RSA® Access Manager Agent 5.0 SP1 for RedHat JBoss Application Server Installation and Configuration Guide

Contact Information

Go to the RSA corporate web site for regional Customer Support telephone and fax numbers**:** **www.rsa.com**

Trademarks

RSA, the RSA Logo and EMC are either registered trademarks or trademarks of EMC Corporation in the United States and/or other countries. All other trademarks used herein are the property of their respective owners. For a list of EMC trademarks, go to **www.rsa.com/legal/trademarks_list.pdf**.

License agreement

This software and the associated documentation are proprietary and confidential to EMC, are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice below. This software and the documentation, and any copies thereof, may not be provided or otherwise made available to any other person.

No title to or ownership of the software or documentation or any intellectual property rights thereto is hereby transferred. Any unauthorized use or reproduction of this software and the documentation may be subject to civil and/or criminal liability. This software is subject to change without notice and should not be construed as a commitment by EMC.

Third-party licenses

This product may include software developed by parties other than RSA. The text of the license agreements applicable to third-party software in this product may be viewed in the **thirdpartylicenses.pdf** file.

Note on encryption technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

Distribution

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

December 2013

Contents

Preface

About This Guide

This guide describes how to install and configure the RSA® Access Manager Agent 5.0 SP1 for RedHat JBoss Application Server. Do not make this guide available to the general user population.

This guide is intended for network, web site, security administrators, and other trusted personnel who are responsible for installing and managing the Agent.

This guide assumes that the personnel possess a basic understanding of web technologies, UNIX or Windows operating systems, and the JBoss Application Server environment. For start-to-finish security administration with this Agent, expertise in JBoss Application Server and the RSA Access Manager administrative tools is essential. The knowledge of JBoss administrative console, and the RSA Access Manager Administrative Console is required.

RSA Access Manager Agent 5.0 SP1 for RedHat JBoss Application Server Documentation

For more information about RSA® Access Manager Agent 5.0 SP1, see the following documentation:

***Readme* or *Release Notes*.** Provides information about what is new and changed in this release, as well as workaround for known issues. The latest version of the *Readme/Release Notes* is available on RSA SecurCare Online at **https://knowledge.rsasecurity.com**.

***Installation and Configuration Guide*.** Describes detailed procedures on how to install and configure RSA® Access Manager Agent 5.0 SP1.

Related Documentation

For more information about products related to RSA® Access Manager Agent 5.0 SP1, see the **RSA Access Manager documentation set.** The documentation related to RSA Access Manager is available from RSA SecurCare Online at **https://knowledge.rsasecurity.com.**

To complete the full deployment of the integrated solution for JBoss, refer the JBoss documentation available at **http://www.jboss.org/jbossas/docs**.

Support and Service

RSA SecurCare Online	https://knowledge.rsasecurity.com
Customer Support Information	www.rsa.com/support
RSA Secured Partner Solutions Directory	www.rsasecured.com

RSA SecurCare Online offers a knowledge base that contains answers to common questions and solutions to known problems. It also offers information on new releases, important technical news, and software downloads.

The RSA Secured Partner Solutions Directory provides information about third-party hardware and software products that are certified to work with RSA products. The directory includes Implementation Guides with step-by-step instructions and other information about interoperation of RSA products with these third-party products.

Before You Call Customer Support

Make sure that you have direct access to the computer running the RSA Access Manager Agent software.

Please have the following information available when you call:

- ❑ Your RSA Customer/License ID.

This is a paper license. You can find this number only on the license distribution medium. If you do not have access to the paper-based RSA Customer/License ID, contact RSA Customer Support.

- ❑ RSA® Access Manager Agent 5.0 SP1 software version number.
- ❑ The make and model of the machine on which the problem occurs.
- ❑ The name and version of the operating system under which the problem occurs.

1 Overview of RSA Access Manager Agent 5.0 SP1

RSA Access Manager Agents Overview

The RSA Agent allows you to protect web resources and Java 2 Enterprise Edition (J2EE) resources on application servers, web applications, JNDIs, EIS, JMS, and Java Database Connectivity (JDBC).

By adding the RSA Agent to JBoss Application Server, you can protect the web applications and implement single sign-on to the web application environment.

The Access Manager Agent performs the following tasks:

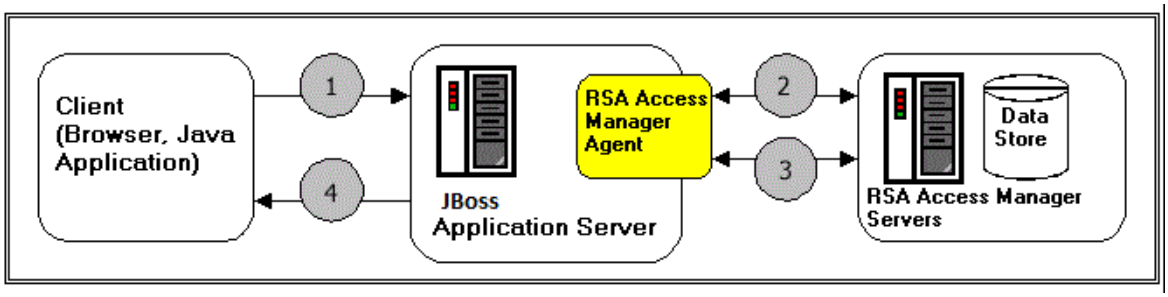
- Protects J2EE resources such as Enterprise Java Beans (EJBs), Java Server Pages (JSPs), Java Servlets and so on.
- Provides Single Sign-On
- Protects web services and applications

After you install the RSA Agent on JBoss Application server, you can use multiple components of Access Manager Agent for providing a standard set of authentication and authorization features for web resources.

The RSA Agent provides integrated administration of users, groups, role-to-user or role-to-group mapping, and provides runtime support for JBoss authorization through J2EE resource access checking.

At runtime, the RSA Agent is invoked to authenticate a user when the user tries to access a protected resource. The Agent authenticates the user by verifying the user's credentials with the RSA Access Manager Servers. Once the authentication is successful, the RSA Agent is invoked again to check the user's authorization. The RSA Agent calls the RSA Access Manager Servers to check whether the authenticated user or the user's group has the required role or roles. Access to the J2EE resource is granted based on whether the user or the user's group has the required role or roles.

The following graphic illustrates this process at a high level.



1. A client, such as a Java application over Remote Method Invocation (RMI) or an end user's browser over HTTP, makes a request for a protected Servlet, EJB, or method.
2. JBoss invokes the Agent, which performs an authentication check and calls the RSA Access Manager Servers and data store for the user and the group data.
3. After successful authentication, JBoss invokes the Agent to perform a role-based authorization check. The Agent calls the RSA Access Manager Servers for role-to-user or role-to-group mapping and performs the role-based authorization.
4. After authentication and role-based authorization checking, JBoss returns either the requested resource or an access denial to the client.

RSA Access Manager Agent Components

The RSA Access Manager Agent for application server is a combination of multiple components.

Web Filter

Web filters are URL filters that are implemented as Servlet Filters. They provide a standard set of authentication and authorization features for web resources like any other RSA Access Manager Agent for web servers. A web filter must be applied to each web application that needs to be protected by RSA Access Manager.

Web filters support different types of authentication. For more information, see Protect Web Applications on page 40.

Java Authorization Contract for Containers (JACC) Provider

The RSA Access Manager Java Authorization Contract for Containers (JACC) provider is a new feature in JBoss Application Server. RSA Access Manager Agent 5.0 SP1 supports this feature. With JACC, authorization is completely delegated to the Agent. This also allows dynamic creation and deletion of resources, groups, and mapping of users and groups to the roles in RSA Access Manager.

The Agent installer automatically configures RSA Access Manager JACC Provider for JBoss Application Server during the installation. If RSA Access Manager JACC Provider is configured, when you deploy an application it will create corresponding RSA Access Manager groups, applications, and resources for authorization based on RSA Access Manager. Administrators can change the authorization requirements using the RSA Access Manager Administrative Console.

As an example of how RSA Access Manager JACC Provider creates resources and groups in Access Manager, consider a banking application (named BankingApp) with resources protected by a role called "Teller".

RSA Access Manager JACC Provider creates all the protected resources as an Enterprise Application Server Resources in RSA Access Manager. RSA Access Manager JACC Provider then creates a group called "Teller_BankingApp_servername_CTROLE" and grants access permissions to all the protected resources in the BankingApp. If the BankingApp has any user-to-role mappings or group-to-role mappings, then the corresponding users and groups are made members of the "Teller_BankingApp_servername_CTROLE" so that they can get access permission to the resources.

Users and groups can be granted access to the protected resources by the following methods:

- Before deploying the application, specifying users and groups in the Deployment Descriptor.
- During the deployment of the application in the JBoss Administrative Console, specifying users and groups through the "Security role to user/group mapping" option.
- After deploying the application, through the RSA Access Manager Administrative Console, by adding users and groups to the group that corresponds to the role in the Application.

Security Roles in JBoss Application Development

When building an application, a developer indicates the JBoss resources to be protected, and defines security roles controlling access to them.

For example, a banking application requires separate roles for managers and supervisors, in which managers are authorized to transfer money between accounts while supervisors are only allowed to view the accounts. In this case, the manager role is granted access to the J2EE resources that provide the transfer functionality, but the supervisor role is denied this access.
This assignment of roles is accomplished through the JBoss Application Assembly Tool, or directly through the application's Deployment Descriptor.

UnChecked Resources

When building an application, a developer declares a resource as unchecked in the Deployment Descriptor. When a resource is unchecked, the authentication for that resource does not take place and the resource is granted to all users. The resource is treated as an unprotected resource.

Excluded Resources

When building an application, a developer can declare a resource as excluded in the Deployment Descriptor. When a resource is excluded, the resource is not granted to any user.

SSO Valve

JBoss supports using external web servers as front-ends for serving web resources. Requests for web resources are routed to JBoss through the plug-ins installed in the front-end web servers.

In order for JBoss to trust these external web servers, the Single Sign-On (SSO) Valve for these front-end web servers must be configured in JBoss. The SSO valve intercepts the requests from the front-end web servers and inform JBoss whether the request is authenticated by the front-end server or not. JBoss can delegate authentication to these front-end web servers and provide single sign-on. Once a user has authenticated to the front-end web server, the user does not need to reauthenticate to JBoss.

The RSA Access Manager Agents supports a number of web servers. These Agents are URL filters that provide a standard set of authentication and authorization features. For a list of JBoss-supported front-end web servers, refer to the JBoss Documentation at **http://www.jboss.org/jbossas/docs**.

Single sign-on from any RSA Access Manager Agent for web servers to the Agent for JBoss can be done through the SSO Valve. With the appropriate SSO Valve installed, the end users do not need to reauthenticate to the Agent for JBoss after they have successfully authenticated to an Agent for web servers.

Web Services Filter

Like web applications, web services can also be protected using the web filters provided by RSA Access Manager. Since web services are invoked using the SOAP protocol, not all the features of RSA Access Manager web filter are available. For more information, see Protecting Web Services.

Supported Features

This section provides information on the features offered by the RSA Agent.

Support for Linux

The RSA Agent supports Red Hat Enterprise Linux version 6, 64-bit.

Authentication and authorization of enterprise applications

The Agent supports J2EE role-based authentication and authorization of enterprise applications. Both authentication and authorization of resources are managed through RSA Access Manager.

SSO Valve

JBoss supports authentication using external web servers through Singe Sign-On (SSO) Valve. The Agent provides the RSA Access Manager implementation of SSO Valve, which can be used to trust external web servers that are protected with RSA Access Manager Web Agents.

Java Authorization Contract for Containers (JACC)

The contract enables third-party authorization providers to plug into JBoss, to make the authorization decisions when a J2EE resource is accessed. The Agent enables the RSA Access Manager implementation of JACC, which can be plugged-into JBoss to protect the J2EE Resource. Authentication and authorization of web services

The Agent supports both authentication and authorization of web services. Web services can be protected at the application level.

Support for JBoss Application Server 7.x

The Agent provides SSO and Authentication support for JBoss Application Server 7.x resources.

Multiple authentication types

You can choose the method with which to authenticate users before allowing access to your protected web resources: Basic and Windows NT (user name/password), X.509 Certificate, RSA SecurID (two-factor authentication), and Custom. The Agent supports the use of form-based authentication, which allows you to customize the logon pages displayed to users.

Granular authentication

You can choose whether to base user authentication on more than one authentication type.

SSL

Protects connections between the Agent and RSA Access Manager Servers.

Uniform resource locator (URL) retention

Enables the web server to retain original page request URLs instead of sending users to a default location, such as a home page, after logging on.

Single sign-on (SSO)

After the user has authenticated once, SSO allows the user to access other protected resources without having to reauthenticate each time, provided the user has access privileges to the resource.

Custom user properties

Allows you to publish user information that you have defined as custom user properties in the RSA Access Manager Administrative Console.

On-demand configurable cacheflush

Allows you to flush the Agent cache based on cache type. By passing selected arguments, you can flush combinations of cached protected resources, unprotected resources, user properties, session tokens, "allow" authorizations, and "deny" authorizations.

Improved performance

Authorization Server pool refresh time outs are now configurable. Also, URL exclusion lists and file extension exclusion lists have been fully implemented. These and other performance settings can be configured in the Agent configuration file, cleartrust.properties.

Proxy and firewall environment support

You can define how cookies are handled by the Agent when proxy servers and firewalls are in use.

User Principal Name (UPN) support

UPN is an Internet-style login name for a user based on the Internet standard RFC 822. The UPN is shorter than the distinguished name and easier to remember. By convention, this should map to the user e-mail id. The value set for this attribute is equal to the length of the user's ID and the domain name. For more information about this attribute, see the Naming Properties topic in the Active Directory guide.

Utilities

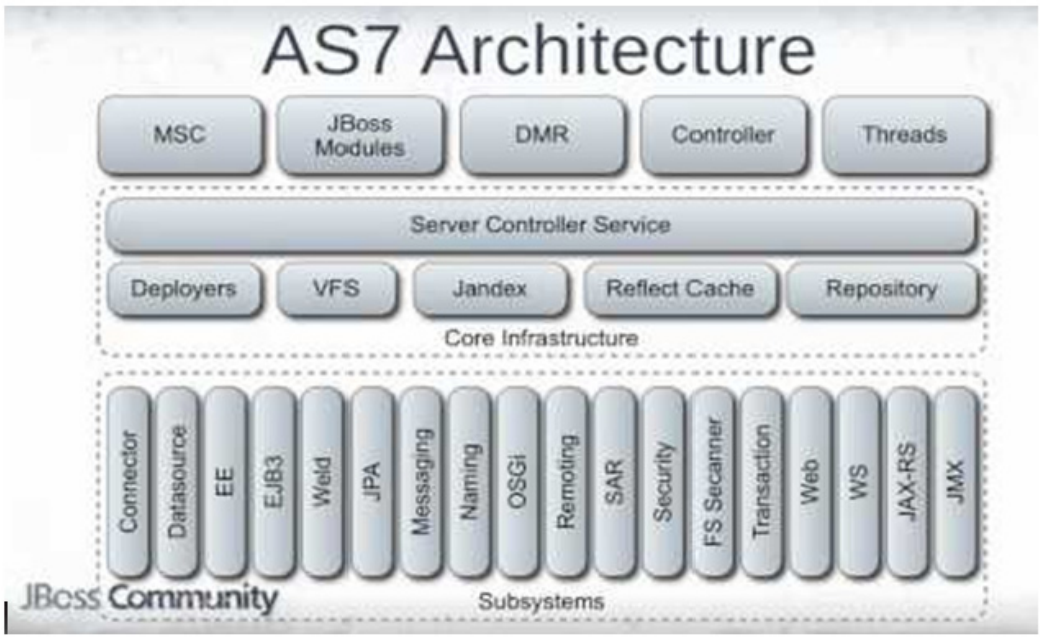
The Agent provides several utilities. Ctencrypt allows you to encrypt sensitive Agent configuration information. Cacheflush allows you to flush the Agent-side cache to affect immediate enforcement of changed RSA Access Manager security policies.

2 JBoss Application Server Overview

JBoss Architecture Overview

JBoss Application Server 7, is a fast, powerful, implementation of the Java Enterprise Edition 6 specification. The state-of-the-art architecture built on the Modular Service Container enables services on-demand when the application requires them. JBoss Application Server 7 is a certified implementation of the Java Enterprise Edition 6 Web Profile specification.

The following figure provides the high level architecture for the JBoss Application Server:



PicketBox or JBoss application Server Security SubSystem Structure

PicketBox is the foundational security framework that provides the authentication, authorization, audit and mapping capabilities to Java applications. The security subsystem operates by using a security context associated with the current request.

The security context enables the relevant container with the following capabilities from the configured security domain.

- Authentication Manager
- Authorization Manager
- Audit Manager
- Mapping Manager

Security on JBoss

The J2EE specifications define a simple role-based security model for EJBs and web components. The JBoss component framework that handles security is the JBossSX extension framework. The JBossSX security extension provides support for both the role-based declarative J2EE security model and integration of custom security via a security proxy layer. The default implementation of the declarative security model is based on Java Authentication and Authorization Service (JAAS) login modules and subjects.

J2EE Declarative Security

The security model advocated by the J2EE specification is a declarative model. In this model you describe the security roles and permissions in a standard XML descriptor rather than embedding security into your business component.

Securing a J2EE application is based on the specification of the application security requirements through the standard J2EE deployment descriptors. You secure access to EJBs and web components in an enterprise application by using the *ejb-jar.xml* and *jboss-web.xml* deployment descriptors.

JAAS Security Manager MBean

The Java Authentication and Authorization Service (JAAS) is a framework for user-level security in Java applications, using pluggable authentication modules (PAM). It is integrated into the Java Runtime Environment (JRE). In JBoss Enterprise Application Platform, the container-side component is *org.jboss.security.plugins.JaasSecurityManager MBean.*

Java 2 Security- Authorization at runtime

The standard Java Security model allows you to grant permissions to code origin or the identity of the code owner. The code origin is based on the classpath location from where a class is loaded; the identity of the code owner is achieved using signed Jar files. JAAS adds to this equation the identity of the user executing the application.

Policy files can be created to grant permissions to applications. The global policy file (java.policy) is loaded from $JAVA_HOME/lib/security. You can also specify the policy file at execution time using the *D-flag: -Djava.security.policy*

Java Authorization Contract for Containers

JACC is the Java Authorization for Container Contracts specification. This allows one to externalize the implementation of the java.security.Policy class that is used to authorize the JACC defined permission.

JACC Permission Model

JACC defines five permission classes that must be used by a JACC-compliant application server to translate the EJB and Web resource declarations in deployment descriptors into equivalent permissions.

- EJBMethodPermission
- EJBRoleRefPermission
- WebResourcePermission
- WebRoleRefPermission
- WebUserDataPermission

The mapping between a J2EE deployment descriptor and the JACC permission model is not one-to-one. A J2EE deployment descriptor defines a sets of roles that have a set of permissions, whereas the JACC model is much simpler and defines a collection of roles that have permissions.

JBoss JACC Integration Service

org.jboss.security.jacc.SecurityService is the service which installs a java.security.Policy implementation that handles the JACC permission checks.

On startup, this service looks to the JACC defined *javax.security.jacc.policy.provider* system property. If this is not specified, it defaults to *org.jboss.security.jacc.DelegatingPolicy* which is the JAAS based implementation

-Djavax.security.jacc.policy.provider=com.rsa.axm.jboss.jacc.RSAJDelegatingPolicy

-Djavax.security.jacc.PolicyConfigurationFactory.provider=
com.rsa.axm.jboss.jacc.RSAJPolicyConfigurationFactory

JBoss Operational Modes

One of the primary features of JBoss application server is the ability to manage multiple JBoss application server instances from a single control point. A collection of such servers are referred to as members of a "domain", with a single Domain Controller process acting as the management control point. Domains can span multiple physical (or virtual) machines, with all instances on a given host under the control of a Host Controller process. The Host Controllers interact with the Domain Controller to control the life cycle of the instances running on the host and to assist the Domain Controller in managing them.

- Standalone Mode
- Domain Mode

When you launch JBoss application server in "domain mode" (using the domain.sh or domain.bat launch scripts) you launch a Domain Controller, a Host Controller, and usually at least one JBoss application server instance.

In many cases, the centralized management capability available through the domain mode is not necessary. You can run in "standalone mode".

Note: RSA Access Manager Agent 5.0 SP1 for JBoss Application Server is officially qualified only on Standalone mode.

3 JBoss Agent Installation

Platform and System Requirements

RSA Access Manager Agent 5.0 SP1 is supported on the following JBoss Application Server versions:

- JBOSS 7.1.2 Enterprise Application Platform (EAP) on RHEL version 6, 64-bit
- JBOSS 7.1.2 Enterprise Application Platform (EAP) on Microsoft Windows Server 2008 R2, 64-bit

RSA Access Manager Agent 5.0 SP1 is supported on the following Access Manager Servers:

- RSA Access Manager Server 6.2 SP1

The Agent must be installed on the same machine as the application server, one Agent for each server instance. A typical installation has the following Agent system requirements:

- 50 MB free disk space, plus another 50 MB free temporary disk space to execute the installer
- 128 MB RAM for the Agent

Installation Overview

The following are the high-level steps for installing this Agent:

1. Set up your JBoss application server as described in the JBoss documentation.
2. Install your RSA Access Manager Servers before you install this Agent. For more information, see the *RSA Access Manager Servers Installation and Configuration Guide.*
3. Perform "Pre-Installation Tasks" and collect the information you need to provide during installation of this Agent.

4. Run the Agent installation program as described in "Install the Agent".
5. Make any other configuration changes in the Agent configuration file as needed. These may include SSL setup tasks or other customized settings listed in Chapter 4, JBoss Configuration and Deployment.
6. Restart JBoss.

Pre-Installation Tasks

This section includes preinstallation tasks, instructions for running the automated installer.

Before you install the Agent, make sure you have all the information described in the following topics:

- Prerequisites on page 20
- Required Information on page 20
- Preinstallation Checklist on page 21

Prerequisites

Make sure you have met these prerequisites:

- JBoss is installed, configured, and running in standalone mode.
- The CLI or Native Interface Port of the JBoss Application Server is open.
- The RSA Access Manager Servers are installed and running.
- The RSA Access Manager Administrative Console is installed.
- The RSA Access Manager data store contains a read-write administrator account for the Agent.

Required Information

During installation, provide the following information:

- JBoss Installation Directory. The path to your JBoss server. If the specified location is not a JBoss installation folder, then the installer throws an error message indicating that the installation path is invalid.
- JBoss Administrator User Name and Password. The installation program sets this user as the JBoss administrator. This user must be present in the RSA Access Manager datastore, but need not be an administrative user in RSA Access Manager.
- RSA Access Manager Entitlements Server Host and Port. Allows the Agent to connect to the Entitlements Server. Be prepared to provide the fully qualified hostname or IP address of your Entitlements Server machine as well as the port number it is listening on.
- RSA Access Manager Entitlements Server Administrative User Name and Password for the Agent. Allows the Agent to connect to the Entitlements Server as an administrative user using the specified administrative user name and password. This user needs only read-write privileges to the RSA Access Manager data store.

- RSA Access Manager Dispatcher Server(s) Host and Port. Allows the Agent to connect to the Dispatcher Server(s), which find operational Authorization Servers. Be prepared to provide the fully qualified hostname or IP address as well as the port number it is listening on for each of your Dispatcher Server machines.
- RSA Access Manager Authorization Server(s) Host and Port. Allows the Agent to connect directly to the Authorization Server. Be prepared to provide the fully qualified hostname or IP address as well as the port number it is listening on for each of your Authorization Server machines.
- RSA Access Manager SSL Mode. The SSL mode for Agent client connections to the Dispatcher or Authorization and Entitlements Servers. The RSA Access Manager Server and Agent SSL mode settings must match. Check the settings in the parameters cleartrust.net.ssl.use in dispatcher.conf and/or aserver.conf and cleartrust.eserver.api_port.use_ssl in eserver.conf. The RSA Access Manager Servers default value is Anon. If your RSA Access Manager Server SSL mode is set to Auth for authenticated SSL connections, you must be prepared to supply the following keystore values to the Agent installer:
  - CA KeyStore Filename
  - CA KeyStore Type
  - CA KeyStore Provider
  - CA KeyStore Passphrase
  - Private KeyStore Filename
  - Private KeyStore Type
  - Private KeyStore Provider
  - Private KeyStore Passphrase
  - Private Key Alias
  - Private Key Passphrase

This information can be found in your RSA Access Manager Servers dispatcher.conf, aserver.conf, and eserver.conf files.

Note: If the RSA Access Manager Server and Agent SSL mode settings do not match, the Agent fails to start.

Preinstallation Checklist

It may be helpful to complete this checklist and keep it available while installing this Agent. Some of this information is very sensitive from a security standpoint, so be careful about circulating or destroying the information after installation.

- ❑ JBoss is installed, configured, and running.
- ❑ The CLI or Native Interface Port of the JBoss Application Server is open.
- ❑ RSA Access Manager Servers are installed.
- ❑ Data servers are installed and running with the appropriate RSA Access Manager Data Adapter.

- ❑ RSA Access Manager Administrative Console is installed.
- ❑ A read-write administrator account for the Agent exists in RSA Access Manager Server.

Installation Modes

The Agent installation program provides different modes for installing the Agent on the required platforms. You need to choose the appropriate installation mode for installing the Agent.

- GUI Mode. The Agent installation program provides a GUI to install the Agent on both Windows and supported UNIX platforms.
  To run the installation program on UNIX, you must have X terminal installed and configured.
- Console Mode. The Agent installation program provides a console-based interface to install the Agent on supported UNIX platforms. To install in console mode, execute the Setup.bin program with "-i console" as the command line argument. For example: $./Setup.bin -i console.

Install the Agent

You perform the same installation steps to install the Agent on Windows and Linux platforms. On Linux, log on as root user to perform the installation. On Windows, log on as Local Administrator with "Act as part of the operating system" privileges.

Note: Before installing JBoss agent on windows, ensure that you uninstall all versions of Visual C++ Redistributable, to avoid the lockbox creation failure.

To install and configure RSA Access Manager Agent 5.0 for RedHat JBoss Application server v7.x using GUI or CLI, perform the following steps:

Note: During installation, click **Next** in GUI mode or press ENTER in console mode to proceed to the next screen.

1. Start your JBoss Application Server
2. Start the RSA Access Manager servers
3. On the customer location of RSA website from where you have permission, download the appropriate JBoss Application agent 5.0 package and extract the contents to a local directory. Double-click **setup.exe** to start the installer.
4. In the **Welcome** screen, read the prerequistes and proceed to next screen.
5. In the **License Agreement** screen, select **I accept the terms of this license agreement,** and proceed to next screen.

Note: If you are using putty on HP-UX platform, change the remote character set to UTF-8. Go to **Change settings > translation** to change the UTF character set.

6. In the **Supported Components** screen, select one of the following options to protect the Application Server resources and proceed to next screen.
   - **Single Sign-On (SSO) and Java Authorization Contract for Containers (JACC)**. Select this option to install the Agent in stand alone mode, enabling protection of resources and Single Sign-on between Web Server Agent and Application Server Agent.
   - **Single Sign-On (SSO) only**. Select this option to install Agent enabling only Single Sign-on between Web Server Agent and Application Server Agent.
7. In the **Locations** screen, specify the **RSA Access Manager Agent Installation** location, **JBoss Application Server Installation** location, and proceed to next screen.

   Note: If you choose to specify a different installation location for the agent, ensure that the directory already exists and has enough disk space.

8. In the **JBoss Server Information** screen, enter the **CLI Port number** and the administrator **Username** and **Password** required to connect to the JBoss Application Server and proceed to next screen.
9. In the **Summary** screen, proceed to **Install.**
10. In the **Connection Type** screen, select the security level for network connection between Access Manager Servers and Application Server Agent. There are three types of network connections:
    a. **Clear.** In this type of connection, communication between RSA Access Manager Servers and Application Server Agent, is not encrypted.
    b. **Anonymous.** In this type of connection, communication between RSA Access Manager Servers and Application Server Agent, is encrypted but does not provide authenticity of the communicating entity.
    c. **Authenticated.** In this type of connection, communication between RSA Access Manager Servers and Application Server Agent, is encrypted over SSL with certificate based authentication between them.
11. If you have chosen **Authenticated** as a Connection Type, **Trusted CA Certificate is** selected.
    Enter following details to create **CA KeyStore**:
    a. In the **Keystore Format** drop-down menu, select the Keystore file format which holds the Trusted CA certificates.
    b. In the **File Name** field, enter the file name that is assigned to the CA keystore file. You can also use the **Browse** button to browse to the CA Keystore file location.
    c. In the **Passphrase** field, enter the passphrase to protect CA keystore.
    d. In the **Confirm Passphrase** field, confirm the entered passphrase.

12. On the **Connection Type** screen, enter the following details to create Private Keystore Key that holds the Private Key details used by RSA Access Manager Servers for selected authentication type:
    a. In the **Keystore Format** drop-down menu, select the Keystore file format.
    b. In the **File Name** field, enter the file name to be assigned to the private keystore file.
    c. In the **Passphrase field,** enter the passphrase to protect private keystore file.
    d. In the **Confirm Passphrase** field, confirm the entered passphrase.
    e. In the **Keystore Alias** field, enter Keystore Alias to be used as the private key by the RSA Access Manager server.
    f. In the **Passphrase** field, enter the passphrase to protect CA keystore.
    g. In the **Confirm Passphrase** field, confirm the entered passphrase.

After updating all the required fields, proceed to next screen.

13. In the **RSA Access Manager Server Information** screen, select the Dispatcher or Authorization Server to connect to Agent and proceed to next screen.
    – Enter the **Server Name/IP** and respective **Port Number**, and click **Add** to enter the list of Dispatcher or Authorization Servers.
    – If you need to remove a server from the list, select the server and click **Remove**.
14. Enter the Entitlement **Server Name/IP**, respective **Port Number**, and RSA Access Manager Administrator **Username** and **Password** required to connect to RSA Access Manager Server. Proceed to next screen.
15. In the **Agent Information** screen, enter the following information:
    – **JBoss Application Server Name** and **Cookie Domain Name**.
    – Select **FIPS** (Federal Information Processing Standard) 140-2 mode if the RSA Access Manager Servers are configured using FIPS mode.
      Do not select FIPS if the RSA Access Manager Servers are configured using Non-FIPS mode.
    – Proceed to next screen.
16. In the **Create Lockbox** screen, enter the following information
    – **File Location** to store the Lockbox file
    – **File Name** for the Lockbox file
      Specify the lockbox file name with **.clb** extension.
    – **Lockbox Passphrase** to protect Lockbox file
      The Passphrase must contain minimum 8 characters with at least one uppercase, one lowercase, one numeric, and one special character
    – Proceed to next screen.
17. In the **Configuration Summary** screen, proceed to next screen.
18. On the **Finish** screen, select **Done** based on the installation completion note.

Note: Based on your success or failure of installation, status is shown as Congratulations or Failed.

Verification and Next Steps

Depending on your requirements, you may need to take additional steps to set up SSL connections and authentication options. These additional steps are described in the Chapter 4, JBoss Configuration and Deployment.

Note: Ensure that you perform the steps in Configure JBoss Application Server on page 31 to complete the installation.

The following sections guide you through verifying the settings that were configured during installation.

- Verifying Directories and Files on page 25
- Verifying XML files on page 27
- Verifying JBoss Environment on page 28
- Verifying cleartrust.properties Settings on page 28

Verifying Directories and Files

In the standalone environment, verify that the following directories and files are present:

- AGENTBASE/tools/properties directory:
  - axm_groups_roles.properties
  - cleartrust.properties
  - debugdisabled.properties
  - debugenabled.properties
  - log4j.properties
- AGENTBASE/tools directory:
  - cacheflush.bat(sh)
  - ctencrypt.bat(sh)
  - lockbox-tool.bat(sh)
- AGENTBASE/lib/com/rsa/axm/jboss/main directory:
  - axm-jboss-agent-api-5.0.jar
  - module.xml

- AGENTBASE/lib/com/rsa/cleartrust/main
  - axm-admin-api-6.2.jar
  - axm-appagent-common-5.0.jar
  - axm-runtime-api-6.2.jar
  - cryptojce-6.1.jar
  - cryptojcommon-6.1.jar
  - CSP-3.1.jar
  - CSPJNI-3.1.jar
  - jcm-6.1.jar
  - jcmFIPS-6.1.jar
  - LB-3.1.jar
  - LBJNI-3.1.jar
  - log4j-1.2.9.jar
  - module.xml
- AGENTBASE/webapps
  - jboss-web.xml
  - jboss-deployment-structure.xml
  - axm-jboss-agent-filterui-5.0.war
  - axm-jboss-agent-cacheflush-ejb-5.0.jar
- AGENTBASE/cst directory
- AGENTBASE/jre directory
- AGENTBASE/Uninstall_jbossAgentv5 directory
- AGENTBASE/webapps directory
- AGENTBASE/lockbox directory
- JBOSS_HOME/standalone/deployments directory
  - axm-jboss-agent-filterui-5.0.war
  - axm-jboss-agent-cacheflush-ejb-5.0.jar
- JBOSS_HOME/bin
  - standalone.conf.bat
- JBOSS_HOME/bin/properties directory:
  - axm_groups_roles.properties
  - cleartrust.properties
  - debugdisabled.properties
  - debugenabled.properties

  - log4j.properties
- JBOSS_HOME/modules/com/rsa/axm/jboss/main directory:
  - axm-jboss-agent-api-5.0.jar
  - module.xml
- JBOSS_HOME/modules/com/rsa/cleartrust/main directory
  - axm-admin-api-6.2.jar
  - axm-appagent-common-5.0.jar
  - axm-runtime-api-6.2.jar
  - cryptojce-6.1.jar
  - cryptojcommon-6.1.jar
  - CSP-3.1.jar
  - CSPJNI-3.1.jar
  - jcm-6.1.jar
  - jcmFIPS-6.1.jar
  - LB-3.1.jar
  - LBJNI-3.1.jar
  - log4j-1.2.9.jar
  - module.xml
- JBOSS_HOME/modules/javax/security/jacc/api/main directory
- JBOSS_HOME/modules/org/jboss/as/security/main directory

Verifying XML files

Verify the following entries in JBOSS_HOME\standalone\configuration\ standalone.xml file:

```
<security-domain name="axm-auth" cache-type="default">
- <authentication>
  <login-module code="com.rsa.axm.jboss.jaas.RSAJLoginModule" flag="required" />
  </authentication>
- <authorization>
- <policy-module code="com.rsa.axm.jboss.auth.modules.RSAJAuthorizationModule"
flag="required">
  <module-option name="delegateMap"
value="WEB=com.rsa.axm.jboss.auth.modules.RSAWebJPolicyModuleDelegate,EJB=com.rs
a.axm.jboss.auth.modules.RSAEjbJPolicyModuleDelegate" />
  </policy-module>
  </authorization>
  </security-domain>
  </security-domains>
```

Verifying JBoss Environment

Verifying Single Sign-On Configuration

To verify Single Sign-On configuration for JBoss:

1. In the JBoss Administrator console, select **Profile** tab.
2. On the navigation tree, click **Profile > Security > Security Domains.**
3. Ensure that the **axm-auth** security domain is included.

Verifying RSA Access Manager Login Application

To verify login application:

1. In the JBoss Administrator console, select **Runtime** tab.
2. On the navigation tree, click **Server > Manage Deployments.**
3. Ensure that **axm-jboss-agent-filterui-5.0.war** is deployed and enabled.

Verifying Java Authorization Contract for Containers (JACC) Provider

To verify authorization contract for containers (JACC) on JBoss:

1. In the JBoss Administrator console, select **Runtime** tab.
2. On the navigation tree, click **Server > Configuration.**
3. Select the **Environment properties** sub tab
4. Ensure that **javax.security.jacc.PolicyConfigurationFactory.provider=com.rsa.axm.jboss.jacc.RSAJPolicyConfigurationFactory** and **javax.security.jacc.policy.provider=com.rsa.axm.jboss.jacc.RSAJDelegatingPolicy** key value pair properties are present.

Verifying cleartrust.properties Settings

Verify whether the cleartrust.properties file located in the AGENTBASE/tools/properties directory contains the following properties with appropriate values that are given during the Agent installation:

- cleartrust.agent.ssl.use
- cleartrust.agent.auth_server_list
- cleartrust.agent.dispatcher_list
- cleartrust.agent.adapi.ssl
- cleartrust.agent.adapi.host

- cleartrust.agent.adapi.port
- cleartrust.agent.adapi.user_id
- cleartrust.agent.adapi.user_password
- cleartrust.agent.ssl.ca.keystore_file
- cleartrust.agent.ssl.ca.keystore_type
- cleartrust.agent.ssl.ca.keystore_provider
- cleartrust.agent.ssl.ca.keystore_passphrase
- cleartrust.agent.ssl.private.keystore_file
- cleartrust.agent.ssl.private.keystore_type
- cleartrust.agent.ssl.private.keystore_provider
- cleartrust.agent.ssl.private.keystore_passphrase
- cleartrust.agent.ssl.private.key_alias
- cleartrust.agent.ssl.private.key_passphrase
- cleartrust.agent.server_name
- cleartrust.agent.web_filter_enable
- cleartrust.agent.default_auth_mode
- cleartrust.agent.auth_resource_list
- cleartrust.agent.url_exclusion_list
- cleartrust.agent.extension_exclusion_list
- cleartrust.agent.sso
- cleartrust.agent.cookie_domain
- cleartrust.agent.path
- cleartrust.agent.enable_cache

Uninstall the Agent

To uninstall the JBoss Agent on Windows or Linux

1. Log on to the machine as admin.
2. Stop your application server.
3. For GUI-based uninstallation, execute the **Uninstall_jbossv5.exe** under *Agent installation directory/Uninstall_jbossAgentv5*.
4. For console-based uninstallation, in the command prompt, navigate to the *Agent installation directory/Uninstall_jbossAgentv5* directory and execute **Uninstall_jbossv5.bin**.

4 JBoss Configuration and Deployment

Configure JBoss Application Server

You must copy the following files to the **JAVA_HOME/jre/lib/ext** directory, to configure the JBoss Application Server:

- AGENT_HOME/lib/com/rsa/cleartrust/main/cryptojce-6.1.jar
- AGENT_HOME/lib/com/rsa/cleartrust/main/cryptojcommon-6.1.jar
- AGENT_HOME/lib/com/rsa/cleartrust/main/jcm-6.1.jar

If you have selected FIPS mode during installation, replace the jcm-6.1.jar in the JAVA_HOME/jre/lib/ext directory with AGENT_HOME/lib/com/rsa/cleartrust/main/jcmFIPS-6.1.jar

Note: If the java extension library points to a folder other than the standard java extension library, place the jar files in that location. You can also set the java system argument -Djava.ext.dirs to the folder where the jar files are located.

Configure JBoss Deployment Descriptors

You must configure the following agent specific deployment descriptors for every web application deployed in JBoss application server.

- jboss-web.xml

```
<?xml version="1.0"?>
<jboss-web>
<security-domain>axm-auth</security-domain>
<use-jboss-authorization>true</use-jboss-authorization>
<valve>
<class-name>
com.rsa.axm.jboss.webfilter.RSAJFilterValve
</class-name>
<!--
<class-name>
 com.rsa.axm.jboss.sso.authenticators.RSASSOBasicAuthValve
</class-name>
<class-name>
com.rsa.axm.jboss.sso.authenticators.RSASSOFormAuthValve
</class-name>
<class-name>
com.rsa.axm.jboss.ws.webfilter.RSAJFilterValve
</class-name>
 -->
</valve>
</jboss-web>
```

- jboss-deployment-structure.xml

```
<jboss-deployment-structure>
<deployment>
    <dependencies>
       <module name="com.rsa.axm.jboss" />
       <module name="com.rsa.crypt" export="true"/>
       <module name="com.rsa.cleartrust" />
    </dependencies>
</deployment>
</jboss-deployment-structure>
```

For more information on deployment descriptors used in JBoss application server, refer to the JBoss Documentation at **http://www.jboss.org/jbossas/docs**.

Configure the RSA Access Manager Agent

All of the Agent configuration parameters are stored in a text file, cleartrust.properties, which is read at startup by the system components. The configuration file provides parameter descriptions, allowed values, and examples. This file is located in the properties directory under the JBoss User Profile folder.

Though a standard installation automatically populates the required values of cleartrust.properties for deployment, you can edit this file to change the configured properties to meet your deployment requirements. Each time you make changes to cleartrust.properties, you must restart JBoss in order to properly load the new settings.

Parameter Settings

The parameters listed in this section must be set during installation in order to start and operate the Agent successfully:

- cleartrust.agent.ssl.use. This property specifies the mode of connections to the RSA Access Manager Servers. The allowed values for this property are Clear, Anon, and Auth. This setting must match the settings for cleartrust.net.ssl.use in your aserver.conf and dispatcher.conf files.

For authenticated SSL connections, you must configure additional parameters as described in cleartrust.agent.dispatcher_list. This is a comma-separated list of RSA Access Manager Dispatcher Servers. Each Dispatcher Server is expressed in the format hostname:listener_port or ipaddress:listener_port. The default listener port for the Dispatcher Server is 5608.

- cleartrust.agent.adapi.host. This property specifies the IP address (or hostname) of the RSA Access Manager Entitlements Server.
- cleartrust.agent.adapi.port. This property specifies the port number of the Entitlements Server. The default value is 5601.
- cleartrust.agent.adapi.ssl. This property specifies the mode of connection to make to the Entitlements Server. The allowed values for this property are Clear, Anon, and Auth. This parameter and the value for the property cleartrust.eserver.api_port.use_ssl in the eserver.conf file of the Entitlements Server must match.

Note: If you select Auth SSL mode in either or both of these settings, you must provide the keystore information listed in Required Information on page 20 and you must provide a valid keystore for the JBoss server. For more information, see Providing Keystore Files for the Agent on page 39.

For authenticated SSL connections, you must configure additional parameters as described in Set up Authenticated SSL on page 38.

- cleartrust.agent.adapi.user_id. This parameter specifies the Entitlements Server administrator's user name. The value of this property is encrypted. To create an encrypted user name, see Encrypted Parameters on page 34.
- cleartrust.agent.adapi.user_password. This parameter specifies the Entitlements Server administrator's password. The value for this parameter is encrypted. To create an encrypted password, see Encrypted Parameters on page 34.

Additional Parameters:

- cleartrust.agent.max_open_connections. This parameter specifies the maximum number of open server connections that the Agent must maintain for a server pool.
- cleartrust.agent.rtapi.retry_count. This parameter specifies the number of times the Runtime API must attempt to connect to an RSA Access Manager Authorization Server before deciding whether the server is dead or not.
- cleartrust.agent.auth_server_mode. This parameter specifies the mode by which the Agent connection pools route outgoing requests to the Authorization Servers. If set to STANDARD, all requests are routed to a single Authorization Server until a failure occurs. If a failure occurs, requests are routed to the next server in the list. If set to DISTRIBUTED, requests are routed in a sequential style to all available Authorization Servers.
- cleartrust.agent.adapi.admin_group. This parameter specifies the administrative group of the Entitlements Server administrative user.
- cleartrust.agent.adapi.admin_role. This parameter specifies the administrative role of the Entitlements Server administrative user.

Encrypted Parameters

Sensitive information in the configuration file must be encrypted in order for the Agent and its tools to work properly. The Agent installer encrypts all the sensitive properties during installation. It is also possible to encrypt this information manually using the ctencrypt tool that is provided with the Agent. This command line tool takes input in the form of name=value pairs and displays the output as name=encrypted value pairs.

Sensitive parameters that need to be encrypted are:

cleartrust.properties

- cleartrust.agent.adapi.user_id
- cleartrust.agent.adapi.user_password
- cleartrust.agent.ssl.ca.keystore_passphrase
- cleartrust.agent.ssl.private.keystore_passphrase
- cleartrust.agent.ssl.private.key_passphrase

For examples, see Ctencrypt on page 78.

Server Pool Settings

These settings provide information about the RSA Access Manager Dispatcher and Authorization Servers, and define how the Agent communicates with these Servers. Most of the relevant parameters in cleartrust.properties begin with *.auth_server* and *.dispatcher*.

While it is possible to run an Agent with only one Dispatcher or Authorization Server in the pool, typical deployment scenarios demand that the Agent create a pool of multiple Authorization Servers. These connection pools are managed either "dynamically" by the Dispatcher, or "statically" according to lists that you define in the cleartrust.properties parameters *.dispatcher_list* and *.auth_server_list*.

When dynamic and static lists are used together in creating connection pools, dynamically managed Servers are sorted first. Static Servers are sorted in the order that you have listed them in *.auth_server_list*.

Note: Static Servers must be listed in IP address format, not hostname format, to interoperate with dynamic lists as described here. The Dispatcher identifies all dynamic Servers as IP addresses and performs a string comparison to the list of static Servers.

If the RSA Access Manager environment is using multiple Authorization Servers, the KeyClient.sec entries for each Authorization Server must be listed in the KeyServer.sec file on all key server machines. Older RSA Access Manager Agents obtain keys directly from the Key Server. In RSA Access Manager Agent 5.0 SP1, the Agent obtains keys from the Authorization Server. For details, see the *RSA Access Manager Servers Installation and Configuration Guide*.

The behavior of connection pools is further controlled by these settings:

- Location Class and Priority on page 35
- Standard or Distributed Connection Mode on page 36
- Pool Refresh on page 37

Location Class and Priority

You can configure the Agent to use a preferred set of Authorization Servers. This allows the Agent to use an Authorization Server in the same geographic location, minimizing the response time for user requests.

You have configuration options to specify a specific location class for each Authorization Server and RSA Access Manager-protected server. In each Agent configuration file you can specify preferred location classes by setting the *.location_class_priority* parameter. Servers without a class or belonging to a class not present in the location class preference list are placed in a separate, implicitly defined class of the lowest priority.

Servers within a connection pool are ordered and sorted by these settings for location class and priority. Within each class, dynamic Servers (those returned from Dispatchers) are sorted before static Servers. Static Servers are sorted in the order that they are listed in the *.auth_server_list* parameter. Dynamic Servers within the same priority class may be inserted in random order to prevent multiple Agents in standard mode from all using the same Server.

To configure location class and priority settings:

1. Install and configure all participating Agents and Servers.
2. Configure all grouped Authorization Servers with the same location class, as defined in aserver.conf by the *.location_class* parameter. Use the same location class name in all participating Agent and Server configuration files.
3. In the cleartrust.properties file for each Agent, specify the location class that the Agent must use. If you are using more than one location class, set the priority using *.location_class_priority* as shown in this example:

```
cleartrust.agent.location_class_priority=NY,LA,UK
```

4. Restart Agents and Authorization Servers for the settings to take effect.

Important: The name you assign for the location class must be consistently used in all participating Agent and Server configuration files. Assigning location classes to Servers does not affect pooling in any way until you assign them a priority in cleartrust.properties.

Location Class and Priority Settings: Example

In this example, dN are dynamic Servers, sN are static Servers and A,B,C,… are location classes. Therefore, s3:F is static Server number three in class F.

The location class parameter sets priority as: B, C

Dispatcher returns Servers: d1:A, d2:B, d3:C, d4

Statically added Servers: s1:A, s2:B, s3:C, s4:C

Resulting list: d2:B, s2:B, d3:C, s3:C, s4:C, {d4, d1:A}, s1:A

The Agent opens connections to Servers in class B first, dynamic ones before static ones, then Servers in class C, and lastly any other classes. Dynamic Servers of the same priority can be sorted in any order (indicated by braces).

Connections are opened to all Servers in the list, unless the *max_open_connections* parameter has been set to a positive value, in which case at most that many connections are created, starting with the first Server in the list and stopping when a sufficient number of connections have been established.

Behavior with *max_open_connections* defined: Example

Using the same notation as the previous example, this example indicates Servers with open connections as underlined. It is assumed that all Servers can be successfully contacted.

The *max_open_connections* parameter is set to 3

Resulting list: d2:B, s2:B, d3:C, s3:C, s4:C, {d4, d1:A}, s1:A

Or, if s2 and d3 are unavailable:

List: d2:B, s2:B, d3:C, s3:C, s4:C, {d4, d1:A}, s1:A

If the same Server appears in both a static and a dynamic list, the dynamic entry is ignored. Also, if a static Server is added multiple times, only the first occurrence is used. Servers are considered identical if they have the same port number, the same hostname, and they belong to a class of the same priority. All classes not present in the priority list are treated as equal.

Standard or Distributed Connection Mode

Connection mode may be Standard or Distributed. In Standard mode, the Agent sends requests to the first available Server in the list. In Distributed mode, the Agent distributes requests among all available Servers, static and dynamic, in the first priority class with open connections.

An Agent, operating in distributed mode, chooses among the Servers in a given priority class the Server with the least number of outstanding packets. When all servers are equal in outstanding packets, it picks the Server with the lowest average response time. Under low loads, a fraction of the requests are distributed randomly among eligible Servers in order to keep the response time estimates updated and to select faster Servers.

Authorization Server connection mode is defined in the Authorization Server configuration file, aserver.conf. For more information, see the *RSA Access Manager Servers Installation and Configuration Guide* and the commenting in aserver.conf.

Pool Refresh

The only time the Agent opens new connections is during pool refresh. Pools are implicitly refreshed upon creation. Agents may explicitly refresh the pool at any time. RSA recommends forcing pool refreshes periodically, because it is the only time changes in Dispatcher Servers lists can be detected.

Pool refresh is a thread-safe operation that does not interfere with threads using existing open Server connections. Connections are not unnecessarily closed, nor are other threads blocked from using connections during the pool refresh.

If the pool has a Dispatcher list, Dispatchers are tried in the order listed, always starting with the first one and continuing until reaching one that supplies a non-empty list of Authorization Servers, of which at least one can be successfully contacted. Whether the Dispatcher connection is kept open between pool refreshes is undefined.

The Server list of the pool is updated with any changes received from the Dispatcher. New Servers are inserted into the list, ordered by class, and Servers no longer present in the Dispatcher list are closed and removed. Dynamic Servers already present in the list should not be re-randomized, to avoid unnecessary opening and closing of connections when *max_open_connections* limits the number of open connections to part of the class. New dynamic Servers are inserted in a random position among the dynamic Servers of the same class.

When there have been no failed connections since the last pool refresh, and the Dispatcher returns the same list of Servers as the last time, though possibly in a different order, no Server connections are created or closed. This is the normal case in a stable production environment. If none of the dynamic Servers returned from a Dispatcher can be successfully contacted, that list is ignored, and any existing open connections to dynamic Servers are not closed. Any existing connections in the pool are verified by sending a test request. If a connection is determined inoperable, it is closed and eligible to be reopened.

Additional Settings

General information is provided here for many of the Agent features. More detailed information is provided for some of the major features. Read the Agent configuration file to familiarize yourself with all of the individual parameters.

Cookie handling. Define cookie lifetime, idle time, expiration, and so on. These are standard cookie handling parameters. For more information, see cleartrust.properties.

Proxy settings. Define how to handle web servers that reside behind a proxy server. See Proxy Environments on page 53

User properties. Define whether to allow custom user properties defined in the RSA Access Manager Administrative Console to be published in HTTP headers.

Debug settings. Agent can be configured to print debug information in the server.log file. For more information, refer to the cleartrust.agent.debug_configuration parameter in cleartrust.properties.

Authentication settings. Define the type of authentication to be used: Basic, Windows NT, Certificate, RSA SecurID, and Custom. The default is Basic.

URL retention settings. Define how original URLs must be retained when users are redirected to the logon forms for authentication. See URL Retention on page 55.

URL and file extension exclusion settings. Define which URLs and file extensions are to be excluded from access control checks. By default, .gif, .jpg, .png, .ico, and .css file formats are excluded from access control checks to enhance Agent performance. See .url_exclusion_list and .extension_exclusion_list in cleartrust.properties.

Cache settings. Configuring cache settings can help reduce the time between when a user requests a protected resource and when that resource is accessed. See Chapter 9, Caching and Performance Tuning.

SSO. Determine whether to use single sign-on to allow users, once authenticated, to access additional protected resources without reauthenticating. See Configure SSO on page 60

Logon information. Define the location of your end-user authentication logon forms: initial logon pages, logon error message pages, inactive or expired account message pages, user lockout pages, access denied pages, server error pages, and invalid certificate message pages. Look for *_location.

Set up Authenticated SSL

If your RSA Access Manager Entitlements and Authorization/Dispatcher Servers are configured for authenticated SSL connections with clients, then the Agent must also be configured to use authenticated SSL. The mode of these SSL connections in the RSA Access Manager Servers is governed by these parameters:

- cleartrust.net.ssl.use in aserver.conf
- cleartrust.net.ssl.use in dispatcher.conf
- cleartrust.net.ssl.use in eserver.conf
- cleartrust.eserver.api_port.use_ssl in eserver.conf for *Admin APIs* only

Make sure that you check these settings before setting up the Agent in authenticated SSL mode. Authenticated SSL setup on the Agent requires you to provide valid keystore and CA keystore files, and to set all relevant SSL parameters in cleartrust.properties. These procedures are detailed in the following sections:

- Providing Keystore Files for the Agent on page 39
- Setting SSL Parameters on page 39

Providing Keystore Files for the Agent

To enable authenticated SSL between the Agent and the RSA Access Manager Servers, you must generate a private keystore file and a CA keystore file for the Agent, or copy valid existing keystore files into a directory.

If you prefer to generate keystore files for the Agent, use the certool utility provided with your RSA Access Manager Servers installation. This tool interoperates with RSA Keon to generate certificates and keystores used for configuring Intercomponent Security in the RSA Access Manager Servers. Guidance for using this certool utility is provided in the Servers Installation and Configuration Guide.

Alternately, you may copy the existing keystore files from the **conf** directory of your RSA Access Manager Servers. The Agent can use the same CA keystore and private keystore files as a Server.

Setting SSL Parameters

The standard installation prompts you to set all SSL parameters if you select Authenticated (Auth) as your SSL mode. If you have opted for manual configuration, you must set these parameters to enable authenticated SSL:

cleartrust.agent.ssl.ca.keystore_file. The filename that contains trusted CA certificates. This name must be an absolute path to the filename.

cleartrust.agent.ssl.ca.keystore_type. The type of CA keystore, either JKS or PKCS12. This parameter defaults to PKCS 12 if the parameter is commented out.

cleartrust.agent.ssl.ca.keystore_provider. The provider of the keystore algorithm used for unlocking and using the CA keystore. This parameter defaults to the RSA PKCS#12 provider if the parameter is commented out. If ca.keystore_type is JKS, you must set this parameter to SUN.

cleartrust.agent.ssl.ca.keystore_passphrase. The password to unlock the CA keystore. Passwords are case sensitive. The value of this parameter must be encrypted. For details about encrypted parameters, see Encrypted Parameters on page 34.

cleartrust.agent.ssl.private.keystore_file. The filename that contains the private key and corresponding certificate used to establish authenticated SSL connections. This name must be a filename with absolute path.

cleartrust.agent.ssl.private.keystore_type. The type of private keystore, either JKS or PKCS 12. This parameter defaults to PKCS 12 if the parameter is commented out.

cleartrust.agent.ssl.private.keystore_provider. The provider of the keystore algorithm used for unlocking and using the private keystore. This parameter defaults to the RSA PKCS#12 provider if the parameter is commented out. If ca.keystore_type is JKS, you must set this parameter to SUN.

cleartrust.agent.ssl.private.keystore_passphrase. The password to unlock the private keystore. Passwords are case sensitive. The value of this parameter must be encrypted. For details about encrypted parameters, see Encrypted Parameters on page 34.

cleartrust.agent.ssl.private.key_alias. The alias of the private key present in the private keystore.

cleartrust.agent.ssl.private.key_passphrase. The password to unlock the private key specified by the cleartrust.agent.ssl.private.key_alias in the private keystore. Passwords are case sensitive. The value of this parameter must be encrypted. See Encrypted Parameters on page 34.

Protect Web Applications

The RSA Access Manager web filters provide enhanced authentication and authorization support for web applications. The protected resources are centrally managed through the RSA Access Manager Administrative Console. The advantage of web filter-based protection of web applications as opposed to the J2EE role-based protection is that the web filters support more authentication types, such as:

- Basic (form and non-form-based)
- Windows NT (form and non-form-based)
- RSA SecurID (form-based)
- Custom (form-based)
- Certificate

Web filters are characterized by the following features:

- URL exclusion
- Extension exclusion
- Multiple authentication support
- Cookie exclusion support
- Cookie IP check
- Cookie IP check exclusion support
- Enhanced caching support (security parameters only)

Setting Up Authentication Using Web Filters

Authentication verifies user credentials before allowing a user to access a protected resource. RSA Access Manager provides its own internal authentication type by validating the user ID and password against the RSA Access Manager data store. RSA Access Manager also interoperates with several external authentication types, such as Microsoft NT PDC and X.509 certificates. For added security, you can combine various authentication types for specific resources.

Basic, Windows NT, RSA SecurID authentication types are handled by the RSA Access Manager Authorization Servers and are logged by the RSA Access Manager system according to the logging preferences you define when configuring your RSA Access Manager Servers. The RSA Access Manager Runtime API can be used to extend the functionality of these authentication types in the Authorization Server. For more information, see the *RSA Access Manager Developer's Guide*.

Custom authentication (using the Authorization Server) and Certificate authentication are performed at the Agent.

The authentication types use logon forms to present web pages to the user for authentication. You can customize the forms so that they have the same look and feel as your other company web pages. The *.form_based_enabled* parameter in cleartrust.properties tells the Agent to present web pages to the user, while the *.login_** parameters specify the location of the web pages to be presented.

Configuring Applications for Authentication

To configure applications for authentication using servlet filter or valve:

1. Add the following lines to the jboss-web.xml file under the WEB-INF folder in the web application that you want to protect with an RSA Access Manager web filter.

For Servlet Filter:

```
<filter>
<filter-name>CTLoginFilter</filter-name>
<filter-class>com.rsa.axm.jboss.webfilter.CTLoginFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>CTLoginFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

For Valves:

```
<jboss-web>
<security-domain>axm-auth</security-domain>
<valve>
<class-name>com.rsa.axm.jboss.webfilter.RSAJFilterValve</class-name>
</valve>
</jboss-web>
```

You can modify the jboss-web.xml file manually to add filter and filter-mapping sections into the jboss-web.xml file.

2. Add the JBoss server as an Enhanced Application Server in RSA Access Manager.
3. Protect the URL of the web application in RSA Access Manager. In order to protect all the web resources under the web application, add the entire context root of that web application as protected resources in RSA Access Manager. While adding the resource to RSA Access Manager, select the Resource Type as **Enhanced App Server Resource (J2EE)** and the J2EE Type as **Web Resource (JSP, HTML, Java servlet, static web resources**).
4. Install the RSA Access Manager web filter-enabled application into the application server.

Configuring Authentication Types

Authentication types are defined in cleartrust.agent.auth_resource_list in cleartrust.properties. The cleartrust.agent.default_auth_mode parameter defines which authentication type to use if none has been defined in the Auth resource list for a protected resource. The default for these parameters is Basic.

RSA Access Manager provides support for the following authentication types:

In all cases, the configured authentication type prompts the user attempting to access an RSA Access Manager-protected resource to provide the appropriate identification credentials. If the authentication type accepts the credentials, RSA Access Manager then checks for the user's authorization privileges on the requested resource.

Basic Authentication

Basic authentication validates the user ID and password provided at logon with the user account information in the RSA Access Manager data store. This is the default authentication type for all RSA Access Manager-protected resources.

See these parameters in cleartrust.properties: cleartrust.agent.auth.resource_list, cleartrust.agent.default_auth_mode, cleartrust.agent.form_based_enabled, and cleartrust.agent.login_*.

Windows NT Authentication

This authentication type supports authenticating users against their Windows NT domain accounts. For each user, an account with the same Windows NT user name must also exist in the RSA Access Manager data store.

This type also requires that you set this value in the aserver.conf file on your RSA Access Manager Authorization Server in the following format:

cleartrust.aserver.nt_domain_controllers=PDC,BDC

Set this parameter to a comma-separated list of NetBIOS names of your Windows NT primary domain controllers (PDC) and backup domain controllers (BDC). Restart the Authorization Server.

If user ids are not guaranteed to be present in all domains, then the mapping between the user ids can be done using the following parameter in the aserver.conf file.

cleartrust.aserver.authn.ntlm_name_mapping=true

See these parameters in cleartrust.properties: cleartrust.agent.auth.resource_list, cleartrust.agent.default_auth_mode, cleartrust.agent.form_based_enabled, and cleartrust.agent.login_*.

NTLM Authentication

This authentication type supports authenticating users against their Windows NT domain accounts. This type also requires that you set this value in the aserver.conf file on your RSA Access Manager Authorization Server in the following format:

cleartrust.aserver.ntlm_domain_controllers=PDC,BDC

Set this parameter to a comma-separated list of NetBIOS names of your Windows NT primary domain controllers (PDC) and backup domain controllers (BDC).

If desired user ids does not present in both NT domain and RSA Access Manager datastore, set the following parameters in the aserver.conf file:

cleartrust.aserver.authn.ntlm_name_mapping=true

For more information on how to run the mapping tool, see the *RSA Access Manager Servers Installation and Configuration Guide*.

See these parameters in cleartrust.properties: cleartrust.agent.auth.resource_list, cleartrust.agent.default_auth_mode, cleartrust.agent.form_based_enabled, and cleartrust.agent.login_*.

RSA SecurID Authentication

This authentication type supports RSA SecurID two-factor authentication to validate a user's user name and passcode at logon against the credentials stored in the RSA Authentication Manager. For each user, an account with the same RSA Authentication Manager user name must also exist in the RSA Access Manager data store. A passcode is a combination of a user's PIN and RSA SecurID valid tokencode entered as one continuous string. If the passcode is valid, the RSA Authentication Manager returns the request to the RSA Access Manager Authorization Server for access control checking.

See these parameters in cleartrust.properties: cleartrust.agent.auth.resource_list, cleartrust.agent.default_auth_mode, cleartrust.agent.form_based_enabled, and cleartrust.agent.login_*.

Custom Authentication

You can use the RSA Access Manager Runtime API to create your own customized authentication routine and create your own error messages and logging.

Setting Up Custom Authentication

In order to use your own customized authentication routine, you need to configure both the Agent and the RSA Access Manager Authorization Server to accept the new Custom authentication type.

To set up custom authentication:

1. Define your Custom authentication type by implementing the Authenticator interface provided in the Java Runtime API included in the RSA Access Manager Server package.
2. Build your Custom authentication Java file and create a JAR file.
3. Place your Custom authentication JAR file in the RSA Access Manager Server's /lib directory.
4. Specify your Custom authentication file in the aserver.conf parameter, cleartrust.aserver.customauth.classes.
5. Specify your Custom authentication class name in the Agent parameter .custom_auth. The name must exactly match the value returned by the method getAuthenticationType in the Custom authentication class. For example:

cleartrust.aserver.customauth.classes=sirrus.runtime.customauth.SampleAuth

Important: You must list multiple Custom authentication classes and names in the same order in the corresponding Server and Agent list parameters. Values in the aserver.conf parameter cleartrust.aserver.customauth.classes and the parameter cleartrust.agent.custom_auth in cleartrust.properties must match each other exactly.

6. Specify the resources that require Custom authentication in the Agent parameter cleartrust.agent.auth_resource_list, using the Custom authentication name. See the example in step 7.
7. Specify the location of form pages for Custom authentication, as shown in this example configuration for two types named "MyAuth" and "SampleAuth":

   cleartrust.agent.custom_auth=MyAuth,SampleAuth
   cleartrust.agent.auth_resource_list=/page1.html=MyAuth,/page2.html=SampleAuth
   cleartrust.agent.login_form_location_custom1=/ctappagent/ct_logon_custom1.html
   cleartrust.agent.login_form_location_custom2=/ctappagent/ct_logon_custom2.html

For details on using the Authenticator interface, see the *RSA Access Manager API Javadocs*.

See these parameters in cleartrust.properties: cleartrust.agent.custom_auth, cleartrust.agent.auth.resource_list, cleartrust.agent.default_auth_mode, cleartrust.agent.form_based_enabled, and cleartrust.agent.login_*.

Certificate Authentication

Certificate authentication supports X.509 certificates for authentication, such as those issued by the RSA Keon Certificate Authority. Setting up Certificate authentication varies depending on the certificate authority (CA) and application server you are using. See the *Application Server* documentation.

Note: Whenever a user's certificate DN is entered into RSA Access Manager, make sure that the DN entry matches what is returned by the application server (or web server if the application server is front-ended by a web server). For example: *CN=myName, OU=myOrgUnit, O=myOrg, L=myLocality, ST=myState, C=myCountry.*

Prerequisites

You must have an identity and CA root certificate installed on your JBoss server. For details, see the JBoss documentation at **http://www.jboss.org/jbossas/docs**.

To set up Certificate authentication:

1. Configure the JBoss Application Server to accept HTTPS requests from client browsers or applications. The application server must be configured to require client certificates.
2. Define which resources require Certificate authentication, and specify CERTIFICATE for those resources in cleartrust.properties. See cleartrust.agent.auth_resource_list and cleartrust.agent.default_auth_mode.

3. Configure your client machines for certificates. The user must be issued a browser certificate or personal certificate from the CA, which must then be installed and configured in the browser. For details, see your web browser documentation.
   - The distinguished name (DN) of the client-side certificate must match the DN in the user's account in the RSA Access Manager data store. The cleartrust.data.ldap.user.attributemap.certdn: dn parameter in ldap.conf specifies the LDAP attribute that stores the user's certificate DN.
   - If the user entry DN in the LDAP directory is not the same as the DN of the user's browser certificate, change the LDAP attribute from DN to ctscUserDN, or to another modifiable attribute in your user object class. In Active Directory, in order for Certificate authentication to work, the selected or added attribute must have a syntax type of "Unicode String" or "Directory String".
   - **For Microsoft Active Directory:** Note that the DN value specified absolutely must be valid within the DIT, due to the directory's active maintenance of referential integrity. According to Microsoft documentation, "If the referenced object is renamed or moved, Active Directory ensures that the attribute reflects the change. If the attribute is reset with a new DN, the attribute is referenced to the object represented by the new DN." For more information, see the DN mapping parameters in the Servers Installation and Configuration Guide.

Note: If the certificate DN contains the uid attribute, certain browsers convert uid to the object id number "0.9.2342.19200300.100.1.1". For example, if the user's DN is "uid=jsmith,ou=sales,o=rsa", the certificate DN in the browser is "0.9.2342.19200300.100.1.1=jsmith,ou=sales,o=rsa". In this case, enter "0.9.2342.19200300.100.1.1=jsmith,ou=sales,o=rsa" as the certificate DN value of the user in RSA Access Manager.

4. Restart your Entitlements and Authorization Servers after modifying the ldap.conf file.
5. Obtain the user's certificate DN and populate the DN field in the Administrative Console for each user. To do this, open each user's certificate (.cer or .crt file), copy the DN inside the certificate, and paste it into the DN field of the user entry in the Administrative Console. An example of a certificate DN is: cn=jsmith,ou=sales,o=rsasecurity,c=US.

See these parameters in cleartrust.properties: cleartrust.agent.auth.resource_list and cleartrust.agent.default_auth_mode. Also see the cleartrust.data.ldap.user.attributemap.certdn: dn parameter in ldap.conf.

Form-Based Authentication

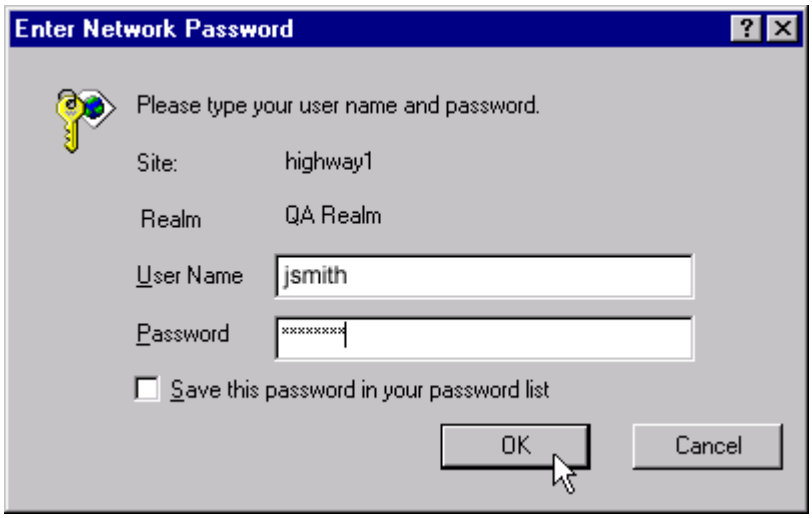
By default, form-based authentication is enabled when you install the Agent. Several HTML and JSP forms are included to use with the authentication types. These forms are presented to users when they request a protected resource. You can customize the logon forms (used to collect the user's identity credentials) to your own organizational identity, and customize what your users see after logon by directing them to a specific web page. When form-based authentication is enabled and the user does not have adequate entitlements to access the requested protected page, the user is redirected to an access denied page.

Authentication forms are set in cleartrust.properties, beginning with the cleartrust.agent.login_home_location. The HTML and JSP forms are installed as an application called **axm-jboss-agent-filterui-5.0.war** in JBoss. After authentication, the user can be directed to the original page or to a default home location. This is decided by the property cleartrust.agent.retain_url in cleartrust.properties.

Form-based authentication is required to use single sign-on (SSO). This enables the Agent to maintain the user's credentials across requests.

Important: When designating directories to protect, use "/*" with caution. The form pages you specify in cleartrust.properties are explicitly unprotected by RSA Access Manager. Using "/*" to protect the entire application server web resources blocks access to any graphics or objects associated with those pages, causing them to display or function improperly.

If you disable form-based authentication, users see an HTTP authentication prompt native to the web browser. Disabling form-based authentication is recommended only if you are using Basic authentication.



When form-based authentication is disabled and the user does not have adequate entitlements to access the requested protected page, the user is given the standard "HTTP 404 - File Not Found" error message.

Pointing to Logon Forms

The format used to point to the appropriate logon form depends upon the type of web page being used.

For example:

- HTML pages

/ctappagent/ct_logon_<%language%>.html

HTML pages are the default, except when using RSA SecurID authentication.

- JSP pages

/ctappagent/ct_logon.jsp?CTAuthMode=BASIC&language=<%language%>

Note that the placement of the language tag is different for JSP pages than for HTML pages. Also, the CTAuthMode tag is used. CTAuthMode can be any of these authentication types: BASIC, NT, SECURID, CERTIFICATE, and CUSTOM.

JSP pages are required when using RSA SecurID authentication. The cleartrust.properties file is already properly configured.

Using Multiple Authentication Types

Granular resource-based authentication allows you to configure different authentication modes for different resources on any given RSA Access Manager protected application server. The Agent supports any or all of the authentication types listed for different resources on the application server. You can configure a global authentication type used for all resources, and you can at the same time specify different authentication types for selected resources at the URL level. For example, you could require users to present a digital certificate to access specific financial resources, while allowing users to access other applications with their Windows NT user name and password.

```
cleartrust.agent.default_auth_mode=BASIC
cleartrust.agent.auth_resource_list=/apps/*=NT,/financial/*=CERTIFICATE
```

You can add as many specific URLs to the authentication resource list as you want, and each one can have a different authentication mode attached to it. The list is comma separated. If you do not specify a particular resource, the default authentication mode (.default_auth_mode) is used.

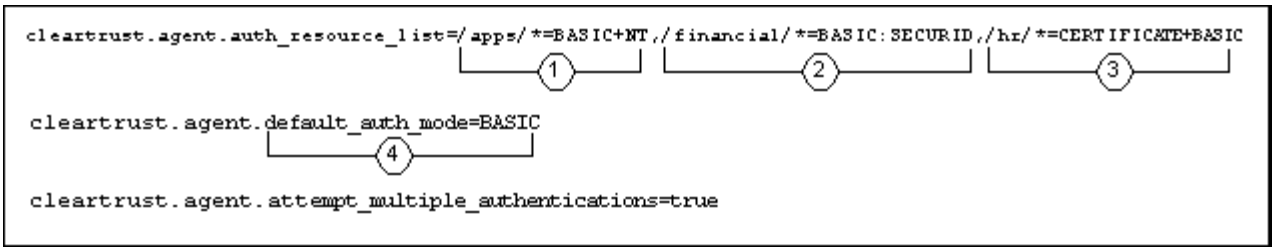
Note: If an invalid authentication type is specified in the .auth_resource_list parameter, the Agent starts up normally. However, when the user tries to access any resource in the web filter-protected application, the Agent throws a ServletError. Be sure to specify only valid authentication types in the .auth_resource_list parameter.

Multiple Authentication Chaining

It is possible to combine or chain together different authentication types for a particular resource. For example, the parameter settings in the following example dictate four different authentication requirements, each on different resources.

- Users trying to access anything in the /apps directory need to first provide Basic authentication, then Windows NT authentication.

- Users trying to access anything under /financial must supply either Basic or RSA SecurID authentication.
- Users trying to access anything in the /hr directory must have a digital certificate and provide Basic authentication.
- Any resources that are not explicitly mentioned in the parameter use the default, Basic authentication.



Rules

- cleartrust.agent.form_based_enabled must be set to true for multiple authentication.
- Boolean operators:
  - **AND.** The plus sign (+). Requires the user to fulfill both authentication types.
  - **OR.** The colon sign (:). Requires the user to fulfill any one of a set of authentication types in a sequential manner. The user is initially challenged to enter the first authentication type in the list.
  - The .attempt_multiple_authentications parameter tells the Agent to try the user's credentials against all authentication types listed without re-prompting the user if the first authentication attempt fails. This parameter applies only if using the OR operator.
  - The .attempt_multiple_authentications parameter tells the Agent to try the user's credentials against all authentication types listed without re-prompting the user if the first authentication attempt fails. This parameter applies only if using the OR operator.
  - For any given URL, you may string together multiple authentication types that are handled in either an AND or an OR relationship.
  - If you have configured password lockout, all of a user's logon failures, regardless of authentication type, are counted in the same counter. For example, a failure using Basic authentication followed by a failure using RSA SecurID authentication results in the counter being set to two. RSA Access Manager does not interact with any logon failure counters that are resident in your underlying user authentication types. For more information on password lockout and how it functions, see the Administrative Console Help or the RSA Access Manager Administrator's Guide.

Setting Up Authorization

In order to protect a web application using RSA Access Manager web filter and grant access rights to users, the web resource must be created in RSA Access Manager Servers, and the user or the user's group must be entitled to access the resource.

To set up authorization for web filter-protected web resources:

1. Log on to the RSA Access Manager Administrative Console.
2. Click **Define Resources > Servers > Add New.**
3. Enter the Server Name. Enter the name given in the cleartrust.agent.server_name parameter in the cleartrust.properties file.
4. Select "Enhanced App Server" for the server type.
5. Select the Product Name from the Product drop-down list. If the required Product Name is not listed in the Product drop down list, you can add it through the enhancedAppManufacturers parameter in the admingui.cfg file.
6. Enter the hostname of the JBoss Application Server.
7. Click **Save.**
8. Click **Define Resources > Add Resources**.
9. Select the newly created "Enhanced App Server" as the Server Name from the drop-down list.
10. Click **Next.**
11. Select an application where you want to keep all your Application server-protected resources.
12. Click **Add Resource** and link the selected Application.
13. On the Add Resource page, specify the context root of the web application to be protected in the **Web Resource** field. For example, if the context root of the web application is "/stock", enter "/stock/*" in the **Web Resource** field.
14. Click **Save.**
15. Click **Done.**

Entitlements can now be granted for the users or groups who need to access the resource. For more details, see the *RSA Access Manager Servers* documentation.

Protect J2EE Resources

Primary responsibility for creating security policies lies with the administrators that develop and deploy a JBoss application. The protection of J2EE resources is determined by roles that are created as part of application development, and their mapping to the RSA Access Manager users and groups during and after the Agent deployment.

Basically, the developer defines what, and the deployment administrator defines who. The developer defines what resources must be protected and gives a name to the security roles protecting them. Then the deploying administrator defines who belongs to those roles by mapping them appropriately to actual users and groups.

After the Agent is installed, the assignment of roles to users or groups is managed through the RSA Access Manager Administrative Console.

Overview of Security Roles in JBoss Application Development

When building an application, a developer indicates which JBoss resources must be protected, and defines security roles controlling access to them. For instance, a banking application might require separate roles for managers and supervisors, in which managers are authorized to transfer money between accounts while supervisors are only allowed to view the accounts. In this case, the manager role is granted access to the J2EE resources that provide the transfer functionality, but the supervisor role is denied this access.

This assignment of roles is accomplished through manipulation of the application Deployment Descriptor. Details of these tools and procedures are provided in the JBoss documentation at **http://www.jboss.org/jbossas/docs**.

Creating JBoss Security Roles in RSA Access Manager

JBoss Application Server is based on the J2EE role-based security model. For RSA Access Manager to work with the JBoss server's role-based security framework, the Agent is designed to map JBoss roles to RSA Access Manager groups using the *axm_groups_roles.properties* located at JBOSS_HOME/bin/properties directory.

The RSA Access Manager Server supports a concept called "Roles" for delegated administration. These "Roles" are entirely different from the security roles defined in JBoss applications.

Note: If J2EE security roles are assigned to resources in the web application deployment descriptor, it is required to provide appropriate mapping in the axm_groups_roles.properties.

Mapping Conventions When JACC is Enabled

The Agent uses the following naming convention to map a JBoss security role to an RSA Access Manager group that represents the role, and vice versa, when JACC is enabled:

- *RoleName_ApplicationName_ServerName_CTROLE*

where:

- *RoleName* is the name of the security role that is defined in the application referenced by ApplicationName.
- *ApplicationName* is the logical name of the deployed application.
- *ServerName* is the unique name of the JBoss server that is represented in RSA Access Manager. This value must be the same as the cleartrust.agent.server_name property value in the cleartrust.properties file.

All the JBoss security roles in RSA Access Manager must end with a tag called CTROLE.

For example:

Consider a **Manager** role that is associated with an enterprise application called **BankingApp**. If the unique JBoss server name is **JBOSSServer**, then the RSA Access Manager group that the Agent refers to for this Manager role is **Manager_BankingApp_JBOSSServer_CTROLE**.

Mapping Users or Groups to Security Roles (Authorization)

After creating the required JBoss security roles as groups in RSA Access Manager, these security roles can be granted to any user or group in RSA Access Manager by adding the user or the group as a member of the newly mapped group.

Note: To perform JACC permissions checks, the user must belong to **JBossAdministrator** group.

For example:

To grant the JBoss Manager role to a user called Tom, add **Tom** as a member user of the RSA Access Manager group **Manager_BankingApp_JBOSSServer_CTROLE**, through the RSA Access Manager Administrative Console.

To grant the JBoss **Manager** role to an RSA Access Manager group called BankManagers, add BankManagers as a member group of the group **Manager_BankingApp_JBOSSServer_CTROLE**, through the RSA Access Manager Administrative Console. By doing this, all the users under the RSA Access Manager **BankManagers** group have the JBoss Manager role.

RSA Access Manager Agent 5.0 SP1 and later for JBoss, the Agent can perform role-based authorization for J2EE resources. The role-to-user or role-to-group mapping that is done during JBoss application deployment is not considered role-based authorization for RSA Access Manager. Only the roles that are mapped to user or groups in RSA Access Manager, as described in this section, are considered for role-based authorization.

Web Services

Web services in simpler terms are web-based enterprise applications that use open, XML-based standards, and transport protocols to exchange data with calling clients. In a typical web services scenario, a business application uses the SOAP protocol over HTTP/HTTPS to send a request to a service at a URL. The service receives the request, processes it, and returns a response. An often-cited example of a web service is that of a stock quote service, in which the request asks for the current price of a specified stock and the response returns the stock price.

Web services can be protected using J2EE roles or RSA Access Manager web filters.

Protecting WebService with J2EE Roles

Since web services are tunneled over HTTP/HTTPS protocol, they take advantage of the security infrastructure provided by the web container. J2EE role-based security can be applied to the web application that hosts a web service. In such a case, authentication and authorization is handled by RSA Access Manager through the Custom User Registry and the Authorization Table Provider.

Basic and Certificate authentication types are supported when using role-based security.

- Setting Up Role-Based Authentication
- Setting Up Role-Based Authorization

Setting Up Role-Based Authentication

1. Add the required roles to the Deployment Descriptor during the web service development.
2. If the web service is an EJB, roles can be applied to the individual methods. Otherwise roles can be applied to the whole web application. If method-level protection is needed and the web service is not an EJB, you can create an enterprise bean with methods matching the web service operations. These EJB methods perform no operation; they are only dummy entities for applying security. Existing JBoss authentication mechanisms can be applied to the enterprise bean. Before any web service operation is invoked, a call is made to the EJB method. If authorization is granted, the web service is invoked.

For more information about how to set up operation-level security for web services, see the JBoss documentation at **http://www.jboss.org/jbossas/docs**.

Setting Up Role-Based Authorization

Once roles have been defined in the application, the roles must be assigned to users or groups for authorization. To assign the roles that are defined in the application, follow the steps in Creating JBoss Security Roles in RSA Access Manager on page 50.

Protecting Web Services with RSA Access Manager Web Filter

Web services can also be protected using the RSA Access Manager web filters. The web filters used for protecting web services are different than those used to protect web applications. The web filter for web services is com.rsa.axm.jboss.ws.webfilter.RSAJFilterValve.

Basic, Certificate, and Windows NT authentication types can be used to protect web services when web filters are used.

Setting Up Web Filter-Based Authentication

To set up Web Filter-Based authentication:

1. Web services are invoked using a SOAP RPC router servlet that is mapped as a URL in the web application. To protect the web service, apply the web filter to the SOAP RPC router servlet. Add the following filter and filter-mapping tags to the deployment descriptor of the web application that hosts the web service.

   <filter-name>CTLoginFilter</filter-name>

```
<filter-class>com.rsa.axm.jboss.webfilter.CTLoginFilter</fil
ter-class>
</filter>
<filter-mapping>
<filter-name>CTLoginFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

2. Protect the SOAP RPC router in the RSA Access Manager Servers.

For example: If your context root of the application is “stockquote”, the URL is /stockquote/servlet/rpcrouter. Protect this URL in the RSA Access Manager Servers.

3. Specify the authentication type in cleartrust.properties. If the authentication type is not specified, the default authentication type configured in cleartrust.properties is used.

For example: cleartrust.agent.auth_resource_list=/stockquote/servlet/rpcrouter=NT

Setting Up Web Filter-Based Authorization

To set up Web Filter-Based authorization:

1. Log on to the RSA Access Manager Administrative Console.
2. Grant entitlements to the users or groups who are required to access the web service.

Proxy Environments

When using the Agent to protect application servers that reside behind a proxy server, you must consider some additional configuration parameters. The following settings are applicable only to the web resources residing in the application server that are protected by web filters.

IP Checking. Specifies whether to validate the IP address stored in the cookie to determine if it is coming from a trusted source. For users accessing protected web resources, through a proxy server, the IP address must be verified only once (by the Agent installed on the proxy server).

Returning Cookies to the Client. If you are using a proxy/web server to front several application servers, potentially two SSO cookies can be returned to the client machine: the cookie generated by the proxy/web server Agent and the cookie generated by the application server Agent. In proxy environments, configure the Agents on your application servers to set the cookie IP address to remain that of the client instead of the proxy server. For more information, see Returning Cookies to the Client on page 54.

IP Checking

The IP Checking feature includes:

- Excluding Certain Agents From the IP Check on page 54
- Excluding Proxy Servers and Firewalls from the IP Check on page 54
- Excluding All Agents Within the Same Class C Subnet From IP Check on page 54

Excluding Certain Agents From the IP Check

The IP check feature validates the IP address stored in the cookie to determine if it is coming from a trusted source. The Agent compares the IP address of the client host with the IP address stored in the cookie before it uses the cookie for SSO.

In a typical proxy implementation, the proxy server and all content servers are protected by RSA Access Manager Agents. Enable IP checking only on the proxy server. If IP checking is enabled on the content servers, when the content server Agent checks the originating IP address of an incoming request, it sees only the IP address of the proxy server or the firewall (if the content servers are behind a firewall), causing the IP check to fail.

By default, IP checking is enabled when the Agent is installed. See .cookie_ip_check in cleartrust.properties.

Excluding Proxy Servers and Firewalls from the IP Check

Frequently, your proxy servers send requests to the RSA Access Manager Authorization Servers. You may not want IP checking for these requests because they originate from a trusted host inside your network. To exclude a proxy server or firewall, list the IP address of the server in the .ip_check_exclusion_list parameter in cleartrust.properties on the application server. Separate the IP addresses of multiple servers with a comma. For example:

cleartrust.agent.ip_check_exclusion_list=111.222.33.44,111.22 2.33.45

Any requests that do not arrive by way of the proxy server or firewall are still checked for a valid originating IP address when .cookie_ip_check is enabled. This parameter is not read or used by the system if the .cookie_ip_check parameter is disabled.

Excluding All Agents Within the Same Class C Subnet From IP Check

In a 32-bit IP address, the number of bits used to identify the network and the host vary according to the network class of the address. In a Class C network, the first 3 bits, or the high-order bits, are always 110. The next 21 bits are used to define the Class C network, and the final 8 bits are used to identify the host.

Instead of listing all proxy servers and firewalls in the IP check exclusion list, you can use a more general setting that instructs the Agent to forgo IP checking for any request coming from an IP address on the same Class C subnet (or internal network) as the Agent. To do this, enable .allow_subnet_masking in cleartrust.properties. This parameter is not read or used by the system if the .cookie_ip_check parameter is disabled.

Returning Cookies to the Client

In a typical proxy setting, the Agent is installed on the proxy and content servers. As a result, when a user requests a protected resource, both the Agent on the proxy server and the Agent on the content server attempt to generate and return an SSO cookie. To avoid this redundancy, configure the content server Agents to suppress generation of the cookie. This ensures that the browser receives only the cookie generated by the proxy server Agent, preserving the authentication relationship between the browser and the proxy server Agent.

To suppress generation of the SSO cookie, add the IP addresses of all proxy servers and firewalls as a comma-separated list in the .cookie_exclusion_list parameter in the Agent configuration file of the content server. For example:

cleartrust.agent.cookie_exclusion_list=111.222.33.44,111.222.33.45

Publish User Properties

By taking the appropriate steps in your administrative tools and Agent configuration file, you can publish all available custom properties or only selected properties. Note these requirements:

- **Marking properties for export with administrative tools.** Each property to be exported must be marked as Export/Publish when the RSA Access Manager administrator creates or modifies the property in the Administrative Console or through a custom Administrative API program. For details, see the Administrative Console Help or the *RSA Access Manager Developer's Guide.*
- **Adding properties for export to the Agent configuration file.** Each property must be added to the cleartrust.agent.userproperties parameter in cleartrust.properties, unless the parameter is set to the wildcard "*", which publishes all properties marked for Export/Publish. The list is comma separated.

Cookie Compatibility

If you have Web Servers still running Agent 1.0 released with RSA ClearTrust 4.6.1, set this parameter in aserver.conf:

cleartrust.aserver.token_version=0

and this parameter in cleartrust.properties:

cleartrust.agent.cookie_name=ctrust-session-v002d

This tells RSA Access Manager Agent 5.0 SP1 to issue cookies in a format compatible with Agent 1.0. This configuration is not required to communicate with newer versions, which share a common format with RSA Access Manager Agent 5.0 SP1. If you are using RSA SecurID as your authentication type and have a mixed environment of Agents 3.0, 4.7, 4.7 SP1 along with RSA Access Manager Agent 5.0 SP1, you cannot use RSA SecurID system-generated PINs. User-defined PINs are supported.

URL Retention

The following settings are applicable only to the web resources residing in application servers that are protected by web filters.

When used with form-based authentication, URL retention enables the web server to retain original page requests instead of sending users to a default location (such as the home page) after they log in. URL retention involves redirecting and storing the original page request in a cookie or in a query string, as described in this outline of the process:

- When requesting a protected resource, the user is redirected to the appropriate logon page as specified in the .login_form_location_ parameter in cleartrust.properties.
- The original URL of the requested resource is stored in one of two ways: either in the encrypted session cookie, or in the login form query string as ct_orig_uri. This is specified in the parameter .retain_url.use_query_string.
- Once the user is authenticated, the user is redirected back to the originally requested protected resource.

Configuring URL Retention

To configure URL retention, changes are required in the cleartrust.properties file. The setup steps differ between cookie-based and query string-based URL retention. By default, cleartrust.properties is configured with cookie-based URL retention parameters.

To configure cookie-based URL retention:

1. Verify that these parameters in cleartrust.properties are enabled with the values shown:

   cleartrust.agent.form_based_enabled= True cleartrust.agent.retain_url= True cleartrust.agent.retain_url.use_query_string= False

2. Specify the location of the default home page. Cookie-based URL retention uses the HTML form provided in the RSA Access Manager logon application.

   cleartrust.agent.login_home_location= /ctappagent/ct_home_<%language%>.html where <%language%> is optional for this parameter.

3. Specify the location of the default logout page to display when a user ends an RSA Access Manager session.

   cleartrust.agent.logout_form_location= /ctappagent/ct_logout_<%language%>.html where <%language%> is optional for this parameter.

4. Specify the location of the logon pages to display to your users.

- **Basic** - Set the following parameters and associated logon forms:

  cleartrust.agent.login_form_location_basic= /ctappagent/ct_logon_<%language%>.html

  cleartrust.agent.login_error_user_location_basic= /ctappagent/ct_logon_failed_<%language%>.html

  cleartrust.agent.login_error_pw_location_basic= /ctappagent/ct_logon_failed_<%language%>.html

- **Windows NT** - Set the following parameters and associated logon forms:

  cleartrust.agent.login_form_location_nt= /ctappagent/ct_logon_nt_<%language%>.html

  cleartrust.agent.login_error_user_location_nt=

/ctappagent/ct_logon_failed_nt_<%language%>.html

cleartrust.agent.login_error_pw_location_nt=
/ctappagent/ct_logon_failed_nt_<%language%>.html

- **RSA SecurID** - Set the following parameters and associated logon forms. RSA SecurID always uses JSP versions of the logon pages.

  cleartrust.agent.login_form_location_securid=
  /ctappagent/ct_logon.jsp?CTAuthMode=SECURID&language=<%language%>

  cleartrust.agent.login_form_location_sid_nexttoken=
  /ctappagent/ct_securid.jsp?SecurIdMode=nexttoken&language=<%language%>

  cleartrust.agent.login_form_location_sid_passcode=
  /ctappagent/ct_securid.jsp?SecurIdMode=passcode&language=<%language%>

  cleartrust.agent.login_form_location_sid_newpin_user_selectable=
  /ctappagent/ct_securid.jsp?SecurIdMode=newpin&NewPinMode=userselectable

  cleartrust.agent.login_form_location_sid_newpin_sys_generated=
  /ctappagent/ct_securid.jsp?SecurIdMode=newpin&NewPinMode=cannotchoose

  cleartrust.agent.login_form_location_sid_newpin_user_specified=
  /ctappagent/ct_securid.jsp?SecurIdMode=newpin&NewPinMode=mustchoose

  cleartrust.agent.login_form_location_sid_newpin_display=
  /ctappagent/ct_securid.jsp?SecurIdMode=newpin&NewPinMode=display

  cleartrust.agent.login_error_sid_newpin=
  /ctappagent/ct_securid.jsp?SecurIdMode=newpin&NewPinMode=
  mustchoose&CTLoginErrorMsg=Invalid%20new%20pin

  cleartrust.agent.login_error_user_location_securid=
  /ctappagent/ct_logon.jsp?CTAuthMode=SECURID&CTLoginErrorMsg=
  Login%20Unsuccessful

  cleartrust.agent.login_error_pw_location_securid=
  /ctappagent/ct_logon.jsp?CTAuthMode=SECURID&CTLoginErrorMsg=Login%20Unsuccessful

  cleartrust.agent.login_error_location_securid=
  /ctappagent/ct_logon.jsp?CTAuthMode=SECURID&CTLoginErrorMsg=Login%20Unsuccessful

  SecurIdMode must be specified in the query string of the URL with one of the following values: nexttoken, newpin, or passcode. If SecurIdMode is not set, the default value is nexttoken.

- **Custom** - Set the following parameters and associated logon forms:

  cleartrust.agent.login_form_location_custom=
  /ctappagent/ct_logon_custom_<%language%>.html

cleartrust.agent.login_error_user_location_custom= /ctappagent/ct_logon_failed_custom_<%language%>.html

cleartrust.agent.login_error_pw_location_custom= /ctappagent/ct_logon_failed_custom_<%language%>.html

Note: Do not change the ct_logon.xxx page name. You can modify the contents of the file as long as you maintain ct_logon as the filename. Use %20 to define spaces in the previous arguments.

To configure query string-based URL retention:

1. Verify that these parameters are enabled with a value of "True" in cleartrust.properties, as shown:
   cleartrust.agent.form_based_enabled= True cleartrust.agent.retain_url= True
2. Change the value of .retain_url.use_query_string to "True", as shown:
   cleartrust.agent.retain_url.use_query_string= True
3. Specify the location of the default home page.
   cleartrust.agent.login_home_location=
   /ctappagent/ct_home.jsp?language=<%language%>
   where ?language=<%language%> is optional for this parameter.
4. Specify the location of the default logout page to display when a user ends an RSA Access Manager session.
   cleartrust.agent.logout_form_location=
   /ctappagent/ct_logout.jsp?language=<%language%>
   where ?language=<%language%> is optional for this parameter.
5. Specify the location of the logon pages to display to your users.

- **Basic** - Set the following parameters and associated logon forms:
  cleartrust.agent.login_form_location_basic=
  /ctappagent/ct_logon.jsp?CTAuthMode=BASIC&language=<%language%>

  cleartrust.agent.login_error_user_location_basic=
  /ctappagent/ct_logon.jsp?CTAuthMode=BASIC&CTErrorLoginMsg=
  Login%20Failed&language=<%language%>

  cleartrust.agent.login_error_pw_location_basic=
  /ctappagent/ct_logon.jsp?CTAuthMode=BASIC&CTErrorLoginMsg=
  Login%20Failed&language=<%language%>
- **Windows NT** - Set the following parameters and associated logon forms:
  cleartrust.agent.login_form_location_nt=
  /ctappagent/ct_logon.jsp?CTAuthMode=NT&language=<%language%>

  cleartrust.agent.login_error_user_location_nt=
  /ctappagent/ct_logon.jsp?CTAuthMode=NT&CTErrorLoginMsg=
  Login%20Failed&language=<%language%>

  cleartrust.agent.login_error_pw_location_nt=

/ctappagent/ct_logon.jsp?CTAuthMode=NT&CTErrorLoginMsg=
Login%20Failed&language=<%language%>

- **RSA SecurID** - Set the following parameters and associated logon forms:

  cleartrust.agent.login_form_location_securid=
  /ctappagent/ct_logon.jsp?CTAuthMode=SECURID&language=<%language%>

  cleartrust.agent.login_form_location_sid_nexttoken=
  /ctappagent/ct_securid.jsp?SecurIdMode=nexttoken&language=<%language%>

  cleartrust.agent.login_form_location_sid_passcode=
  /ctappagent/ct_securid.jsp?SecurIdMode=passcode&language=<%language%>

  cleartrust.agent.login_form_location_sid_newpin_user_selectable=
  /ctappagent/ct_securid.jsp?SecurIdMode=newpin&NewPinMode=userselectable

  cleartrust.agent.login_form_location_sid_newpin_sys_generated=
  /ctappagent/ct_securid.jsp?SecurIdMode=newpin&NewPinMode=cannotchoose

  cleartrust.agent.login_form_location_sid_newpin_user_specified=
  /ctappagent/ct_securid.jsp?SecurIdMode=newpin&NewPinMode=mustchoose

  cleartrust.agent.login_form_location_sid_newpin_display=
  /ctappagent/ct_securid.jsp?SecurIdMode=newpin&NewPinMode=display

  cleartrust.agent.login_error_sid_newpin=
  /ctappagent/ct_securid.jsp?SecurIdMode=newpin&NewPinMode=
  mustchoose&CTLoginErrorMsg=Invalid%20new%20pin

  cleartrust.agent.login_error_user_location_securid=
  /ctappagent/ct_logon.jsp?CTAuthMode=SECURID&CTLoginErrorMsg
  =Login%20Unsuccessful

  cleartrust.agent.login_error_pw_location_securid=
  /ctappagent/ct_logon.jsp?CTAuthMode=SECURID&CTLoginErrorMsg=
  Login%20Unsuccessful

  cleartrust.agent.login_error_location_securid=
  /ctappagent/ct_logon.jsp?CTAuthMode=SECURID&CTLoginErrorMsg=
  Login%20Unsuccessful

  SecurIdMode must be specified in the query string of the URL with one of the following values: nexttoken, newpin, or passcode. If SecurIdMode is not set, the default value is nexttoken.

- **Custom** - Set the following parameters and associated logon forms:

  cleartrust.agent.login_form_location_custom=
  /ctappagent/ct_logon.jsp?CTAuthMode=CUSTOM&language=<%language%>

  cleartrust.agent.login_error_user_location_custom=
  /ctappagent/ct_logon.jsp?CTAuthMode=CUSTOM&CTErrorLoginMsg=
  Login%20Failed&language=<%language%>

```
cleartrust.agent.login_error_pw_location_custom=
/ctappagent/ct_logon.jsp?CTAuthMode=CUSTOM&CTErrorLoginMsg=
Login%20Failed&language=<%language%>
```

CTLoginErrorMsg can be any error message that you want.

Note: Do not change the ct_logon.xxx page name. You can modify the contents of the file as long as you maintain ct_logon as the filename. Use %20 to define spaces in the previous arguments.

Configure SSO

The following settings are applicable only to the web resources residing in application servers that are protected by web filters. SSO can be configured between RSA Access Manager-protected application servers and also between RSA Access Manager protected application servers and web servers. Perform these steps on every application and web server that will be participating in SSO.

To configure SSO:

1. Install the RSA Access Manager Agent on each application server and web server you want to protect (if you have not already done so).
2. Set these parameters in each RSA Access Manager-protected application and web server Agent's configuration file (cleartrust.properties for application server Agents and **webagent.conf** for web server Agents):

- Enable single sign-on:
  cleartrust.agent.sso=True
- Set the domain name. The domain name that is set in the SSO cookie at runtime must be the same for all applications and web servers participating in SSO:
  cleartrust.agent.cookie_domain=.company.com
  Domain name values need a period before the domain name. If this parameter is left unset, the browser uses the name of the host server as the domain name in the SSO cookie. This allows SSO to work for the specific server only.

Review the other SSO and cookie Agent parameters to determine if you need to adjust the default settings for your application server environment. See .session_lifetime, .idle_timeout, .secure, .fudge_factor, .cookie_touch_window, and .auto_challenge parameters in the cleartrust.properties.

If BASIC or FORM auth is configured in the web.xml then ensure the following appropriate entries are un-commented in the jboss-web.xml

```
<!--
<class-name>
com.rsa.axm.jboss.sso.authenticators.RSASSOBasicAuthValve
</class-name>
<class-name>
com.rsa.axm.jboss.sso.authenticators.RSASSOFormAuthValve
</class-name>
```

5 RSA Access Manager Agent 5.0 SP1 Upgrade

This section describes how to upgrade the RSA Access Manager Agent 5.0 version to Agent 5.0 SP1 version on Jboss Application Server.

Note: RSA Access Manager Agent 5.0 SP1 supports upgrade from RSA Access Manager Agent 5.0 version only. If you are using version prior to Agent 5.0, you must upgrade to 5.0 before upgrading to 5.0 SP1. For more information on installing Agent 5.0, see *RSA Access Manager Application Server Agent 5.0 for RedHat Jboss Application Server Installation and Configuration Guide.*

Upgrading to RSA Access Manager Agent 5.0 SP1

The upgrade of Agent 5.0 SP1 includes the following high-level tasks:

1. Backup the existing version of Agent 5.0. To backup the existing files, see Backup Existing Version.
2. For upgrade instructions see, Upgrade to Agent 5.0 SP1.
3. To verify the installation, start the Jboss Application Server.

Backup Existing Version

Note: When backing up files, do not rename it to old files, but remove it from the directory.

1. Stop the Jboss server on which Agent 5.0 is installed.
2. Navigate to the existing **JBOSS_HOME/modules** directory and backup the **com/rsa** directory.
3. Navigate to the existing **JBOSS_HOME/standalone/deployments** directory and backup the following applications:
   - axm-jboss-agent-cacheflush-ejb-5.0.jar
   - axm-jboss-agent-filterui-5.0.war
   - (optional) You can also backup the files present in the **AGENT_HOME/webapps** directory.
4. Navigate to the existing **AGENT_HOME/tools** directory and backup the following utilities:
   - ctencrypt.bat/sh
   - cacheflush.bat/sh

- lockbox-tool.bat/sh

5. Navigate to the existing **JBOSS_HOME/bin/properties** directory and backup the following property files.
   - debugenabled.properties
   - debugdisabled.properties

Upgrade to Agent 5.0 SP1

1. Navigate to the **/lib/com** directory in Agent 5.0 SP1 package and copy the **com/rsa** directory to **JBOSS_HOME/modules/com** directory.
2. Navigate to **/webapps/** directory in Agent 5.0 SP1 package and copy the following applications to **JBOSS_HOME/standalone/deployments** directory:
   - axm-jboss-agent-cacheflush-ejb-5.1.jar
   - axm-jboss-agent-filterui-5.1.war
3. Navigate to the **/tools/** directory in Agent 5.0 SP1 package and copy the following tools to the existing **AGENT_HOME/tools** directory:
   - ctencrypt.bat/sh
   - cacheflush.bat/sh
   - lockbox-tool.bat/sh

   Note: You must update the AGENT_HOME variable in the above mentioned files before replacing the old utility files.
4. Navigate to **JBOSS_HOME/bin/properties** directory and update the cleartrust.properties file with the additional properties provided in **/conf/config_parameter.txt** file of the Agent 5.0 SP1 package.
5. Copy and replace the **following files** present in the /conf/ directory in the Agent 5.0 SP1 to **JBOSS_HOME/bin/properties** directory:
   - debugenabled.properties
   - debugdisabled.properties
6. Start JBoss Application Server.

6 Internationalization Support

Internationalization support is provided for Access Manager application agent login pages.

The internationalization feature is enabled for JSP login pages.

Setting up Internationalization

This section provides an overview to set up Internationalization for web agent logon pages.

No.	Task Description	Reference
1	**Language Selection.** You must select the language for the logon pages. The default language is English.	For more information see, Language Selection on page 63.
2	**Append Language Query String.** You must append the language query string in all the logon pages location. You can refer to the respective logon pages for configuration.	For more information see, Appending Language Query String on page 64.
3	**Property File.** You can also configure the default property file.	For more information see, Property File on page 64.
4	**Locale Specific Property File.** You must create the locale specific property file for the selected language. By default, a property file for English is provided.	For more information on JSP pages, Setting up Locale value and Locale specific File for JSP on page 65.
5	**Enable i18n Support.**	For more information see, Enable i18n support for JSP Login Pages on page 66.

Language Selection

You can define the supported languages by configuring the **cleartrust.agent.accepted_languages_list** parameter present in the cleartrust.properties file.
For example, **cleartrust.agent.accepted_languages_list =ja,fr-FR,en**.

For every HTTP request, the **Accept-Language** header is read to identify the client browser accepted Languages list.

The list is compared with the values specified for the parameter **cleartrust.agent.accepted_languages_list**:

- If there is a match, the matched language is selected as the language used for the request.
- If there is no match or the client browser does not send an 'Accept-Language' header, then language defined in the **cleartrust.agent.default_language**, is used.

By default **cleartrust.agent.default_language parameter** is set to **en** in the **cleartrust.properties** file. You can set the required language code supported as the default language and create the corresponding language property file.

Appending Language Query String

Append **language=<%language%>** as a query string in all the logon pages locations in the **cleartrust.properties** file. For example:

cleartrust.agent.login_form_location_basic=/cleartrust/ct_logon.jsp?language=<%language%>

Property File

A property file is a resource bundle that stores locale-specific messages, labels and image paths. The property file name is saved using a standard naming convention that includes the basename combined with a locale name. The property file contains key variables to store the values of the messages, labels and image paths.

The location of the property file for JSP is mentioned below:

- JSP-**\WEB-INF\classes\com\rsa\cleartrust\i18n** of the deployed cleartrust web application folder.

The default property file is **LogonPages.properties** for JSP, which corresponds to the default language, English. If the selected locale specific property file is not present, then the values are read from the properties file corresponding to the default language defined.

You can perform the following with the property file:

- **Add new values.** To add new messages, error messages, labels or image paths, you must define a key and assign a value for it in the property file. Following this, you must also include the key in the logon pages.
- **Modify existing values.** You can modify the existing values for the keys defined in the property file.

Note: Do not delete or replace the default property file.

Setting up Locale value and Locale specific File for JSP

You must make a copy of the existing **LogonPages.properties** file, rename the file by appending the language code to the filename, and provide the values for the keys in the file specific to the language selected. For example, to set up the Locale Specific Property File for French, make a copy of the **LogonPages.properties** property file and rename it to **LogonPages_fr.properties**. Then provide the new values for the keys specific to the language. You can create similar property files for each of the languages supported.

Note: You must not modify or delete existing keys in the property file. The values entered for the keys in the language specific file must be in Unicode-UTF 8 format. To create Locale Specific Property file, use the editor that supports UTF-8 encoding. This is applicable to all JSP and Servlets property files.

Note: The property file needs to be converted from native-encoded characters to Unicode-encoded characters. You can use tools such as native2ascii.exe to perform the conversion.

Internationalization Support for Images

Place locale specific images in the **images** folder of the deployed **axm-jboss-agent-filterui-5.0.war** application for internationalization support for images.

Enable i18n support for JSP Login Pages

You can enable internationalization support after configuring the language parameters.

To enable internationalization support for JSP logon pages, complete the following steps:

1. Configure the accepted languages parameter and default language parameter in the **cleartrust.properties** file, that is **cleartrust.agent.accepted_languages_list** and **cleartrust.agent.default_language**.
2. Append **language=<%language%>** as a query string in all the login form locations in the **cleartrust.properties** file.

   For example:

   cleartrust.agent.login_form_location_basic=/cleartrust/ct_logon.jsp?language=<%language%>

3. Create a new locale specific property file, namely LogonPages_<locale>.properties file in **\WEB-INF\classes\com\rsa\cleartrust\i18n** folder where axm-jboss-agent-filterui-5.0.war file is deployed.

   where,

   the value of locale can be **< 2 letter ISO language code in lower case > or <2 letter ISO language code in lower case_2 letter ISO country code in UPPER case>**, and must correspond to the values specified in the **cleartrust.agent.accepted_languages_list** parameter of the **cleartrust.properties** file.

   - Country code is optional and can be specified when specific dialects are required. For example, if you are configuring for French language, without country code specific then **cleartrust.agent.accepted_languages_list = fr** must be configured in the cleartrust.properties file and a new **LogonPages_fr.properties** file must be created as locale specific property file.
   - If you are configuring for French language in France, then **cleartrust.agent.accepted_languages_list = fr-FR** must be defined in the cleartrust.properties file and a new **LogonPages_fr_FR.properties** file must be created as locale specific property file.

   Note: The default LogonPages.properties file that supports English is present in **\WEB-INF\classes\com\rsa\cleartrust\i18n** folder of the web application.

4. In the locale specific property file, update the value for all messages, labels and error messages using unicodes in **UTF-8 encoding**.

   For example, in the default property file, a property named UserID, is defined as UserID=User ID. You must add the values in the property file for other languages in that particular language specific property file.

5. Place locale specific images in the **images** folder of the deployed **axm-jboss-agent-filterui-5.0.war** application for internationalization support for images.

Note: For example, to create the French language specific images, you must create a folder called **fr** in the **images** folder as mentioned above and and place the french language specific images in **fr** folder and update that folder path in the locale specific property file **LogonPages_fr.properties** file.

Verifying Internationalization support

Before accessing a web page, ensure that the language you have configured in cleartrust.properties is part of accepted language list in browser , that is if you have configured for French language (fr) then browser must have **fr** in its **ACCEPT LANGUAGE** list.

If the configured language and the browser language do not match, the default language defined in the **cleartrust.agent.default_language** parameter is used.

You can also provide language precedence, if you have configured multiple languages in cleartrust.properties.

To provide language precedence, complete the following tasks:

1. Launch the web browser.
2. Go to **Tools> Options> Choose Languages**. Include the language codes that you want to localize.
   For example, if you have configured for two languages French and Spanish, then include the language code **fr** prior to **es**.

7 General Configurations

RSA Access Manager Agent 5.0 SP1 supports logging of Agent logs to a Centralized log server. This feature facilitates distributed Agent deployments across the enterprise to redirect all the Agent logs to a central log server for better troubleshooting, maintenance and archiving.

RSA Access Manager Agent 5.0 SP1 also provides an option to limit the number of log files in the log directory.

This chapter details the Default and Centralized logging modes.

Logging Modes

RSA Access Manager Agent 5.0 SP1 supports the following logging modes:

- Default Logging
- Centralized logging

Default Logging

RSA Access Manager Agent 5.0 SP1 uses Log 4j framework for default logging. In this mode **cleartrust.agent.centralized.logging.enable** parameter is disabled.

Limit the number of files in log directory

Agent 5.0 SP1 uses **org.apache.log4j.RollingFileAppender** appender to limit the number of log files.

You can append maximum file size limit to file in the log directory by configuring the following parameter:

log4j.appender.cleartrustfile.MaxFileSize

You can also include the maximum number of backup files index value in the log directory by configuring the following parameter:

log4j.appender.cleartrustfile.MaxBackupIndex

Enabling Default Logging

This section explains how the default logging can be configured using the debugenabled.properties file.

To enable default logging, set the following parameter in cleratrust.properties file present in **JBOSS_HOME/bin/properties** directory to the following:

```
cleartrust.agent.debug_configuration=debugenabled.properties
```

Disabling Default Logging

This section explains how the default logging can be removed using the debugdisabled.properties file.

To disable default logging, set the following parameter in cleratrust.properties file present in **JBOSS_HOME/bin/properties** directory to the following:

```
cleartrust.agent.debug_configuration=debugdisabled.properties
```

Additional Information

The following section provides an overview of debugenabled.properties.

Following is the content with some values assigned as an example:

```
log4j.rootCategory=DEBUG, cleartrustfile, stdout
log4j.appender.cleartrustfile=org.apache.log4j.RollingFileAppen
der
log4j.appender.cleartrustfile.File=cleartrust.log
log4j.appender.cleartrustfile.Append=false
log4j.appender.cleartrustfile.MaxFileSize=5 MB
log4j.appender.cleartrustfile.MaxBackupIndex=2
log4j.appender.cleartrustfile.layout=com.rsa.cleartrust.common.
log.log4j.AgentLogPatternLayout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=com.rsa.cleartrust.common.log.log4
j.AgentLogPatternLayout
log4j.appender.stdout.Target=System.out
```

In the above example, you can apprehend the following:

- The level of the root logger is defined as DEBUG and attaches appender named cleartrustfile and stdout to it.
- The appender cleartrustfile is defined as org.apache.log4j.RollingFileAppender and writes to a file named " cleartrust.log “.
- The maximum permissible size of each log file is 5MB.
- The maximum number that the output is allowed to reach before being rolled over to backup files.

When debug option is disabled, no information is logged, but you can configure to log at ERROR log level by uncommenting the following lines:

#log4j.logger.com.rsa=OFF

#log4j.logger.sirrus=OFF

Centralized Logging

Agent uses Log 4j framework to support centralized logging for logging messages. Log 4j framework consists of Appenders and loggers, that helps log messages on the basis of message type and format.

This mode includes configuring Agent for centralized logging with Access Manager Servers that are running on Java Virtual machine, with the configuration cleartrust.agent.centralized.logging.enable=True

Failover Logging

Agent also provides failover support, which is if the primary Access Manager Log server fails then the log files are logged to secondary log server, when single or multiple instances of log server (s) are configured.

When all or single host server(s) fail, then the failover support provided is similar to default logging.

Following are the two failover logging types supported:

- Failover logging using Single log server host- This includes failover logging support only when single instance Access Manager Log server host is configured. This failover mechanism is similar to default logging where in the default file **cleartrust.log** is used to log messages and the configuration is governed by the Agent FileHandler configuration in standalone.xml. For more information see, Centralized Logging Configuration Parameters
- Failover logging using multiple log server hosts-This includes failover logging support when multiple instances Access Manager Log server hosts are configured. When all the log server hosts fail, then the default file 'cleartrust.log' is used to log messages and the configuration is governed by the Agent FileHandler configuration in standalone.xml. For more information see, Centralized Logging Configuration Parameters

Centralized Logging Configuration Parameters

Following parameters must be set, to utilize Centralized logging option:

Parameter	Description
cleartrust.agent.centralized.logging.enable	Specifies whether Centralized Logging is enabled or disabled in Application agents.

cleartrust.agent.logserver.clientId	Specifies the Agent instance that is logging to Centralized log server. The value of this parameter is added to the beginning of each log message. When several different agents are sending log messages to a Access Manager log server, this parameter can be used to identify the origin of each message. A value is appended to this parameter, which is displayed in the beginning of each log message.
cleartrust.agent.logserver_list	Specifies the Access Manager Log Server connection details. You can use this parameter to configure the Access Manager Log Server host and port details. Default value:None Allowed Value:host1:port1, host2:port2 **Note:** To use this parameter, you must set the following parameters: cleartrust.agent.centralized.logging.enable and cleartrust.agent.logserver.clientId
cleartrust.agent.logserver.log_delimiter	Specifies the separated entries in log messages in Access Manager Log server. Allowed Values: Any string or special characters.
cleartrust.aserver.log.server_reconnect_delay	When the Access Manager Log server connection is lost, this parameter specifies the number of milli seconds to wait before reestablishing the connection. Default Value:30000 ms (30 seconds) Allowed Value: A positive integer. **Note:** This parameter is only used for remote logging.

Agent uses AxM Socket Appender to establish connection with Access Manager log server using the following modes:

- Anon
- Auth

Important: RSA recommends the communication between Agent and Access Manager Log server using Auth mode.

Configuring Centralized Logging for Jboss Application Server

JBoss Server logging configuration is represented by the logging subsystem. It consists of the following sections:

- handler configurations
- logger Category definitions
- root logger declarations

Each logger references a handler or set of handlers and these handlers declare the log format and output. You must include these handlers and loggers to obtain log format and log message output.

To configure Agent 5.0 SP1 for Jboss application server, complete the following steps:

1. Under **<JBOSS_HOME>/modules**, go to org\jboss\log4j\logmanager\main location, and add the following line in the module.xml file:

```
<module name="com.rsa.cleartrust"/>

Module.xml file:
<dependencies>
        <module name="com.rsa.cleartrust"/>
        <module name="javax.api"/>
        <module name="org.dom4j" optional="true"/>
        <module name="org.jboss.logmanager"/>
        <module name="org.jboss.modules"/>
    </dependencies>
```

2. Go to <JBOSS_HOME>/standalone/configuration location and add the following content in the **standalone.xml** file under the **urn:jboss:domain:logging:1:1** subsystem tag:

```
<custom-handler name="AxMLogger"
class="com.rsa.axm.jboss.logging.AgentLoggingHandler"
module="com.rsa.axm.jboss">
<level name="DEBUG"/>
<formatter>
<pattern-formatter %s%E%n"/>
</formatter>
</custom-handler>
<custom-handler name="AxMFLogger"
class="com.rsa.axm.jboss.logging.AgentFileHandler"
module="com.rsa.axm.jboss">
<level name="DEBUG"/>
<formatter>
<pattern-formatter %s%E%n"/>
</formatter>
<properties>
<property name="logFilePath" value="/opt/agent123.log"/>
<property name="logLimit" value="1M"/>
<property name="numFiles" value="7"/>
</properties>
</custom-handler>
```

3. Define **Agent Logger** for 5.0 SP1 and include previously defined handlers:

```
<logger category="com.rsa">
<level name="DEBUG"/>
<handlers>
handler name="AxMLogger"/>
</handlers>
</logger>
```

4. Add **Agent Logger** to the <root-handler> module:

```
<root-logger>
<handlers>
<handlers>
handler name="AxMLogger"/>
</handlers>
</root-logger>
```

Configuring Centralized Logging for RSA Access Manager Agent 5.0 SP1

Pre-requisites

- RSA Access Manager Servers along with Access Manager Log Server must be installed and running.
- Required configuration parameters are set in cleartrust.properties file.

To configure Centralized Logging in the application agent, complete the following steps:

1. Set the connection mode to Auth or Anon, for connecting to Access Manager Log server.
2. Specify the following parameter in cleartrust.properties file:
   - cleartrust.agent.centralized.logging.enable
   - cleartrust.agent.logserver.clientId
   - cleartrust.agent.logserver_list
   - cleartrust.agent.logserver.log_delimiter
   - cleartrust.aserver.log.server_reconnect_delay

Note: If clientID is not configured, IP Address of the Agent host is appended. If connection is not established to the configured Access Manager logserver host, log events are written by default to the Agent Host log file by name cleartrust.log file.

8 RSA Access Manager Agent Tools

Cacheflush

This Agent has seven types of caches. These caches will cache the calls made to the RSA Access Manager Authorization Server so that subsequent similar requests will be served from the cache rather than the Authorization Server. While this results in faster responses and increases the performance of the Agent, any changes made at the Authorization Server will not be reflected immediately in the Agent until the cache entries expire.

To solve this problem, the Agent includes a tool that can flush the entries in its caches. This cacheflush tool is implemented as an Enterprise Java Bean and is installed in JBoss during the installation of the Agent. This bean can be invoked using the cacheflush shell script (UNIX machines) or batch file (Windows machines) present in the tools subdirectory of the Agent installation folder.

The cacheflush tool can be invoked with the following format:

cacheflush [-t cache type] [-s]

-t: The type of cache to be flushed.

Cache type is a numeric value with the following mappings:

1	Authorization Allow Cache
2	Authorization Deny Cache
4	Groups Cache
8	User Properties Cache
16	Protected Resource Cache
32	Unprotected Resource Cache
64	Tokens Cache

Multiple cache types can be passed with the -t option by adding the cache type numeric values. For example, to flush Authorization Allow and Groups Cache, run the cacheflush tool with "-t 5", where 5 represents the sum of the Authorization Allow Cache (1), and Groups Cache (4) cache type values.

-s: Displays the current cache statistics in the Agent. When the -s option is specified, the cache is not flushed. The statistics displayed include the current size of the cache, the number of requests made, and the cache hit count.

Lockbox File

This section describes about the steps that need to be followed to create the Lockbox file which is used by the Ctencrypt tool.

The Lockbox file stores the encryption key used by Ctencrypt tool. The encryption key is derived from the encryption password supplied in the arguments while creating or adding new item in the Lockbox file.

The Lockbox file securely stores secrets such as encryption keys. Each value stored in the Lockbox file is referenced by using a key item.

The Lockbox file stores data in key-value pairs. There by, one Lockbox file can be used for different components for storing their encryption keys. To Store multiple encryption keys in a Lockbox file choose different key items.

Note: If you enable the LockBox feature and run an existing Lockbox file, the utility adds the key item and encryption key to the Lockbox file.

To create and generate the Lockbox file, you must:

1. Open command prompt and navigate to **JBOSS_HOME/tools.**
2. Run the lockbox-tool. Type:

   lockbox-tool -passphrase <*phrase*> [-lockbox <*filepath*>]
   -[-{create|delete|exist|changepwd}] <item-name> <value>
   [-{addhost|deletehost|listhosts}] <hosts>

   The switch names can be abbreviated and are not case sensitive.

   -passphrase <phrase>: Lockbox Passphrase used to generate the encryption key. The passphrase must meet the following requirements.

   – 8 or more characters in length
   – Contain at least one numeric character
   – Contain at least one uppercase and lowercase character

– Contain at least one non-alphanumeric or special character such as # or !

-lockbox <filepath>: Path of the Lockbox file. If the file is not found, a new file will be created with the name "*lockbox.clb*". If found, the file gets updated.

-create | -delete | -exist: The key (or item) must be present to perform lockbox keyitem management. Create, delete or check for the existence of an item in the lockbox. It can be an arbitrary string but should not start with __.

-changepwd: Changes the lockbox passphrase.

<item-name>: The key (or item) required for all item actions. This item-name is used to associate the password in lockbox. Update the same key name in all the relevant configuration files.

<value>: The key (or item) required to create item action and lockbox passphrase change action. In case of create item action this value is the Password which will be used for encrypting/decrypting the configuration.

-addhost|-deletehost|-listhosts: The key (host actions) must be present to perform lockbox host management. Adds, deletes and prints the hostnames in the lockbox.

<hosts>: List of host names separated by comma in which the lockbox file is used. Required for addhost and delete host actions. Currently lockbox file is required for Ctencrypt tool.

The generated Lockbox file can be further used by Ctencrypt Tool, RSA Access Manager Agents or Lockbox file tool.

Install the Lockbox File Dependencies

Lockbox file relies on the applications to function properly. These applications need to be installed in those machines where Lockbox file needs to be used. The Ctencrypt Tool use the Lockbox file tool.

The Lockbox file dependency has to be installed in the machines where EncryptUtil Tool is deployed or used.

To install Lockbox file on a Windows platform:

1. Navigate to the **/cst** directory in the installation kit.
2. Select the Windows 32 bit or Windows 64 bit CST kit.
3. Copy the CST kit to a temporary directory.
4. From the temporary location, run **vcredist.exe** to install Visual C++ 2010 redistributable package.
5. Update the PATH environment to point to the temporary directory.

To install Lockbox file on a Linux or UNIX platform:

1. Navigate to the **/cst** directory in the installation kit.
2. Copy the CST kit to a temporary directory.
3. Set LD_LIBRARY_PATH environment variable to point to the temporary directory.

Ctencrypt

The Agent requires that certain confidential property values to be encrypted in the cleartrust.properties file. The Agent installer automatically encrypts all the confidential property values during installation. Using the ctencrypt utility, these confidential property values can be changed and re-encrypted at a later time.

You can generate encrypted values for the following parameters:

- cleartrust.agent.adapi.user_id
- cleartrust.agent.adapi.user_password

For Example:

ctencrypt mode lockbox_file_path lockboxkeyitem param1=value1 param2=value2 param3=value3 param4=value4 param5=value5

where,

- ***mode*** specifies whether to encrypt the value using the FIPS algorithm. Allowed values, fips or nonfips.
- ***lockbox_file_path*** specifies the location where the lockbox file is stored.
- ***lockboxkeyitem*** specifies the key item name in Lockbox file, which will be used to get the encryption key.
- ***param1*** and ***param2*** are the two parameters to encrypt.
- ***value1*** and ***value2*** are the values assigned to the two parameters.

9 Caching and Performance Tuning

Exclusion Lists

Exclusion lists improve the performance of the Agent by excluding access checks for the specified resources.
By default, .gif, .jpg, .png, .ico, and .css file formats are excluded from access control checks to enhance Agent performance. See .url_exclusion_list and .extension_exclusion_list in cleartrust.properties.

URL Exclusion List

This specifies a list of URLs to be excluded from access control checks. There is no need to specify the RSA Access Manager logon forms here, as the Agent automatically includes them in the exclusion list. The configured parameter is cleartrust.agent.uri_exclusion_list.

For example:

cleartrust.agent.url_exclusion_list=/pub/,/private/pub.html

Extension Exclusion List

This specifies a list of file extensions. The URLs that end with these file extensions are excluded from access control checks. The configured parameter is cleartrust.agent.extension_exclusion_list.

For example: cleartrust.agent.extension_exclusion_list=gif,jpg

Caching Parameters

When a request is made to the Authorization Server, the result of the request is cached in the Agent so that the subsequent request can be served from the cache rather than the Authorization Server.

The Authorization Server and Agent caches are independent, and are not related in any way. RSA recommends enabling caching in both the Authorization Server and the Agent. There are seven different types of cache maintained in the Agent. The size and time to live property of each of these caches can be configured independently. You can disable caches independently or disable all the caches.

All the caches in the Agent can be disabled by setting the cleartrust.agent.enable_cache property to False.

There are two types of cache parameters that can be configured for each cache.

- **Cache size.** This specifies the maximum number of entries that can be cached in a single cache. When the cache table reaches the size specified by this parameter, the oldest entries are removed from the cache first. A value of zero (0) means the cache table can grow infinitely. There is no limit on the table size, but entries are deleted as they expire. If this property is not set for a cache then that particular cache is disabled.
- **Time to Live (TTL).** The value specified for this parameter is used to determine how frequently the Agent deletes the cache entries. A value of '0 Secs' means that the cache entries are never deleted based on TTL. If this property is not set for a cache then that particular cache is disabled.

The cache parameters set cache size and the amount of time that a entry is stored in the cache. Setting these parameters helps increase the performance of the Agent. The RSA Access Manager Server caches update whenever there is a change through the Administrative Console or through the Administrative API. The Agent cache is updated based on the "Time To Live" (TTL) settings.

The different types of Cache maintained in the agent are:

- Protected Resource Cache
- Unprotected Resource Cache
- Authorization Allow
- Authorization Deny
- Group
- Token
- User Properties

Protected Resource Cache

Properties:

cleartrust.agent.protected_resource_cache_size
cleartrust.agent.protected_resource_cache_ttl

Unprotected Resource Cache

Properties:

cleartrust.agent.unprotected_resource_cache_size
cleartrust.agent.unprotected_resource_cache_ttl

Authorization Allow Cache

Properties:

cleartrust.agent.authz_allow_cache_size
cleartrust.agent.authz_allow_cache_ttl

Authorization Deny Cache

Properties:

cleartrust.agent.authz_deny_cache_ttl
cleartrust.agent.authz_deny_cache_size

Group Cache

Properties:

cleartrust.agent.groups_cache_ttl
cleartrust.agent.groups_cache_size

Token Cache

Properties:

cleartrust.agent.token_cache_ttl
cleartrust.agent.token_cache_size

User Properties Cache

Properties:

cleartrust.agent.userprops_cache_ttl
cleartrust.agent.userprops_cache_size