get claims.

  For claims-aware applications you add a servlet filter to the `web.xml` file and copy the Centrify SAML JSP Tag library, `centrifydc_fs_taglib.jar` to the application's `WEB-INF/lib` directory.

Working with traditional applications

In addition to the `web.xml` file, each traditional application must have a `centrifydc_fs.xml` file in its `WEB-INF` directory. A template version of this file is installed by default in the `/usr/share/centrifydc/java/web/templates` directory on Linux- and UNIX-based computers or in the `C:\Program Files\Centrify\Centrify\java\web\templates` folder on Windows computers. Each Centrify AD FS sample applications also includes a copy of the `centrifydc_fs.xml` file.

The application configuration is composed of the following steps:

- Add the Centrify SAML realm to the application
- Add the SAML filter to web.xml
- Set the authentication method and realm in web.xml
- Configure the security constraints in web.xml
- Modify centrifydc_fs.xml

Add the Centrify SAML realm to the application

For traditional applications to use Active Directory Federation Services, the applications must be configured to use the Centrify SAML realm. You do this by creating a `context.xml` file for individual applications in the application's `WEB-INF` directory.

Use the following steps to use the Centrify SAML realm:

1. Navigate to the applications Web application archive (WAR) directory and create a `context.xml` file if one does not already exist.
2. Open the file and enter the following to specify the Centrify SAML realm:

   ```
   <Context>
   <Realm className=""com.centrify.fs.tomcat.SamlRealm"/>
   </Context>
   ```

3. Save your changes and close the file.

Add the SAML filter to web.xml

For applications that use the standard Java EE APIs, you need to modify the application's `web.xml` file to include the SAML filter. The SAML filter intercepts requests to the application that match the URL pattern you specify and enables the processing of AD FS messages for the application.

To add the SAML filter to JBoss applications:

1. Open the application's `web.xml` file.
2. Add the following to the file to use the Centrify SAML filter as a servlet filter. The SAML filter intercepts requests and enables the processing of SAML-based AD FS messages for the application.

```
<filter>
<filter-name>saml</filter-name>
<filter-class>com.centrify.fs.SamlFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>saml</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

   Save but do not close web.xml yet.

Set the authentication method and realm in web.xml

In this step you change the auth-method setting in web.xml to use the custom authenticator CENTRIFYFS and set the realm name to the realm name configured context.xml.

In the web.xml `login-config` section add the following lines:

```
<!-- Define the Login Configuration for this Application-->
<login-config>
<auth-method>CENTRIFYFS</auth-method>
<realm-name>SamlRealm</realm-name>
</login-config>
```

Save but do not close web.xml yet.

Configure the security constraints in web.xml

For each application, you need to modify the `web.xml` file to define the security constraints for the application.

To modify the security constraints for an application, edit the `<security-constraint>` and `<auth-constraint>` sections as appropriate to your application.

For example:

```
...
<security-constraint>
<web-resource-collection>
<web-resource-name>ProtectedResource</web-resource-name>
<url-pattern>/*</url-pattern>
<http-method>GET</http-method>
<http-method>POST</http-method>
</web-resource-collection>
<auth-constraint>
<role-name>user</role-name>
</auth-constraint>
</security-constraint>
```

Save your changes and close the `web.xml` file.

Modify centrifydc_fs.xml

To configure a traditional application to use Centrify and Active Directory Federation Services, you need to customize settings in the `centrifydc_fs.xml` file to identify the resource federation and the application URL that's been defined for the application in the resource federation server. By placing this file in an application's `WEB-INF` directory, you can control these custom settings on an application-by-application basis.

To customize the `centrifydc_fs.xml` file for an application:

1. Copy the default version of the `centrifydc_fs.xml` file from:
   - Linux and UNIX:`/usr/share/centrifydc/java/web/templates`
   - Windows: `C:\Program Files\Centrify\Centrify\java\web\templates`

to the application's `WEB-INF` directory.

2 Open the `centrifydc_fs.xml` file that is in the application's `WEB-INF` directory with a text editing tool.

3 Edit the appropriate sections of the template file to configure authentication through Active Directory Federation Services for the application. For example, modify the following elements in this file:

- Set `federationServerUrl` to the URL of the Active Directory Federation Services resource federation server.
- Set `entryUrl` to the URL for accessing an application. The entryUrl should be exactly the same as the entry URLs you specify when you add the Centrify sample applications to the resource federation server.
- Set the attributes in the `RoleMapping` section to map Active Directory groups and users to the role names defined for an application in its `web.xml` file.

The `centrifydc_fs.xml` template file is used for all Java-based applications. For more information about the `centrifydc_fs.xml` elements and settings defined in this file, see Chapter 9, "Understanding the centrifydc_fs.xml file."

4 Save your changes and close the file.

For further examples of customized `web.xml` and `centrifydc_fs.xml` files, see the Centrify sample applications in the `$CATALINA_HOME/webapps/adfs-*` directories.

Working with claims-aware applications

To handle claims and support Active Directory Federation Services, Centrify includes APIs that enable an application to query for claim information, query SAML information, obtain raw SAML tokens, and control log-on and log-off operations.

Use the following procedure to make a claims-aware application:

1 Copy the Centrify SAML JSP Tag library, `centrifydc_fs_taglib.jar`, from the JBOSS_HOME/`server/`*`myserver`*`/lib` directory to the application's `WEB-INF/lib` directory.

2 Open the application's `web.xml` file and add Centrify SAML as a servlet filter. The SAML filter intercepts requests to the application that match the URL pattern you specify and enables the processing of AD FS messages for the application.

For example:

```
<filter>
<filter-name>saml</filter-name>
<filter-class>com.centrify.fs.SamlFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>saml</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

3 Use the tags and attributes defined in the SAML JSP Tag library to make your application understand and respond to SAML-based claims.

You can then use the tags and attributes defined in the file `centrifydc_fs_taglib.jar` file to make your application understand and respond to SAML-based claims. The `aware.jsp` file in the `adfs-claims-aware` and `adfs-ordering` sample applications illustrate how to configure claims-aware applications. For reference information about the Centrify SAML JSP tags and attributes see Chapter 8, "Developing claims-aware Java EE applications for Centrify."

Configuring a WebLogic Server for AD FS

At this juncture you should have deployed and confirmed the proper installation and configuration of the Centrify package for Active Directory authentication and completed the optional server configuration procedures (for example, run as a Windows service). See the *Centrify Authentication Guide for Java Applications* for those instructions.

This chapter serves two purposes:

- Finish WebLogic configuration: This section tells you how to finish up WebLogic configuration to use Centrify and AD FS for user authentication.
- Configuring WebLogic applications to use AD FS: This section describes how to modify Java EE applications running on WebLogic servers to use Centrify and AD FS for authentication.

If you used the configure.pl option 0 (runs all of the configure.pl options) to install and configure the Centrify DirectControl suite package proceed to Finish WebLogic configuration to configure the sample applications to work in your AD FS environment. Once you complete this procedures go to Chapter 7, "Add sample applications and verify configuration" to run the sample applications to confirm your set up.

If you chose the manual configuration option instead of configure.pl option 0, proceed to Finish WebLogic configuration and perform the procedures in that section. Then go to Chapter 7, "Add sample applications and verify configuration" to run the sample applications to confirm your set up.

After you have completed the sample applications testing, return to Configuring WebLogic applications to use AD FS in this chapter to learn how to modify your applications to use AD FS.

Finish WebLogic configuration

This section is composed of the following procedures. You may not need to perform them all to complete the configuration:

- Configure sample applications to work in your AD FS environment
- Configure SSL
- Import AD FS resource server certificate
- Creating a validation certificate
- Configuring a Proxy Server for WebLogic

Configure sample applications to work in your AD FS environment

In this step, you customize the centrifydc_fs.xml for each sample application to work with your AD FS configuration.

When you installed the Centrify sample applications, you created a separate directory on the Web server for the following sample applications:

```
centrifycd-main.war
centrifydc-kerberos.war
centrifydc-ntlm.war
centrifydc-basic.war
centrifydc-form.war
adfs-traditional.war
adfs-claims-aware.war
adfs-ordering.war
```

For the adfs-traditional.war, adfs-claims-aware.war, and adfs-ordering.war samples only, you need to edit each application's `WEB-INF/centrifydc_fs.xml` and replace the following keywords.

- Replace `APP_SERVER_HOST` with fully qualified name domain name of your WebLogic server.
- Replace `7002` with your WebLogic domain SSL port.
- Replace `ADFS_SERVER_HOST` with the fully qualified domain name of your AD FS resource server.

- Replace 443 with SSL port of your AD FS resource server.

Restart the server.

If you used configure.pl option 0 to install and configure Centrify DirectControl suite package, this completes the WebLogic configuration. Go to Chapter 7, "Add sample applications and verify configuration" to run the sample applications to confirm your set up.

If you did Centrify DirectControl suite package installation and configuration manually proceed with the remaining configuration instructions.

Configure SSL

If you are using Centrify for AD FS authentication and are using Sun JDK 6 version 19, IBM JDK 6 refresh 7, or HP JDK 6.0.07 or higher, the TLS/SSL renegotiation option must be enabled for SSL communication with the AD FS server.

Use the following steps to enable the option:

On Linux and UNIX systems:

1. Navigate to your WebLogic domain
2. Copy the file

   `/usr/share/centrifydc/java/web/scripts/weblogic91/startCentrify.sh`

   to your WebLogic domain
3. Edit `startCentrify.sh`
4. Find the line: `export JAVA_OPTIONS`
5. Add the following line **just before** that line:

   For Sun's and HP's JDK:

   ```
   JAVA_OPTIONS="$JAVA_OPTIONS
   -Dsun.security.ssl.allowUnsafeRenegotiation=true"
   ```

   For IBM's JDK:

   ```
   JAVA_OPTIONS="$JAVA_OPTIONS -Dcom.ibm.jsse2.renegotiate=ALL"
   ```

On a Windows servers:

1. Navigate to your Weblogic domain
2. If the weblogic server is running as a windows service, stop the service and uninstall it by running the following command:

   ```
   WL_HOME\server\bin\beasvc -remove svcname: "beasvc
   MyDomain_myServer"
   ```

3. Edit `startCentrify.bat` and `installCentrifySvc.bat`.
4. Find the line that starts with `set JAVA_OPTIONS=...`
5. Add the following line **just after** that line:

   For Sun's and HP's JDK:

   ```
   JAVA_OPTIONS=$JAVA_OPTIONS
   -Dsun.security.ssl.allowUnsafeRenegotiation=true
   ```

   For IBM's JDK:

   ```
   JAVA_OPTIONS=$JAVA_OPTIONS -Dcom.ibm.jsse2.renegotiate=ALL
   ```

6. Re-install the Windows service by running `installCentrifySvc.cmd`.

See the links below for more information:

http://java.sun.com/javase/javaseforbusiness/docs/TLSReadme.html

http://www-01.ibm.com/support/docview.wss?uid=swg21415499

http://blogs.technet.com/b/srd/archive/2010/08/10/ms10-049-an-inside-look-at-cve-2009-3555-the-tls-renegotiation-vulnerability.aspx

http://docs.hp.com/en/JDKJRE60RN/jdk_rnotes_6.0.07.html#whatsnew

Import AD FS resource server certificate

The WebLogic server must trust the Certificate Authority (CA) that issued the AD FS resource server's certificate used for SSL communications. You do this by loading the AD FS resource server's certificate in your application server's `cacerts` keystore. Before you can import the certificate you must export it from the AD FS resource server as a binary DER encoded (`.cer`) file and copy the file to your WebLogic server.

Note If you are using a Microsoft-based AD FS resource server, WebLogic does not support some digital certificate algorithms, including the 1.3.14.3.2.29 - SHA1 with RSA signature algorithm used by the Microsoft SelfSSL utility to install a self-signed certificate on a server. In other words, if your AD FS resource server at the other end of the SSL communications has a SelfSSL-generated certificate, the authentication protocol will fail. See Creating a validation certificate which follows immediately to create a certificate authority and then generate a certificate WebLogic trusts.

Run the following JDK `keytool` command to import the CA certificate into your Tomcat server's `cacerts` trusted keystore. You may need root permission if the `JDK_HOME/jre/lib/security/cacerts` file is owned by root.

```
JAVA_HOME/jre/bin/keytool -import -keystore
JAVA_HOME/jre/lib/security/cacerts -file <your-exported-
CA-certificate-file> -alias <a-unique-name-for-this-ca>
```

The `keytool` command prompts you for a password for the `cacerts` keystore. If you have never changed it, the default keystore password is `changeit`.

The `keytool` command prompts you for a password for the cacerts keystore. If you have never changed the password for the cacerts keystore, the default password is `changeit`.

If your WebLogic server is a cluster, run the `keytool` command on every system in the cluster. See the cluster configuration appendix in the *Authentication Guide for Java Applications* for information about setting up the WebLogic server in a cluster.

Note If the certificate host name does not match the expected name, for example, if the certificate does not use fully-qualified name for the server, you may need to add the following option to the startup script:

```
-Dweblogic.security.SSL.ignoreHostnameVerification=true
```

Creating a validation certificate

AD FS requires the web application server to communicate using the SSL protocol. This requires the application server to have a valid certificate from the AD FS resource server in its keystore.

Note Use this procedure only if you are setting up AD FS for testing in a lab or evaluation environment. For production environments, use more a reliable certificate authority to generate the AD FS resource server certificate.

Creating the Certificate Authority

To begin, you need to create a certificate authority on the AD FS server. To create a Certificate Authority for the AD FS server:

1. Log on to the AD FS server using the Administrator account and password.
2. Click **Start > Control Panel > Add or Remove Programs**.
3. Click **Add/Remove Windows Components**.
4. Select **Certificate Services**, then click **Next**.
5. Select **Enterprise root CA**, then click **Next**.
6. Type a **Common name** for identifying the certificate authority, for example, you may want to use a server name or company name, set the validity period for this certificate authority, then click **Next**.
7. Review the database settings for the certificate authority, then click **Next**.

   Note If the Internet Information Service (IIS) is currently running, you may be prompted to stop the service. You may also be prompted to enable Active Server Pages for the IIS server.
8. Click **Finish**.

Generate a certificate

Use the following steps to create a valid certificate for the default Web site on the AD FS server to work with WebLogic:

1. On the AD FS server, click **Start > Administrative Tools > Internet Information Services (IIS) Manager**.
2. Expand **Web Sites**, select **Default Web Site** and right-click, then select **Properties**.

3 Click the **Directory Security** tab, then click **Server Certificate**.

4 On the welcome page, click **Next**.

5 Select **Create a new certificate**, then click **Next**.

6 Select **Send the request immediately to an online certification authority**, then click **Next**.

7 Type a name for the new certificate, then click **Next**.

8 Type the name of the organization and your organization unit, for example, type the company name and department or division, then click **Next**.

9 Type the Common DNS name for the site, then click **Next**.

10 Select a Country/Region, then type the names of your state or province and city or locality, then click **Next**. You must use the full state and city names with no abbreviations

11 Type the port number to use for SSL connections, then click **Next**. In most cases, you should use the default port of 443.

12 Select the resource server as the Certificate Authority, then click **Next**.

13 Review the information, then submit the request by clicking **Next**.

14 Click **Finish**.

15 Click **View Certificate** to verify the server certificate.

16 Click the **Details** tab, then click **Copy to File** to export the certificate to a file. Click through the wizard to create the file, then click **OK** to close the Properties dialog box. For example, copy the certificate to a file name `ice-fs-cert.cer`.

17 After you create the certificate file, copy it to the WebLogic server (see "Import AD FS resource server certificate" on page 64 for the instructions).

Configure AD FS Authentication Provider

You configured the WebLogic Default and Centrify Authentication Providers in *Authentication for Java Applications* for Active Directory authentication (`CentrifyDCADAuthenticator`). AD FS uses a separate authentication provider. Use the following procedure to configure the authentication provider for AD FS.

1. Change to the directory that contains the WebLogic domain with which you want to work. For example:

   ```
   cd C:\Oracle\middleware\user_projects\domains\mydomain
   ```

2. Run the `startweblogic.sh` script on Linux and UNIX systems or `startweblogic.cmd` script on Windows systems to start up the WebLogic server with appropriate classpaths and Java options for Centrify. Allow time for the server to start.

3. Open a Web browser and go to the WebLogic console. For example:

   `http://`*`fully_qualified_host_name`*`:7001/console`

4. Type the username and password for the WebLogic Administrator account.

5. In the navigation pane, click **Security Realms**, then click **myrealm**. Click the **Providers** tab.

6. Click **Lock & Edit** in the navigation pane.

7. Click **DefaultAuthenticator**.

   In **Control Flag**, select **SUFFICIENT** then click **Save**

8. Configure the `CentrifyDCADFSAuthenticator`:

   Back in the navigation pane, click **Security Realms**, then click **myrealm** and click the **Providers** tab.

   - Click **New**
   - In the **Name** field, enter a unique name for the AD FS authentication provider, for example, `Centrify ADFS Authenticator`.
   - In the **Type** field select **CentrifyDCADFSAuthenticator** and click **OK**.

- Click the name you entered in the **Name** field. Then, in the **Control Flag** field, select **SUFFICIENT** and click **Save**
- In the navigation pane, click **Activate Changes**

9 Run the appropriate command for the local operating environment to stop the WebLogic domain.

Unless you need to configure a proxy server for WebLogic, this concludes the WebLogic server and domain configuration for AD FS. Go to Chapter 7, "Add sample applications and verify configuration" to add the sample applications to AD FS and run them to confirm your configuration.

Configuring a Proxy Server for WebLogic

If a proxy server is required for the WebLogic server to communicate with the ADFS server, set the `JAVA_OPTIONS` environment variable to the proxy host and port before starting the WebLogic server, as follows:

On Linux and UNIX systems:

1 In the WebLogic domain, edit `startCentrify.sh`.

2 Find the following line: `export JAVA_OPTIONS`

3 Add the following definition to `JAVA_OPTIONS` **just before** that line:

```
JAVA_OPTIONS="$JAVA_OPTIONS
-Dhttps.proxyHost=proxyhost -Dhttps.proxyPort=proxyport"
```

where *`proxyhost`* is the proxy server host name and *`proxyport`* is the proxy server port number.

4 Restart the WebLogic server.

On Windows servers:

1 In the WebLogic domain, edit both of the following files:

- `startCentrify.cmd`
- `installCentrifySvc.cmd`

2 Find the line that starts with `set JAVA_OPTIONS`

3 Add the following definition to `JAVA_OPTIONS` **just after** that line

```
set JAVA_OPTIONS=%JAVA_OPTIONS%
-Dhttps.proxyHost=proxyhost
-Dhttps.proxyPort=proxyport
```

where *proxyhost* is the proxy server host name and *proxyport* is the proxy server port number.

4 Restart the WebLogic server.

This concludes WebLogic server and domain configuration for AD FS. Go to Chapter 7, "Add sample applications and verify configuration" to add the sample applications to AD FS and run them to confirm your configuration.

Configuring WebLogic applications to use AD FS

The sample applications are preconfigured to use Centrify and AD FS for authentication. This section describes how to modify Java EE applications to use AD FS. There are two types of Java applications:

- Traditional: Traditional applications do use Active Directory Federation Services claims directly. Instead they rely on standard Java EE APIs andJava EE security constraints defined in an application's `web.xml` file to authenticate users.

  For traditional applications, you add a context file modify its web.xml file and add the centrifydc-fs.xml file to its `WEB-INF` directory.

- Claims-aware: Claims-aware applications are applications that comply with Security Assertion Markup Language (SAML) and WS-Federation standards for authorization messages. Because these applications are specifically written or modified to recognize the content and format of Active Directory Federation Service claims, Centrify validates and passes along any verified claims from the client to the application. The application then decides the level of service to provide the client based directly on those claims. If the application needs claims and none are present, it redirects to AD FS to get claims.

  For claims-aware applications you add a servlet filter to the web.xml file and copy the Centrify SAML JSP Tag library, `centrifydc_fs_taglib.jar` to the application's `WEB-INF/lib` directory.

Working with traditional applications

Traditional applications do not take advantage of Active Directory Federation Services claims directly. Instead they rely on standard Java EE APIs, the security constraints defined in an application's `web.xml`, and an application-specific configuration file, `centrifydc_fs.xml`, placed in the application's `WEB-INF` directory. A template version of this file is installed by default in the the following directories

- On Linux and UNIX: `/usr/share/centrifydc/java/web/templates`
- On Windows: C`:\Program Files\Centrify\Centrify\java\web`

Each Centrify sample applications includes its own copy of this file.

Configuring traditional WebLogic applications that use the standard Java EE APIs to use Centrify and Active Directory Federation Services involves the following steps:

1 Adding the SAML filter to web.xml

2 Adding the SamlAuthServlet to web.xml

3 Setting the authentication method in web.xml

4 Configuring the security constraint in web.xml

5 Modifying settings in centrifydc_fs.xml

6 Mapping roles to claims in weblogic.xml

7 Configuring SamlAuthFilter in weblogic.xml

8 Adding jar files to your traditional application

9 Adding jar files to your claims-aware application

Adding the SAML filter to web.xml

For applications that use the standard Java EE APIs, you need to modify the application's `web.xml` file to include the SAML filter. The SAML filter intercepts requests to the application that match the URL pattern you specify and enables the processing of AD FS messages for the application.

To add the SAML filter to WebLogic applications:

1 Open the application's `web.xml` file

2 Add the following lines to install the Centrify SAML filter:

```
<filter>
<filter-name>saml</filter-name>
<filter-class>com.centrify.fs.SamlFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>saml</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

Adding the SamlAuthServlet to web.xml

For each application, add the SamlAuthServlet to `web.xml` to intercept requests and use AD FS for authentication.

When you add the SamlAuthServlet, map it to the `/adfs url pattern` as follows.

```
…
<servlet>
<servlet-name>adfs</servlet-name>
<servlet-class>com.centrify.fs.weblogic.AuthServlet</servlet-
class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>adfs</servlet-name>
<url-pattern>/adfs</url-pattern>
<servlet-mapping>
…
```

Setting the authentication method in web.xml

For each application, you need to modify the `web.xml` file to define the authentication method in the `<login-config>` element. For Java EE AD FS traditional applications set your `<login-config>` as follows:

```
...
<login-config>
<auth-method>FORM</auth-method>
<form-login-config>
<form-login-page>/adfs</form-login-page>
```

```
<form-error-page>/error.jsp</form-error-page>
</form-login-config>
</login-config>
...
```

Configuring the security constraint in web.xml

For each application, you need to modify the `web.xml` file to define the security constraints for the application.

To modify the security constraints for an application edit the `<security-constraint>` and `<auth-constraint>` sections as appropriate to your application. For example:

```
...
<security-constraint>
<web-resource-collection>
<web-resource-name>ProtectedResource</web-resource-name>
<url-pattern>/*</url-pattern>
<http-method>GET</http-method>
<http-method>POST</http-method>
</web-resource-collection>
<auth-constraint>
<role-name>user</role-name>
</auth-constraint>
</security-constraint>
...
```

Modifying settings in centrifydc_fs.xml

To configure a traditional application to use Centrify and Active Directory Federation Services, you need to customize settings in the `centrifydc_fs.xml` file to identify the resource federation and the application URL that's been defined for the application in the resource federation server. By placing this file in an application's `WEB-INF` directory, you can control these custom settings on an application-by-application basis.

To customize the centrifydc_fs.xml file for an application:

1. Copy the default version of the `centrifydc_fs.xml` file to the application's `WEB-INF` directory. For example:

   ```
   cd mysample.war/WEB-INF
   cp /usr/share/centrifydc/java/web/templates/centrifydc_fs.xml
   ```

2 Open the `centrifydc_fs.xml` file that is in the application's `WEB-INF` directory with a text editing tool.

3 Set the `federationServerUrl` to the URL of the AD FS resource server.

4 Set the `entryUrl` to the URL for accessing the application. The `entryUrl` should be exactly the same as the entry URL you specify when you add the Centrify sample applications to the resource federation server.

5 Set the attributes in the `<RoleMapping>` section to map Active Directory groups and users to the role names defined for an application in its `web.xml` file.

6 Save your changes and close the file.

In addition to these settings, you can also use the `centrifydc_fs.xml` file to define other aspects of your environment. For more information about the other elements and settings defined in this file, see Chapter 9, “Understanding the centrifydc_fs.xml file.”

Mapping roles to claims in weblogic.xml

WebLogic role names are mapped to ADFS group claims through settings in the `weblogic.xml` file. In addition to other settings, the `weblogic.xml` file allows you to specify how the WebLogic roles names should map to ADFS Identity claims (authenticated user principals).

To map group claims to WebLogic roles:

1 Open or create the `weblogic.xml` file in the application’s `WEB-INF` directory

2 Add or edit the `<security-role-assignment>` section of the `weblogic.xml` file to define how WebLogic roles are mapped to group claims. Within this section, you specify how a WebLogic role-name should map to a group claim or identity claim (the authenticated user's name).

   For example, to map the WebLogic role of `admin` to the AD FS group claim `WebAdmins` add a section similar to the following in the `weblogic.xml` file:

```
...
<weblogic-web-app>
```

```
<security-role-assignment>
<role-name>
admin
</role-name>
<principal-name>
WebAdmins
</principal-name>
</security-role-assignment>
</webloc-web-app>
...
```

Configuring SamlAuthFilter in weblogic.xml

In addition to mapping roles to group names in `weblogic.xml` you must also configure the SamlAuthFilter in `weblogic.xml`. To configure the SamlAuthFilter, add the following after the `</security-role-assignment>` in `weblogic.xml`:

```
<auth-filter>com.centrify.fs.weblogic.SamlAuthFilter</auth-
filter>
```

See the `web.xml`, `weblogic.xml` and `centrifydc_fs.xml` files for each of the Centrify sample applications for more examples.

Adding jar files to your traditional application

If your WebLogic server already contains a different version of a jar files required by Centrify in the WebLogic startup script, you need to copy the correct version of the file from the Centrify package to the `WEB-INF/lib` directory of your application.

Copy the following jar files to your applications' `WEB-INF/lib` directory:

Linux and UNIX:

```
/usr/share/centrifydc/java/weblib/centrifydc_common.jar
/usr/share/centrifydc/java/weblib/centrifydc_fs.jar
/usr/share/centrifydc/java/weblib/centrifydc_fs_taglib.jar
/usr/share/centrifydc/java/weblib/weblogic91/
centrifydc_fs_weblogic_9.1.jar
/usr/share/centrifydc/java/weblib/ext/jstl.jar
/usr/share/centrifydc/java/weblib/ext/standard.jar
/usr/share/centrifydc/java/weblib/ext/xmldsig.jar
/usr/share/centrifydc/java/weblib/ext/xmlsec.jar
```

Windows:

```
C:\Program
Files\Centrify\Centrify\java\web\lib\centrifydc_common.jar
C:\Program
Files\Centrify\Centrify\java\web\lib\centrifydc_fs.jar
C:\Program Files\Centrify\Centrify\java\web\lib\
centrifydc_fs_taglib.jar
C:\Program Files\Centrify\Centrify\java\web\lib\weblogic91\
centrifydc_fs_weblogic_9.1.jar
C:\Program Files\Centrify\Centrify\java\web\lib\ext\jstl.jar
C:\Program
Files\Centrify\Centrify\java\web\lib\ext\standard.jar
C:\Program
Files\Centrify\Centrify\java\web\lib\ext\xmldsig.jar
C:\Program Files\Centrify\Centrify\java\web\lib\ext\xmlsec.jar
```

Working with claims-aware applications

Claims-aware applications are applications that comply with Security Assertion Markup Language (SAML) and WS-Federation standards for authorization messages. Because these applications are specifically written or modified to recognize the content and format of Active Directory Federation Service claims, Centrify validates and passes along any verified claims from the client to the application. The application then decides the level of service to provide the client based directly on those claims. If the application needs claims and none are present, it redirects to the Resource Federation Server to get claims.

To handle claims and support Active Directory Federation Services, Centrify includes APIs that enable an application to query for claim information, query SAML information, obtain raw SAML tokens, and control log-on and log-off operations.

To make an application claims-aware:

1. Copy the Centrify SAML JSP Tag library, `centrifydc_fs_taglib.jar` file to the application's `WEB-INF/lib` directory.

2. Use the tags and attributes defined in the SAML JSP Tag library to make your application understand and respond to SAML-based claims.

   Note The `aware.jsp` file in the `adfs-claims-aware` and `adfs-ordering` sample applications illustrate how to use the tags to configure

claims-aware applications. For reference information about the Centrify SAML JSP tags and attributes, see Chapter 8, "Developing claims-aware Java EE applications for Centrify."

3 Add the Centrify SAML filter to the applications' `web.xml`. The SAML filter is used to intercept requests to the application URLs matching the pattern you specify. This filter enables the processing of SAML-based ADFS messages for the application. For example:

```
...
<filter>
<filter-name>saml</filter-name>
<filter-class>com.centrify.fs.SamlFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>saml</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

Adding jar files to your claims-aware application

If your WebLogic server already contains a different version of a jar files required by Centrify in the WebLogic startup script, you need to copy the correct version of the file from the Centrify package to the `WEB-INF/lib` directory of your application.

Copy the jar files from the following Linux, UNIX, or Windows directory to your applications' `WEB-INF/lib` directory:

Linux and UNIX: `/usr/share/centrifydc/java/web`

Windows: `C:\Program Files\Centrify\Centrify\java\web`

(The jar files for claims-aware applications are the same as for traditional applications. See Adding jar files to your traditional application for the list.)

Configuring WebSphere for AD FS

At this juncture you should have deployed and confirmed the proper installation and configuration of the Centrify package for Active Directory authentication and completed the optional server configuration procedures (for example, run as a Windows service). See the *Centrify Authentication Guide for Java Applications* for those instructions.

This chapter serves two purposes:

- Finish WebSphere configuration: This section tells you how to finish up WebSphere configuration to use Centrify and AD FS for user authentication.
- Configuring WebSphere applications for AD FS: This section describes how to modify Java EE applications running on WebSphere servers to use Centrify and AD FS for authentication.

If you used the configure.pl option 0 (runs all of the configure.pl options) to install the Centrify DirectControl suite package AND you verified proper installation, there are no more configuration procedures. You can skip to Chapter 7, "Add sample applications and verify configuration" to run the sample applications to confirm your AD FS set up.

If you chose the manual configuration option instead of configure.pl option 0, proceed to Finish WebSphere configuration and perform the procedures in that section. Then go to Chapter 7, "Add sample applications and verify configuration" to run the sample applications to confirm your set up.

After you have completed the sample applications testing, return to Configuring WebSphere applications for AD FS in this chapter to learn how to modify your applications to use AD FS.

Finish WebSphere configuration

This section is composed of the following procedures. You may not need to perform them all to complete the configuration:

- Configure sample applications for AD FS
- Add the Centrify AD FS Trust Association Interceptor
- Configure SSL settings
- Configuring a proxy server for WebSphere

Configure sample applications for AD FS

In this step, you customize the centrifydc_fs.xml for each sample application to work with your AD FS configuration.

When you deployed the Centrify sample applications in centrifydc-samples.ear, the WebSphere Application Server exploded the following sample application in a directory under the profile:

```
centrifydc-main.war
centrifydc-kerberos.war
centrifydc-ntlm.war
centrifydc-basic.war
centrifydc-form.war
adfs-traditional.war
adfs-claims-aware.war
adfs-ordering.war
```

For the adfs-traditional.war, adfs-claims-aware.war, and adfs-ordering.war samples only, you need to edit each application's `WEB-INF/centrifydc_fs.xml` and replace the following keywords.

- Replace `APP_SERVER_HOST` with fully qualified name domain name of your WebSphere server.
- Replace `7002` with your WebSphere server's SSL port.
- Replace `ADFS_SERVER_HOST` with the fully qualified domain name of your AD FS resource server.
- Replace `443` with SSL port of your AD FS resource server.

Restart the server.

Add the Centrify AD FS Trust Association Interceptor

In *Authentication for Java Applications*, you installed the SPNEGO Trust Association Interceptor. To use AD FS for authentication you need to use the Centrify AD FS Trust Association Interceptor for authentication.

To begin, start the WebSphere server if it is not running, open a browser and log on to the WebSphere administration console.

The start of the procedure depends upon your WebSphere version:

- For WebSphere 6.0: Select **Security > Global Security > Authentication mechanisms > LTPA**.
- For WebSphere 6.1: Select **Security > Secure administration, applications, and infrastructure > Web security > Trust association**.
- For WebSphere 7.0:Select S**ecurity > Global security> Web and SIP security > Trust association**. Check **Enable trust association**. Click **OK** and then click the **Save** link at the top of the page.

Then proceed with the following steps:

1. Click Interceptors, then click **New**.
   - In **Interceptor class name**, type `com.centrify.fs.was.CentrifyFSTAI`.
   - Click **OK**, then click **Save** at the top of the page and **Save** again in any subsequent pages.
2. Return to the Interceptors page, then click **com.centrify.fs.was.CentrifyFSTAI**.
3. Click **Custom Properties**, then click **New**.
   - For the **Name**, type **targetURI**.
   - For the **Value**, type the URI pattern for each Java EE traditional application that you want to authenticate using AD FS. Separate entries with a space.

To run the AD FS sample that uses Java EE traditional authentication you need to enter the following:

```
/centrifydc-samples/ADFS-traditional/*
```

Note targetURI is one of several custom properties supported. The table on page page 86 describes the targetURI property and the optional custom properties.

4 Click **OK**, then click **Save** at the top of the page and **Save** again in any subsequent pages.

5 Restart the WebSphere application server to enable the configuration changes before deploying applications that rely on Centrify DirectControl Active Directory or Active Directory Federation Services for authentication.

At this point, if you have not applications configured to use SPNEGO authentication, leave the field blank and add valued later when you deploy application that use SPNEGO authentication

Note If your WebSphere application server is a cluster, be sure to synchronize the nodes in the cluster after making configuration changes. Restart all node agents on the managed server by executing the following command in each node:

```
/opt/IBM/WebSphere/AppServer/profiles/xxxx/bin/startNode.
sh
```

Custom properties for CentrifyFSTAI

Property	Description
targetURI	Required. List of URI patterns of Java EE traditional applications using AD FS for authentication. Separate entries with a space. The URI pattern is based on `glob` patterns similar to those used in Linux and UNIX shells for file filters. Leave this property blank if you have no Java EE AD FS traditional applications.
configFile	Optional. A global `centrifydc_fs.xml` file to use for all Java EE traditional AD FS applications. If this property is not set, each application must have a `centrifydc_fs.xml` file in its `WEB-INF` directory to use for authentication.
ignoreCase	Optional. Ignore case when matching the `targetURI` patterns. If not set, the default is `true`.
useShortName	Optional. After a user has been authenticated, set the user's short name (without the domain) as the authenticated user name. For example, if this property is true, the authenticated user name for a user with a `UPN` of `username@domain.com` is `username`. If this property is not set, the default is `false`.

Configure SSL settings

When you use AD FS, you must configure your WebSphere application server to trust the Certificate Authority (CA) that issued your AD FS resource server SSL certificate. You do this by loading the AD FS resource server's certificate into the WebSphere application server's `cacert` keystore. Go to Import AD FS resource server certificate for the instructions.

Note If the WebSphere application server is installed on a cluster, you need to have the AD FS SSL certificate on every computer in the cluster.

In addition, if you are using Centrify for AD FS authentication and are using Sun JDK 6 version 19, IBM JDK 6 refresh 7, or HP JDK 6.0.07 or higher, the TLS/SSL renegotiation option must be enabled for SSL communication with the AD FS server.

Use the following steps to enable the option:

On Linux and UNIX systems:

1 Navigate to the WebSphere server directory and edit the bin/startServer.sh file

2 Look for the line that starts with `$JAVA_HOME"/bin/java \`

3 Add the following line **just before** that line:

For IBM's JDK:

```
JVM_EXTRA_CMD_ARGS="$JVM_EXTRA_CMD_ARGS -
Dcom.ibm.jsse2.renegotiate=ALL"
```

For Sun's and HP's JDK:

```
JVM_EXTRA_CMD_ARGS="$JVM_EXTRA_CMD_ARGS
-Dsun.security.ssl.allowUnsafeRenegotiation=true
```

4 Restart the WebSphere server.

On Windows systems:

1 Navigate to the WebSphere server directory and edit the bin/startServer.bat file

2 Look for the line that starts with `set CLASSPATH=%WAS_CLASSPATH%`

3 Add the following line **just after** that line:

For IBM's JDK:

```
SET USER_INSTALL_PROP=%USER_INSTALL_PROP% -
Dcom.ibm.jsse2.renegotiate=ALL
```

For Sun's and HP's JDK:

```
USER_INSTALL_PROP=%USER_INSTALL_PROP%
-Dsun.security.ssl.allowUnsafeRenegotiation=true
```

4 Restart the WebSphere server

See the links below for more information:

http://java.sun.com/javase/javaseforbusiness/docs/TLSReadme.html

http://www-01.ibm.com/support/docview.wss?uid=swg21415499

http://blogs.technet.com/b/srd/archive/2010/08/10/ms10-049-an-inside-look-at-cve-2009-3555-the-tls-renegotiation-vulnerability.aspx

http://docs.hp.com/en/JDKJRE60RN/jdk_rnotes_6.0.07.html#whatsnew

Import AD FS resource server certificate

Before you can import the certificate, you must export it export it from the AD FS resource server system as a binary DER encoded file (`.cer`) and copy the file to your WebSphere server.

To import the `*.cer` file into the JDK `cacerts` keystore using JDK commands:

1. Log on the WebSphere application server and use the JDK `keytool` command to import the CA certificate to the JDK `cacerts` keystore. You may need root permission if the `JDK_HOME/jre/lib/security/cacerts` file is owned by root.

   ```
   JDK_HOME/jre/bin/keytool -import -keystore
   JDK_HOME/jre/lib/security/cacerts -file
   <your-exported-CA-certificate-file> -alias
   <a-unique-name-for-this-ca>
   ```

2. Type the password for the `cacerts` keystore. If you have never changed the password for the `cacerts` keystore, the default password is `changeit`.

   For WebSphere application server 6.1 in a clustered environment, it is also necessary to import the ADFS resource server CA certificate to the `NodeDefaultTrustStore` (or to the `CellDefaultTrustStore` for WebSphere Network Deployment Server).

To import the certificate from the WebSphere administration console:

1. Navigate to **Security > SSL certificate and key management**.
2. Under **Related Items**, click **Key stores** and **certificates**.

3 Click **NodeDefaultTrustStore** (or **CellDefaultTrustStore** for WebSphere Network Deployment Server).

4 Under Additional Properties, click **Signer certificates**. Then click **Add**.

5 In **Alias**, enter a unique name for the ADFS resource server CA certificate.

6 In **Filename**, enter the path of the .cer certificate file.

7 Under **Data type** select Binary DER data from the pull down list.

8 Click **OK**, then click **Save** at the top of the page and **Save** again in any subsequent pages.

Note After configuring your WebSphere application server, you must restart the WebSphere application server for the configuration changes to take effect and for the sample applications to work.

Unless you need to configure a proxy server for WebSphere, this concludes the final WebSphere configuration steps to support AD FS authentication. Go to Chapter 7, "Add sample applications and verify configuration" to add the sample applications to AD FS and run them to confirm your configuration.

Configuring a proxy server for WebSphere

If a proxy server is required for the WebSphere server to communicate with the AD FS server, set the `JAVA_OPTIONS` environment variable to the proxy host and port before starting the WebSphere server, as follows:

On Linux and UNIX systems:

1 Edit `WAS_HOME/bin/startServer.sh (where WAS_HOME` is the base directory for the WebSphere installation).

2 Find the line that starts with "$JAVA_HOME"/bin/java \

3 Add the following line **just before** that line:

```
JVM_EXTRA_CMD_ARGS="$JVM_EXTRA_CMD_ARGS -DproxyHost=proxyhost
-DproxyPort=proxyport"
```

where *`proxyhost`* is the proxy server host name and *`proxyport`* is the proxy server port number.

4 Restart the WebSphere server.

On Windows systems:

1 In the WebSphere domain, edit the `startServer.bat` file

2 Find the line that starts with `set CLASSPATH=%WAS_CLASSPATH%`

3 Add the following line **just after** that line

```
SET USER_INSTALL_PROP=%USER_INSTALL_PROP% -
DproxyHost=proxyhost
-DproxyPort=proxyport
```

where *`proxyhost`* is the proxy server host name and *`proxyport`* is the proxy server port number.

4 Restart the WebSphere server.

This concludes WebSphere configuration for AD FS. Go to Chapter 7, "Add sample applications and verify configuration" to add the sample applications to AD FS and run them to confirm your configuration.(Note that the Centrify sample applications for AD FS testing were installed when you installed the Active Directory sample applications in the *Centrify for Web Applications Authentication Guide for Java Applications*.)

Configuring WebSphere applications for AD FS

The sample applications are preconfigured to use Centrify and AD FS for authentication. This section describes how to modify Java EE applications to use AD FS. There are two types of Java applications:

- Traditional: Traditional applications do use Active Directory Federation Services claims directly. Instead they rely on standard Java EE APIs andJava EE security constraints defined in an application's `web.xml` file to authenticate users.

  For traditional applications, you add a context file modify its web.xml file and add the centrifydc-fs.xml file to its `WEB-INF` directory.

- Claims-aware: Claims-aware applications are applications that comply with Security Assertion Markup Language (SAML) and WS-Federation standards for authorization messages. Because these applications are specifically written or modified to recognize the

content and format of Active Directory Federation Service claims, Centrify validates and passes along any verified claims from the client to the application. The application then decides the level of service to provide the client based directly on those claims. If the application needs claims and none are present, it redirects to AD FS to get claims.

For claims-aware applications you add a servlet filter to the web.xml file and copy the Centrify SAML JSP Tag library, `centrifydc_fs_taglib.jar` to the application's `WEB-INF/lib` directory.

Working with traditional applications

Traditional applications do not take advantage of Active Directory Federation Services claims directly. Instead they rely on standard Java EE APIs, the security constraints defined in an application's `web.xml`, and an application-specific configuration file, `centrifydc_fs.xml`, placed in the application's `WEB-INF` directory. A template version of this file is installed by default in the the following directories

```
/usr/share/centrifydc/java/web/templates
C:\Program Files\Centrify\Centrify\java\web
```

Each Centrify sample applications includes its own copy of this file.

Configuring traditional WebSphere applications that use the standard Java EE APIs to use Centrify and Active Directory Federation Services involves the following steps:

- Adding the SAML filter to web.xml
- Setting the authentication method in web.xml
- Configuring the security constraint in web.xml
- Modifying settings in centrifydc_fs.xml
- Adding the application URI to the Centrify AD FS trust association interceptor

Adding the SAML filter to web.xml

For applications that use the standard Java EE APIs, you need to modify the application's `web.xml` file to include the SAML filter. The SAML filter intercepts requests to the application that match the URL pattern you specify and enables the processing of AD FS messages for the application.

To add the SAML filter to WebSphere applications:

1. Open the application's `web.xml` file
2. Add the following lines to install the Centrify SAML filter:

```
<filter>
<filter-name>saml</filter-name>
<filter-class>com.centrify.fs.SamlFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>saml</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

Setting the authentication method in web.xml

For each application, you need to modify the `web.xml` file to define the authentication method in the `<login-config>` element. For Java EE AD FS traditional applications set your `<login-config>` as follows:

```
...
<login-config>
<auth-method>FORM</auth-method>
<form-login-config>
<form-login-page>/adfs</form-login-page>
<form-error-page>/error.jsp</form-error-page>
</form-login-config>
</login-config>
...
```

Configuring the security constraint in web.xml

For each application, you need to modify the `web.xml` file to define the security constraints for the application.

To modify the security constraints for an application sdit the `<security-constraint>` and `<auth-constraint>` sections as appropriate to your application.

For example:

```
...
<security-constraint>
<web-resource-collection>
<web-resource-name>ProtectedResource</web-resource-name>
<url-pattern>/*</url-pattern>
<http-method>GET</http-method>
<http-method>POST</http-method>
</web-resource-collection>
<auth-constraint>
<role-name>user</role-name>
</auth-constraint>
</security-constraint>
...
```

Modifying settings in centrifydc_fs.xml

To configure a traditional application to use Centrify and Active Directory Federation Services, you need to customize settings in the `centrifydc_fs.xml` file to identify the resource federation and the application URL that's been defined for the application in the resource federation server. By placing this file in an application's `WEB-INF` directory, you can control these custom settings on an application-by-application basis.

To customize the `centrifydc_fs.xml` file for an application:

1. Copy the default version of the `centrifydc_fs.xml` file to the application's `WEB-INF` directory. For example:

   ```
   cd $WAS_HOME/server/default/deploy/mysample.war/WEB-INF
   cp /usr/share/centrifydc/java/web/templates/centrifydc_fs.xml
   ```

2. Open the `centrifydc_fs.xml` file in a text editing tool.

3. Set the `federationServerUrl` to the URL of the AD FS resource server.

4. Set the `entryUrl` to the URL for accessing the application. The field should be exactly the same as the entry URL you specify when you

add the Centrify sample applications to the resource federation server.

5 Set the attributes in the `<RoleMapping>` section to map Active Directory groups and users to the role names defined for an application in its `web.xml` file.

6 Save your changes and close the file.

In addition to these settings, you can also use the `centrifydc_fs.xml` file to define other aspects of your environment. For more information about the other elements and settings defined in this file, see Chapter 9, "Understanding the centrifydc_fs.xml file."

Adding the application URI to the Centrify AD FS trust association interceptor

To add the application URI to the CentrifyFS trust association interceptor:

1 Be certain the WebSphere application is running.

2 Open a browser, enter the WebSphere administration console URL, then login.

3 Go to the CentrifyFSTAI trust association interceptor and do one of the following:

- For WebSphere 6.0, navigate to **Security > Global Security > Authentication mechanisms > LTPA > Trust association > Interceptors**.
- For WebSphere 6.1, navigate to **Security > Secure administration, applications, and infrastructure > Web security > Trust association > Interceptors**.
- For WebSphere 7.0, navigate to **Security > Global Security > Web and SIP security > Trust Association > Interceptors**

4 Click **com.centrify.fs.was.CentrifyFSTAI** and click **Custom properties.**

5 Click **targetURI** and add the URI pattern for your application to the **Value** field, for example:

```
/centrifydc-samples/adfs-traditional/*
```

Use a blank space to separate URI entries. For more information on the `targetURI` property see the table on page 86.

Working with claims-aware applications

Claims-aware applications are applications that comply with Security Assertion Markup Language (SAML) and WS-Federation standards for authorization messages. Because these applications are specifically written or modified to recognize the content and format of Active Directory Federation Service claims, Centrify validates and passes along any verified claims from the client to the application. The application then decides the level of service to provide the client based directly on those claims. If the application needs claims and none are present, it redirects to the Resource Federation Server to get claims.

To handle claims and support Active Directory Federation Services, Centrify includes APIs that enable an application to query for claim information, query SAML information, obtain raw SAML tokens, and control log-on and log-off operations.

To make an application claims-aware:

1. Copy the Centrify SAML JSP Tag library, `centrifydc_fs_taglib.jar` file to the application's `WEB-INF/lib` directory.

2. Use the tags and attributes defined in the SAML JSP Tag library to make your application understand and respond to SAML-based claims.

   The `aware.jsp` file in the `adfs-claims-aware` and `adfs-ordering` sample applications illustrate how to use the tags to configure claims-aware applications. For reference information about the Centrify SAML JSP tags and attributes, see Chapter 8, "Developing claims-aware Java EE applications for Centrify."

3. Add the Centrify SAML filter to the applications' `web.xml`.

   The SAML filter is used to intercept requests to the application URLs matching the pattern you specify. This filter enables the processing of SAML-based ADFS messages for the application.

For example:

```
...
<filter>
<filter-name>saml</filter-name>
<filter-class>com.centrify.fs.SamlFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>saml</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

Add sample applications and verify configuration

This chapter contains the final steps required before you can use the sample applications to verify that AD FS is used to authenticate users for applications running on Tomcat, JBoss, WebLogic, WebSphere or Apache servers. After you verify proper installation, go back to the platform chapter to find out how to configuration your own applications to use AD FS.

DirectControl for Web Applications supports authentication using AD FS 1.0 and AD FS 2.0. That is, your web server can use either protocol to authenticate the user. However, adding the sample applications for AD FS authentication is very different for each environment. The difference is in the procedures you use to add the sample applications to the resource organization, create claims and populate the claims in the account organization. Proceed with the section corresponding to your AD FS version to add the sample application and then go to "Verifying authentication using the sample applications" on page 105 to run them.

After you have successfully run the sample applications, you are done withe Centrify for Web Applications installation for AD FS authentication. The next step is to configure your existing Java applications to

Note At this juncture you should have already deployed the sample applications on your Apache, Tomcat, JBoss, WebLogic or WebSphere server, imported the resource server's certificate on the Web application server, and imported the Web application server's certificate on the resource server.

Add sample applications to AD FS 1.0

Use the following instructions to prepare the AD FS 1.0 environment to authenticate users for the sample applications. The process has the following steps.

1 Adding the sample applications to the resource server

2 Enable UPN on the account server

3 Creating group organization claims on the account server (Optional)

Adding the sample applications to the resource server

Use the following procedure to add each sample application to the resource server.

1 Click **Start > Administrative Tools > Active Directory Federation Services**.

2 Select **Federation Services > Trust Policy > My Organization > Applications**, then right-click, and select **New > Application** to start the Add Application Wizard.

3 At the Welcome page, click **Next**.

4 Select **Claims-aware applications**, then click **Next**.

Regardless of whether the application you are adding is configured as a claims-aware application or as a traditional application, you must always add the application to AD FS 1.0 as a claims-aware application.

5 Type the **Application Name** and the **Application URL**, then click **Next**. The application URL is the entry point into the application. The URL must match the `entryURL` element value in the application's `Centrifydc-fs.xml` file.

If you are using a Tomcat, JBoss, WebLogic or WebSphere server the locations are:

- For the **Traditional** sample application, set the URL to

`https://server:port/centrifydc-samples/adfs-traditional/entry.jsp`

- For the **Claims-aware** sample application, set the URL to

`https://server:port/centrifydc-samples/adfs-claims-aware/entry.jsp`

- For the **Ordering** sample application, set the URL to

`https://server:port/centrifydc-samples/adfs-ordering/entry.jsp`

For `Server:` you must use the fully-qualified domain name for the Web server.

For *port*: use the port number you configured for SSL communications. This is different for each server. The following table lists the default port numbers:

Server	Default port
Tomcat and JBoss	8443
Weblogic	7002
WebSphere	9443

If you are using an Apache server, the URLs are

```
https://server/samples/adfs-traditional/entry.cgi
https://server/samples/adfs-claims-aware/entry.cgi
https://server/samples/adfs-ordering/entry.cgi
```

For *server*: you must use the fully-qualified domain name for the Web server.

6 Select **User Principal Name (UPN)** as the identity claim. Click **Next**.

7 Verify that **Enable this application** is selected, then click **Next**.

8 Click **Finish**.

Repeat for each sample application.

9 For adfs-ordering.war only: To test authentication and authorization using Centrify and AD FS for the adfs-ordering application sample application you need to create and enable the following organization claims:

Claim Type	Resource partner organization claims
Group organization claims	Administrator Purchaser Silver, Gold or Platinum
Custom organization claims	Displayname Title

Note For your existing applications with claims: After the application has been added, be sure to enable any claims that have previously been created for the application.

Enable UPN on the account server

You need to make sure the User Principal Name is enabled for the resource server on the account server.

1 Click **Start > Administrative Tools > Active Directory Federation Services**.

2 Select **Federation Services > Partner Organizations > Resource Partners** and click on the resource server.

3 If the **User Principal Name** is undefined (that is, it does NOT appear in the Organization Claim column), right click on it, select **Properties**, check the **Enabled** box and click OK.

On the other hand, if the **User Principal Name** already appears in the **Organization Claim** column, you are already set.

Creating group organization claims on the account server

Note You do NOT need to create a group claim to run the sample applications. This step is for illustration purposes only.

Organization claims are defined in Active Directory Federation Services on the account federation server and populated with information from Active Directory. The organization claims are mapped to outgoing claims that are sent to the resource federation server. The resource federation server receives these as incoming claims and maps them into its own organization claims. Each of the claims required by an application is then enabled so that the application may receive the claims.

Note Claim names are case-sensitive.

1 Log on to account federation server using the Administrator account and password.

2 Click **Start > Administrative Tools > Active Directory Federation Services**.

3 Expand the **Federation Service > Trust Policy > My Organization**.

4 Select **Organization Claims**, right-click, then select **New > Organization Claim**.

5 Enter a group name, for example, `Purchasing`, select **Group**, then click **OK**.

Populating the group claims on the account server

Now that the group claim is created you need to say which Active Directory groups are associated with it.

Note This step assumes that the account federation server contains the Active Directory domain controller.

You populate the Arcade, Purchasing Agent and Purchasing Administrator claims from Active Directory Users and Computers on the account federation server.

1 Log on to account federation server using the Administrator account and password.

2 Click **Start > Administrative Tools > Active Directory Federation Services**.

3 Expand the **Federation Service > Trust Policy > My Organization > Account Stores**.

4 Right click **Active Directory** then select **New > Group Claim Extraction**.

5 Click **Add** and type characters for the search and the **Check Names**. For example, enter “au” to find Authenticated Users or “do” for Domain users.Select from the list then click **OK**.

6 Select the Organization Claim you just created (for example, `Purchasing`) from the drop-down menu and click **OK**.

Mapping outgoing group claims on the account server

Finally the organization claims on the account server need to be converted to match organization claims the resource partner is configured to accept as incoming claims.

To map group claims to outgoing group claims:

1. Log on to account server using the Administrator account and password.
2. Click **Start > Administrative Tools > Active Directory Federation Services**.
3. Expand the **Federation Service > Trust Policy > Partner Organizations > Resource Partners**.
4. Right click resource server then select **New > Outgoing Group Claim Mapping**.
5. Select the **Organization group claim** you just created from the drop down menu and type the name of the corresponding Outgoing group claim, then click **OK**.

Add sample applications to AD FS 2.0

Use the following instructions to prepare the AD FS 2.0 environment to authenticate users for the sample applications. The instructions assume you have already set up the AD FS 2.0 environment as follows:

- The account and resource servers have the other's certificate in the Trust Root Certification Authority.
- On the account server you have the resource server as a Relying Party Trust
- On the resource server you have the accounts server as a Claims Provider Trust

The following instructions add some claim rules for the resource server's relying party trust on the account server and the account server's claim provider trust on the resource server. You may already have these in place.

This section is broken down in two parts:

- Part 1: On the account server: Describes how to set up the claim rules for the resource server relying party trust. In addition, instructions are provided to set up group claims; this is for demonstration purposes only, you do not need to set up group claims to run the sample applications.
- "Part 2: On the resource server" on page 101: This describes how to add claim rules for your account server's claim provider trust for testing purposes only, add the demo applications as relying party trusts and create the claim rules for the applications.

Part 1: On the account server

Log on to the account server as the administrator.

Edit Claim Rules for the AD FS resource

From the AD FS 2.0 Management tool window, right click on the **Relying Party Trust** for your resource server and select **Edit Claim Rules**.

1 Select the claim attributes from an LDAP attribute store (in our case, Active Directory):

   a Click **Add Rule**

   b Select **Send LDAP Attributes as claims** and click **Next**.

   c Fill in the fields as follows

Claim rule name	LDAP UPN to AD FS 1.0 UPN
Attribute store	Active Directory
LDAP Attribute	User-Principal-Name
Outgoing claim type	AD FS 1.x UPN

   Note In this example, the outgoing type `AD FS 1.x UPN` was selected. If you select `Name ID` instead, you can skip the next step.

2 Transform an incoming `AD FS 1.0 UPN` claim type to an outgoing `Name ID` claim type.

   a Click **Add Rule**

b Select **Transform an Incoming Claim** and click **Next**

c Fill in the fields as follows

Claim rule name	AD FS 1.0 UPN to Name ID
Incoming claim type	AD FS 1.0 UPN
Outgoing claim type	Name ID
Outgoing name ID format	UPN

d Select **Pass through all claim values** (may be selected by default), and click **Finish.**

Add group claims

Note You do NOT need to create a group claim to run the sample applications. This step is for illustration purposes only.

From **Edit Claim Rules**

- Click **Add Rule**
- In **Claim rule template** select **Send Group Membership as Claim** and click **Next.**
- Enter the following fields in the window:

Claim rule name	Make up a name for this rule
User's group	the Active Directory group you want mapped to this claim rule name
Outgoing claim type	Group
Outgoing claim value	Enter the name of the group you will use at the relying party trust for this group (can be the same as the claim rule name)

OPTIONAL: Here's how you would add a rule to pass through all group claims

a Click **Add Rule**

b Select **Pass Through or Filter an Incoming Claim**

c Fill in the fields as follows

Claim rule name	Pass through all group claims
Incoming claim type	Group

d Select **Pass through all claim values**, and click **Finish**

Part 2: On the resource server

Log on to the resource server as the administrator.

Edit claim rules for the AD FS 2.0 account server

1 Open **Administrative Tools > AD FS 2.0 Management**

2 Expand **AD FS 2.0 > Trust Relationships**

3 Click on **Claims Provider Trusts** and right click the Claims Provider Trust for your account server.

4 Select **Edit Claim Rules** and for the sake of testing, add a claim rule that lets all Name ID UPN claims pass through.

a Click **Add Rule**

b Click on **Pass through or filter an incoming claim** and click **Next**

c Fill in the fields as follows

Claim rule name	Pass through all Name ID UPN claims
Incoming claim type	Name ID
Outgoing claim format	UPN

d Select **Pass through all claim values**, and click **Finish**

OPTIONAL: Here's how you would add a rule to pass through all group claims

a Click **Add Rule**

b Select **Pass Through or Filter an Incoming Claim**

c Fill in the fields as follows

Claim rule name	Pass through all group claims
Incoming claim type	Group

d Select **Pass through all claim values**, and click **Finish**

Add the sample applications as a Relying Party Trust

You have to make both sample applications a Relying Party Trust on the resource server and add claim rules for each. Perform the following steps to configure the AD FS traditional application and then repeat them to configure the AD FS claims-aware application.

Log on to the resource server and open the **Administrative Tools > AD FS 2.0 Management**.

Start with adding each sample application as a Relying Party Trust:

1 Expand **AD FS 2.0 > Trust Relationships.**

2 Right click on **Relying Party Trusts** and select **Add Relying Party Trust.**

3 Click **Start.**

4 Select **Enter data about the relying party manually** because you are adding an application rather than a resource server. Click **Next.**

5 Enter the description for the **Display name** of the traditional application, for example, AD FS traditional app, click **Next.**

6 Select **AD FS 2.0** profile, click **Next.**

7 Click **Next** again (you do not need at optional token encryption certificate).

8 In the next window

a Check the **Enable support for the WS-Federation Passive protocol** box.

b Under **Relying party WS-Federation Passive protocol URL**, enter the URL for the entry page of the traditional application. For

example, if the you are configuring the `adfs-traditional` application on a Tomcat, JBoss, WebLogic or WebSphere server the URL is in the form:

```
https://[myhostname.domain.com]:port/centrifydc-samples/adfs-traditional/entry.jsp
```

where *myhostname.domain.com* is the fully qualified name and *port* is one of the following

Server	Default port
Tomcat and JBoss	8443
Weblogic	7002
WebSphere	9443

c If you are using an Apache server, the URL is slightly different

```
https://[myhostname.domain.com]/samples/adfs-traditional/entry.cgi
```

d Click **Next**.

9 Confirm that the application URL you just entered s under Relying party trust identifiers in the second box, then click **Next.**

10 Leave **Permit all users to access this relying party** selected, click **Next.**

11 Click on **Identifiers** and **Endpoints** tabs and confirm that the urls are correct as you entered, click **Next.**

12 Uncheck the **Open the Edit Claims Rules dialog for this** ... , click **Close**.

To finish, edit the application's relying party trust claim rules to let all Name ID UPN claims pass through.

13 Right click on the application Relying Party Trust you just created and select **Edit Claim Rules** to pass through all Name ID type claims.

a Click **Add Rule.**

b Click on **Pass through or filter an incoming claim** and click **Next.**

c Fill in the fields as follows:

Claim rule name	Pass through all Name ID UPN claims
Incoming claim type	Name ID
Outgoing claim format	UPN

d Select **Pass through all claim values**, and click **Finish**

14 OPTIONAL: Here's how you would add a rule to pass through all group claims

a Click **Add Rule**

b Select **Pass Through or Filter an Incoming Claim**

c Fill in the fields as follows

Claim rule name	Pass through all group claims
Incoming claim type	Group

d Select **Pass through all claim values**, and click **Finish**

Repeat all of these steps to make the ADFS-claims-aware application a relying party trust and add the pass through claims rule. Here's the URL format

Apache: `https://[myhostname.domain.com]/samples/ADFS-claims-aware/entry.cgi`

Others: `https://[myhostname.domain.com]:[port]/centrifydc-samples/ADFS-traditional/entry.jsp`

15 Select hashing algorithm: You must use the SHA-1 hashing algorithm.

a Right click on the relying party trust you just created and click on **Properties.**

b Click the **Advanced** tab

c Select **SHA-1** from the drop down menu

d Click **OK.**

After you have completed the claims rules for the claims-aware sample application you are done with the configuration and ready to proceed with running the applications.

Verifying authentication using the sample applications

Note The sample traditional application is preconfigured to map all authenticated Active Directory groups and users to the web server's `user` role. The sample claims-aware application authenticates using the credentials you logged on with.

To verify authentication through AD FS for the DirectControl sample applications:

1. Start the Tomcat, JBoss, WebLogic, WebSphere or Apache server.
2. Open a Web browser and go to the DirectControl sample applications main page.

   For a Tomcat, JBoss, WebLogic or WebSphere server use the following command format:

   `https://servername:port/centrifydc-samples`

   For example, if the fully-qualified host name for the server is `test.env.org` you would enter the following

   `https://test.env.org:8443/centrifydc-samples`

   For an Apache server, use the following command format:

   `https://servername/sample`

   instead using the fully-qualified host name for the server. (Apache uses the standard SSL port.)
3. Click each of the Active Directory Federation Services (AD FS) authentication options to test the behavior and verify that your test user is authenticated properly.

   The specific behavior for each of the DirectControl sample applications is different.

   - For the AD FS traditional application, you are prompted to select your home realm and provide your user name and password.

- For the AD FS claims-aware application, click **Authenticate** to authenticate the current user in the account domain and view the user's claims.
- For the AD FS ordering application, click **Place Order** to see about the current user account and its claims. If the account you used to log in is authenticated successfully, the sample application displays detailed information about the account and the authentication provided.

Note If you are not logged on as a valid Active Directory user, you may be denied access to a sample application in some Web browsers. If you see an error message that indicates you are not authorized to view a page, try logging on with a different user account or using a different Web browser.

If authentication is successful, the web page displayed indicates the authenticated user's identity and other details about the user and web environment.

Developing claims-aware Java EE applications for Centrify

This chapter provides reference information and examples to help you modify your existing Java EE applications to work to use the DirectControl for Web Applicationslibraries on your Tomcat, JBoss, WebLogic or WebSphere server.

Claims-aware applications are applications that comply with Security Assertion Markup Language (SAML) and WS-Federation standards for authorization messages. Because these applications are specifically written or modified to recognize the content and format of Active Directory Federation Service claims, DirectControl validates and passes along any verified claims from the client to the application. The application then decides the level of service to provide the client based directly on those claims. If the application needs claims and none are present, it redirects to the Resource Federation Server to get claims.

To handle claims and support Active Directory Federation Services, DirectControl includes APIs that enable an application to query for claim information, query SAML information, obtain raw SAML tokens, and control log-on and log-off operations.

To make an application claims-aware:

1. Copy the DirectControl SAML JSP Tag library, `centrifydc_fs_taglib.jar` file to the application's `WEB-INF/lib` directory.
2. Use the tags and attributes defined in the SAML JSP Tag library to make your application understand and respond to SAML-based claims.

The `aware.jsp` file in the `adfs-claims-aware` and `adfs-ordering` sample applications illustrate how to use the tags to configure claims-aware applications. For reference information about the DirectControl SAML JSP tags and attributes, see

The following topics are covered:

- Understanding claims-aware applications

Understanding claims-aware applications

Claims-aware applications are applications that are configured to understand and use the SAML-based security mechanisms and interfaces and make authorization decisions based on the claims returned in SAML assertions.

To create a claims-aware application, the application needs to include code that generates properly formatted SAML messages that adhere to the requirements described in the WS-Federation standards for authorization messages. To accomplish this, DirectControl provides a library of SAML tags, `centrifydc_fs_taglib.jar`, that can be incorporated into a JSP file, enabling an application to request, receive, and interpret group and custom claim information in interactions with the federation server.

Understanding the SAML tags and attributes

This section describes the JSP tags that are defined in the SAML tag library.

loginURL

Generates a URL that users can use to log on.

Parameters

The `loginURL` tag takes the following parameters:

Use this parameter	To specify
realm	The Active Directory Federation Services URI account to use for logging on. For example: `urn:federation:acme` • If the `realm` is not specified, the `logonRealm` configured in the `centrifydc_fs.xml` is used. • If the `logonRealm` is not specified in the `centrifydc_fs.xml` or in this parameter tag, the login URL generated by the `loginURL` tag will redirect users to the default AD FS account configured on the AD FS server for login. If the `realm` parameter is specified as an empty string, the login URL generated will give users a list of AD FS account partners to select from.
`returnURL`	The URL to redirect user to after a successful login. If not specified, user is returned to the application's entry URL.

Return value

String

Example

```
Please login now: <a href=<saml:loginUrl
realm="urn:federation:acme"/> >login</a>
```

login

Forces a login. This tag calls the `loginURL` tag to generate a login URL and redirects the user to it.

Parameters

The `login` tag takes the following parameters:

Use this parameter	To specify
realm	The Active Directory Federation Services URI account to use for logging on. For example: `urn:federation:acme` • If the `realm` is not specified, the `logonRealm` configured in the `centrifydc_fs.xml` is used. • If the `logonRealm` is not specified in the `centrifydc_fs.xml` or in this parameter tag, the login URL generated by the `loginURL` tag will redirect users to the default AD FS account configured on the AD FS server for login. If the `realm` parameter is specified as an empty string, the login URL generated will give users a list of AD FS account partners to select from.
`returnURL`	The URL to redirect user to after a successful login. If not specified, user is returned to the application's entry URL.

Return value

None

Example

```
<saml:login realm="urn:federation:acme" returnURL="https://
acme.com/app/index.jsp" />
```

logoutURL

Generates a URL that user can use to logout.

Return value

String

Example

```
Logout: <a href=<saml:logoutURL> >logout</a>
```

isAuthenticated

A conditional tag that returns true if the user is authenticated.

Return value

true or false

Example

```
<saml:isAuthenticated var="loggedOn" />
```

ifUserInRole

A conditional tag that returns true if the user is in the given role. If the user has a group claim with the given role name, this tag returns the value true. Note that this is not related to and does not call the standard Java EE `HttpServletRequest` function `isUserInRole()`.

Parameters

The `ifUserInRole` tag takes the following parameters:

Use this parameter	To specify
role	The name of the role to be tested. This parameter is required.
`var`	The name of a variable to set with the result of the test. This parameter is optional.

Return value

true or false

Example

```
<saml:isUserInRole role="Administrator"
var="isUserAdministrator" />
```

getUser

Returns the login user's information as a `SamlPrincipal`, or null if user is not logged in. See the next section for a description of `SamlPrincipal` attributes.

Parameters

The `getUser` tag takes the following parameters:

Use this parameter	To specify
`var`	The name of a variable to set with the resulting user This parameter is optional.

Return value

SamlPrincipal

For information about the `SamlPrincipal` attributes.

Example

```
<saml:getUser var="samlUser" />
```

SamlPrincipal

The following table describes the `SamlPrincipal` attributes.

Attribute	Description	Value type
assertion	The SAML assertion for this user.	SamlAssertion
name	The authenticated user name. For example: `eric@acme.com`	String
samlClaims	List of claims in the SAML assertion.	Array list of Claims

For information about the `SamlAssertion` attributes, see `SamlAssertion`. For information about the `Claim` attributes, see Claim.

Claim

The following table describes the `Claim` attributes.

Attribute	Description	Value Type
`name`	The name for this claim. For example: `group`	String
`namespace`	The `AttributeNameSpace` attribute value of the claim. For example: `http://acme.com/federation/group`	String
value	Value of the claim. For example: `Administrator`	String

SamlAssertion

The following table describes the `SamlAssertion` attributes.

Attribute	Description	Value Type
`audience`	Audience for this assertion, the entry URL of the application. For example: `http://acme.com/app/index.jsp`	String
`authenticationStatement`	The AuthenticationStatement in the assertion.	String
`claims`	List of claims in this assertion.	Array list of claims
`claimSource`	Source of claims in this assertion, i.e. the AD FS account where the user was authenticated. For example: `urn:federation:acme.`	String
`issueInstant`	Time of assertion issuance. For example: `2005-08-15T23:16:58Z`	String
issuer	Issuer of assertion, the AD FS resource. For example: `urn:federation:treyresearch`	String

Attribute	Description	Value Type
majorVersion	Major version number of this assertion, for example, 1.	String
minorVersion	Minor version number of this assertion, for example, 1.	String
notBefore	Value of the `notBefore` attribute in the SAML assertion. For example: `2005-08-15T23:16:58Z`	String
notAfter	Value of the `notOnOrAfter` attribute in the assertion. For example: `2005-08-16T00:16:58Z`	String

For information about the `AuthenticationStatement` attributes, see `SamlAssertion`. For information about the `Claim` attributes, see Claim.

AuthenticationStatement

The following table describes the `AuthenticationStatement` attributes.

Attribute	Description	Value Type
`authenticationInstant`	Time of authentication. For example: `2005-08-15T23:16:57Z`	String
`authenticationMethod`	The authentication method. For example: urn:oasis:names:tc:SAML:1.0:am:password	String
`nameIdentifier`	The authenticated subject name. For example: `eric@acme.com`	String

Using the SAML tag library in a JSP file

The following example shows a sample JSP file that uses the SAML tag library to create a claims-aware application. This JSP file is similar to the `aware.jsp` in the Tomcat `adfs-claims-aware` sample application.

```
<%@ page language="java" contentType="text/html" %>
<%@ page import="com.centrify.fs.*" %>
<%@ page import="com.centrify.fs.saml.*" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/
core" %>
<%@ taglib prefix="saml" uri="http://www.centrify.com/
ADFS/samltaglib" %>
<%@ include file="nocache.jspf" %>

<saml:isAuthenticated var="loggedOn">
<saml:getUser var="samlUser" />
</saml:isAuthenticated>
<html>
  <head>
    <title>welcome <c:out
value="${loggedOn?samlUser.name:''}" /></title>
  </head>
  <body bgcolor="white">
  <img src="centrify_horiz_color_pos.jpg"/>
  <c:choose>
      <c:when test="${!loggedOn}">
      <p>Greetings! Welcome to Centrify sample claims-
aware application.
     <p>You are not logged in. Click here to login now:
  <a href=<saml:loginUrl returnUrl="aware.jsp"/>>login</
a></p>
   </c:when>
    <c:otherwise>
      <p>Congratulations! You are logged in as
${samlUser.name}.</p>
         You were authenticated by
${samUser.assertion.claimSource}
         ${samlUser.assertion.Issuer}.<br>
         Your SAML token was valid from
${samlUser.assertion.NotBefore}
```

```
            until ${samlUser.assertion.NotOnOrAfter}.<br>
            Your authentication method was

${samlUser.assertion.authenticationStatement.authenticat
ionMethod}.<br>
      <p>All claims in your SAML assertion:</p>
      <table border="1">
      <th>Name</th><th>Value</th><th>NameSpace</th>
      <c:forEach var="claim"
items="${samlUser.samlClaims}">
            <tr><td>${claim.name}</td>
                <td>${claim.value}</td>
                <td>${claim.nameSpace}</td></tr>
      </c:forEach>
      </table>
      <p>Click here to logout: <a href=<saml:logoutUrl /
>>logout</a></p>
    </c:otherwise>
  </c:choose>
  </body>
</html>
```

Understanding the sample application layout

The `adfs-claims-aware` application is a simple claims-aware application that demonstrates using the SAML JSP tag library to control access to the application. With it, users log on and if authenticated properly can see information about their claims and configuration details about their current environment.

The `adfs-claims-aware.war` for this application consists of the following:

```
META-INF/MANIFEST.MF
WEB-INF/centrifydc_fs.xml
WEB-INF/lib/Centrifydc_fs_taglib.jar
WEB-INF/web.xml
bye.gif
centrify_horiz_color_pos.jpg
aware.jsp
welcome.jsp
```

```
entry.jsp
nocache.jsp
```

The main JSP file for this application is the `aware.jsp` file. It uses various JSP tags from the DirectControl SAML tag library to control access to the application and to display information from SAML tokens.

Understanding the centrifydc_fs.xml file

To enable authentication for any supported, Java-based Web application, DirectControl provides two **deployment descriptor** files that enable you to define important characteristics of your environment and how individual applications implement authentication and authorization: through Active Directory or Active Directory Federation Services.

- The `centrifydc.xml` file in this directory is used to configure options for applications in an Active Directory environment.
- The `centrifydc_fs.xml` file in this directory is used to configure options for applications when you are using Active Directory Federation Services.

Default versions of these deployment descriptor files are installed when you install DirectControl for Web Applications in the `/usr/share/centrifydc/java/templates` directory.

Centrifydc_fs.xml layout

The default `centrifydc_fs.xml` file is an XML file you can use as a template for configuring individual Web applications to use Active Directory Federation Services for authentication and authorization. Once you copy this sample file to an application's `WEB-INF` directory, you can customize the content of the file to suit that specific application.

The primary purpose of the `centrifydc_fs.xml` file is to configure an application to work with Active Directory Federation Services. Within this file, you can configure many aspects of the authentication and authorization process by adding or modifying the elements defined in the file.

Extensible Markup Language (XML) files, like the `centrifydc_fs.xml` file, are structured documents that contain a set of supported elements enclosed in opening and closing angle (< >) brackets. The elements can be required or optional depending on the requirements

of the application. The following example illustrates how elements are defined in the `centrifydc_fs.xml` file. In this example, the AD FS resource server is `fire.arcade.com` and the Web server is `zen.arcade.com`:

```
<!--
  Set attribute forceAuth="true" to automatically redirect
   unauthenticated users to the federation service logon
page.
   Defaults to false if not set.
-->
<CentrifydcFS forceAuth="false">
<!--
   Set federationServiceUrl to the Federation Service url.
-->
  <federationServerUrl>https://fire.arcade.com/ADFS/fs/
   federationserverservice.asmx</federationServerUrl>

<!--
  Set entryUrl to your application's entry url.
-->
  <entryUrl>https://zen.arcade.com:8080/startpage.html</
entryUrl>

<!--
   Set maxClockSkew to the maximum allowed clock skew, in
seconds,
   between the server and federation server in validating
SAML
   assertions. Defaults to 5 minutes if not set.
-->
   <maxClockSkew>300</maxClockSkew>

<!--
   Set useCookies to false if you do not want cookies to be
   used. If false, user will be authenticated at the
beginning of
   each session. The default is true.
-->
   <useCookies>true</useCookies>
...
```

The template `centrifydc_fs.xml` file contains some default settings you must modify in the copy you make for your application. Make the changes before you place the file in the application's `WEB-INF` directory.

Centrify_fs.ml elements

The following table describes the elements you set in the `centrifydc_fs.xml` file:

Use this element	To do this
`CentrifydcFS`	Identify the contents of the file and set the `forceAuth` or `Realm` attributes. As the top-level element, the `CentrifydcFS` element is required. If the `forceAuth` attribute is set to `true`, the application automatically redirects unauthenticated users to the federation service logon page. Setting this attribute to `true` is useful if you are configuring traditional applications with no predefined roles declared in the `web.xml` `auth-constraints` but still want to force users to login when they access the application. The default setting for this attribute is `false`. The `realm` attribute allows to specify a realm name to automatically authenticate to for an application. If this attribute is not specified, a list of realms for user to choose from is displayed by default. If this attribute is set to `default`, the default realm on the AD FS resource server is used.
`federationServerUrl`	Set the URL of the Active Directory Federation Services resource federation server. Replace the `ADFS_SERVER_HOST` and `ADFS_SERVER_PORT` placeholders in this element with the fully qualified name of the resource federation server and, if appropriate, the port number it uses for communication with the web server. This element is required.

Use this element	To do this
`entryUrl`	Set the URL for accessing an application. Replace the APP_SERVER_HOST, APP_SERVER_PORT, and APP_PATH placeholders in this element with the fully qualified name of the web server, the port number the web server uses, if appropriate, and the path to use to access the application. This element is required.
`maxClockSkew`	Set the maximum number of seconds apart the system clock on the web server and the system clock on the federation server can be when validating SAML assertions. The default clock difference is 5 minutes. This element is optional.
`useCookies`	Specify whether you want to allow cookies to be stored for authenticated users. If set to `true`, once authenticated, users can start new application sessions without being prompted to re-enter their credentials. If set to `false`, users must be authenticated at the beginning of each session. The default is `true`. This element is optional.
`cookiePath`	Set to the path to where cookies are stored. If you do not specify a path and set `useCookies` to `true`, the default path used is the path to the application URL. If you use the application URL, the cookie can only be used for that application. To share the cookie across multiple applications on the same server, set the `cookiePath` element to the root (`/`) path or a single common location. This element is optional.
`cookieDomain`	Set the domain name for the cookie placed in the browser after a successful authentication. If you do not specify a domain, the default is the server host name. This element is optional.

Use this element	To do this
`logonRealm`	Set the realm name for the account federation service to which users log on. For example: `urn:federation:acme` If you do not specify a logon realm, users are redirected to the default realm on the account federation service server. If this element is set to an empty string, users can select a logon realm from a list of realms available. This element is optional.
`queryInterval`	Set the interval in seconds for retrieving the logon URL and trust information from the federation service. If you set the `queryInterval` to 0, the information is obtained only once when starting the application and not again. The default interval is every 30 seconds. This element is optional.
`signOffImage`	Specify the image you want displayed after a user signs off and ends a session. This element is required.
`SetHeaders`	Set the authenticated user's attributes in HTTP headers. In most cases, there's no need to change the default values for user attributes. These elements are optional.

Use this element	To do this
`SetRequestAttrs`	Set the authenticated user's attributes in the http servlet request attributes. In most cases, there's no need to change the default values for user attributes. These elements are optional.

Use this element	To do this
`RoleMapping`	Define how security constraint roles in Tomcat and JBoss applications map to Active Directory group names. The `separator` attribute defines the character to use as a separator between multiple entries. For example, if you want to specify multiple Active Directory group or user names, you can use this character to separate entries: `group="z1.com/HR;z1.com/AP"` The `<RoleMapping>` element contains the sub-element `<Role>` that uses following attributes: • `name` specifies the name of a application role. This attribute is required. • `group` specifies one or more Active Directory groups that the specified role `name` maps to. You must use the full canonical group name. Use the character defined in the `separator` attribute if you are specifying multiple Active Directory groups. To allow all groups in a domain or organizational unit, you can use an asterisk (*) in place of the group name. • `user` specifies one or more Active Directory users that the application role maps to. Use the character defined in the `separator` attribute if you are specifying multiple Active Directory users. To allow all users in a group, you can set this attribute to `user="*"` or not specify this attribute. You must specify define at least one `group` or `user` attribute for a role `name`. You can specify any number of `<Role>` elements in the `<RoleMapping>` section. These settings do not apply for applications in the WebSphere Application Server or for WebLogic applications. Those application servers have their own role mapping mechanisms. For information about role mapping, see the appropriate chapter for your Web application environment.

Index

A

C

D

E

F

G

T

U

V

W