Red Hat JBoss Enterprise Application Platform 7.3

Login Module Reference

Lists and descriptions of the login modules available for Red Hat JBoss Enterprise Application Platform.

Last Updated: 2021-08-27

Red Hat JBoss Enterprise Application Platform 7.3 Login Module Reference

Lists and descriptions of the login modules available for Red Hat JBoss Enterprise Application Platform.

Legal Notice

Abstract

The intent of this document is to provide a reference for the login modules available in Red Hat JBoss Enterprise Application Platform. For more background information on how login modules work in JBoss EAP, see the Security Architecture for JBoss EAP document.

Table of Contents

PREFACE

IMPORTANT

The Login Modules described in this guide are deprecated due to Elytron being introduced. For instructions on using the **elytron** subsystem see the Elytron Subsystem section in the *How to Configure Server Security* Guide.

CHAPTER 1. LOGIN MODULE OVERVIEW

The basics of login modules and their use within security domains are covered in the Security Domains section in the JBoss EAP *Security Architecture* guide.

1.1. ABOUT THE ORGANIZATION OF THIS DOCUMENT

The login modules covered in this document are organized into the following functional areas:

Login Module Functional Organization

- Login Modules Without External Identity Store
  - Identity Login Module - Used when a fixed or hard-coded user name is needed.
  - UsersRoles Login Module - Loads user names and roles from a local Java properties files.
  - PropertiesUsers Login Module - Loads only user names from a local Java properties files.
  - SimpleUsers Login Module - Defines user names and passwords directly in the login module configuration.
  - SecureIdentity Login Module - Legacy, allows for a static principal and encrypted password to be defined directly in the module configuration.
  - ConfiguredIdentity Login Module - Associates a static principal to any authenticated user.
  - Simple Login Module - A module for quick security setup for testing.
  - Disabled Login Module - A module that always fails authentication.
  - Anon Login Module - A module to specify the identity for an unauthenticated user.
  - RunAs Login Module - Helper module for adding an additional static role during the authentication phase.
  - RoleMapping Login Module - Helper module for adding to or replacing the roles of an authenticated user with one or more roles.
  - RealmDirect Login Module - Delegates authentication to a security realm.
  - RealmUsersRoles Login Module - Legacy module replaced by RealmDirect.
- Login Modules With External Identity Store
  - Database Login Module - Uses a database to store users and role mappings.
  - DatabaseUsers Login Module - Alias to Database for compatibility.
  - Ldap Login Module - Uses an LDAP server to store users and role mappings.
  - LdapExtended Login Module
  - AdvancedLdap Login Module - Provides additional functionality when authenticating using an LDAP server.

    - AdvancedAdLdap Login Module - Provides additional functionality used in Microsoft Active Directory.
    - LdapUsers Login Module - Legacy module replaced by LdapExtended and AdvancedLdap.
    - Kerberos Login Module - Used with Kerberos authentication.
    - SPNEGO Login Module - Used with Kerberos authentication.
- Certificate-Based Login Modules
    - Certificate Login Module - Authenticates users via X509 certificates.
    - CertificateRoles Login Module - Extends Certificate module with role mapping.
    - DatabaseCertificate Login Module - Extends Certificate module with role mapping stored in a database.
- Login Modules for EJBs and Remoting
    - Remoting Login Module - Used in securing remote EJB invocations.
    - Client Login Module - Used in local, in-JVM, EJB calls for establishing client identity.
- Custom Login Modules

This guide also provides reference information for related topics such as authorization modules, password stacking and password hashing.

1.2. EXTENSION HIERARCHY

The vast majority of the login modules covered in this document actually extend the configuration options and functionality of other login modules. The structure the login modules use to extend functionality forms a hierarchy:

Login Module Extension Hierarchy

- AbstractServer Login Module
    - AbstractPasswordCredential Login Module
        - SecureIdentity Login Module
        - ConfiguredIdentity Login Module
    - Certificate Login Module
        - CertificateRoles Login Module
        - DatabaseCertificate Login Module
    - Common Login Module
        - AdvancedLdap Login Module
            - AdvancedAdLdap Login Module
        - SPNEGO Login Module

- Identity Login Module
- RoleMapping Login Module
- Remoting Login Module
- UsernamePassword Login Module
  - Database Login Module
  - LdapExtended Login Module
  - Ldap Login Module
  - LdapUsers Login Module
  - Simple Login Module
  - Anon Login Module
  - RealmDirect Login Module
  - UsersRoles Login Module
    - RealmUsersRoles Login Module
    - PropertiesUsers Login Module
      - SimpleUsers Login Module
- Client Login Module
- DatabaseUsers Login Module
- Disabled Login Module
- Kerberos Login Module
- RunAs Login Module

Most of the login modules in the hierarchy are concrete Java classes that are instantiated and used by JBoss EAP, but there are a few abstract classes that cannot be instantiated and used directly. The purpose of these abstract classes are to provide common functionality and to serve purely as a base class for other login modules to extend.

IMPORTANT

By default, login modules inherit all behavior and options from login modules they extend, but they may also override that behavior from their parent login module. In some cases, this may lead to certain options that are inherited by a login module from their parent but go unused.

CHAPTER 2. ABSTRACT LOGIN MODULES

The abstract login modules are abstract Java classes that are extended by the other login modules in order to provide common functionality and configuration options. The abstract login modules may never be used directly, but the configuration options are available to any login modules that extend them.

2.1. ABSTRACTSERVER LOGIN MODULE

Short name: AbstractServerLoginModule

Full name: org.jboss.security.auth.spi.AbstractServerLoginModule

The AbstractServer Login Module serves as a base class for many login modules as well as several abstract login modules. It implements the common functionality required for a JAAS server side login module and implements the PicketBox standard Subject usage pattern of storing identities and roles.

Option	Type	Default	Description
principalClass	A fully qualified classname	org.jboss.security.SimplePrincipal	A Principal implementation class which contains a constructor that takes String argument for the principal name.
module	String	none	A reference to a jboss-module that can be used to load a custom callback/validator.
unauthenticatedIdentity	String	none	This defines the principal name that should be assigned to requests that contain no authentication information. This can allow unprotected servlets to invoke methods on EJBs that do not require a specific role. Such a principal has no associated roles and can only access unsecured EJBs or EJB methods that are associated with the unchecked permission constraint. See the Unauthenticated Identity section for more details.

Option	Type	Default	Description
password-stacking	useFirstPass or false	false	See the Password Stacking section for more details.

2.1.1. Unauthenticated Identity

Not all requests are received in an authenticated format. The **unauthenticatedIdentity** login module configuration assigns a specific identity, **guest** for example, to requests that are made with no associated authentication information. This can be used to allow unprotected servlets to invoke methods on EJBs that do not require a specific role. Such a principal has no associated roles and so can only access either unsecured EJBs or EJB methods that are associated with the unchecked permission constraint. For example, this configuration option can be used in the UsersRoles and Remoting Login Modules

2.1.2. Password Stacking

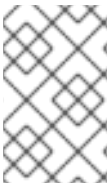
Multiple login modules can be chained together in a stack, with each login module providing both the credentials verification and role assignment during authentication. This works for many use cases, but sometimes credentials verification and role assignment are split across multiple user management stores.

Consider the case where users are managed in a central LDAP server but application-specific roles are stored in the application's relational database. The password-stacking module option captures this relationship.

To use password stacking, each login module should set the **password-stacking** attribute to **useFirstPass**, which is located in the **<module-option>** section. If a previous module configured for password stacking has authenticated the user, all the other stacking modules will consider the user authenticated and only attempt to provide a set of roles for the authorization step.

When password-stacking option is set to **useFirstPass**, this module first looks for a shared user name and password under the property names **javax.security.auth.login.name** and **javax.security.auth.login.password** respectively in the login module shared state map.

If found, these properties are used as the principal name and password. If not found, the principal name and password are set by this login module and stored under the property names **javax.security.auth.login.name** and **javax.security.auth.login.password** respectively.



NOTE

When using password stacking, set all modules to be required. This ensures that all modules are considered, and have the chance to contribute roles to the authorization process.

2.2. USERNAMEPASSWORD LOGIN MODULE

Short name: UsernamePasswordLoginModule

Full name: org.jboss.security.auth.spi.UsernamePasswordLoginModule

Parent: AbstractServer Login Module

The UsernamePassword Login Module is an abstract login module that imposes an **identity == String username**, **credentials == String password** view on the login process. It inherits all the fields from Abstract Server login module in addition to the below fields.

Option	Type	Default	Description
ignorePasswordCase	boolean	false	A flag indicating if the password comparison should ignore case.
digestCallback	A fully qualified classname	none	The class name of the **org.jboss.crypto.digest.DigestCallback** implementation that includes pre/post digest content like salts for hashing the input password. Only used if **hashAlgorithm** has been specified and **hashUserPassword** is set to **true**.
storeDigestCallback	A fully qualified classname	none	The class name of the **org.jboss.crypto.digest.DigestCallback** implementation that includes pre/post digest content like salts for hashing the store/expected password. Only used if **hashStorePassword** is true and **hashAlgorithm** has been specified.
throwValidateError	boolean	false	A flag that indicates whether validation errors should be exposed to clients or not.
inputValidator	A fully qualified classname	none	The instance of the **org.jboss.security.auth.spi.InputValidator** implementation used to validate the user name and password supplied by the client.

NOTE

The **UsernamePassword** Login Module options, regarding password hashing, are described in the next section.

2.2.1. Password Hashing

Most login modules must compare a client-supplied password to a password stored in a user management system. These modules generally work with plain text passwords, but can be configured to support hashed passwords to prevent plain text passwords from being stored on the server side. JBoss EAP supports the ability to configure the hashing algorithm, encoding, and character set as well as when the user password and store password are hashed.

The following are password hashing options that can be configured as part of a login module that has **UsernamePassword** Login Module as a parent:

Option	Type	Default	Description
hashAlgorithm	String representing a password hashing algorithm.	none	Name of the **java.security.MessageDigest** algorithm to be used to hash the password. There is no default so this option must be specified to enable hashing. Typical values are **SHA-256**, **SHA-1** and **MD5**. When **hashAlgorithm** is specified and **hashUserPassword** is set to **true**, the clear text password obtained from the **CallbackHandler** is hashed before it is passed to **UsernamePasswordLoginModule.validatePassword** as the **inputPassword** argument.
hashEncoding	String	base64	The string format for the hashed password, if **hashAlgorithm** is also set. May specify one of three encoding types: **base64**, **hex** or **rfc2617**.
hashCharset	String	The default encoding set in the container's runtime environment	The name of the charset/encoding to use when converting the password string to a byte array.
hashUserPassword	boolean	true	A flag indicating if the user entered password should be hashed. The hashed user password is compared against the value in the login module, which is expected to be a hash of the password.

Option	Type	Default	Description
hashStorePassword	boolean	false	A flag indicating if the store password returned should be hashed. This is used for digest authentication, where the user submits a hash of the user password along with a request-specific tokens from the server to be compared. The hash algorithm, for digest, this would be **rfc2617**, is utilized to compute a server-side hash, which should match the hashed value sent from the client.
passwordIsA1Hash	boolean		A flag used by the **org.jboss.security.auth.callback.RFC2617Digest** when it is configured as the **digestCallback** or **storeDigestCallback**. If true, incoming password will not be hashed since it is already hashed.

2.3. ABSTRACTPASSWORDCREDENTIAL LOGIN MODULE

Short name: AbstractPasswordCredentialLoginModule

Full name: org.picketbox.datasource.security.AbstractPasswordCredentialLoginModule

Parent: AbstractServer Login Module

AbstractPasswordCredential Login Module is a base login module that handles PasswordCredentials.

2.4. COMMON LOGIN MODULE

Short name: CommonLoginModule

Full name: org.jboss.security.negotiation.common.CommonLoginModule

Parent: AbstractServer Login Module

Common Login Module is an abstract login module that serves as a base login module for some login modules within JBoss Negotiation.

CHAPTER 3. LOGIN MODULES WITHOUT EXTERNAL IDENTITY STORE

3.1. IDENTITY LOGIN MODULE

Short name: Identity

Full name: org.jboss.security.auth.spi.IdentityLoginModule

Parent: AbstractServer Login Module

Identity login module is a simple login module that associates a hard-coded user name to any subject authenticated against the module. It creates a **SimplePrincipal** instance using the name specified by the principal option. This login module is useful when a fixed identity is required to be provided to a service. This can also be used in development environments for testing the security associated with a given principal and associated roles.

Table 3.1. Identity Login Module Options

Option	Type	Default	Description
principal	String	guest	The name to use for the principal.
roles	comma-separated list of Strings	none	A comma-delimited list of roles which will be assigned to the subject.

3.2. USERSROLES LOGIN MODULE

Short name: UsersRoles

Full name: org.jboss.security.auth.spi.UsersRolesLoginModule

Parent: UsernamePassword Login Module

UsersRoles login module is a simple login module that supports multiple users and user roles loaded from Java properties files. The primary purpose of this login module is to easily test the security settings of multiple users and roles using properties files deployed with the application.

Table 3.2. UsersRoles Login Module Options

Option	Type	Default	Description
usersProperties	Path to a file or resource.	users.properties	The file or resource which contains the user-to-password mappings. The format of the file is **username=password**

Option	Type	Default	Description
rolesProperties	Path to a file or resource.	roles.properties	The file or resource which contains the user-to-role mappings. The format of the file is **username=role1,role2,role3**
defaultUsersProperties	String	**defaultUsers.properties**	The name of the properties resource containing the username-to-password mappings that will be used as the default properties passed to the **usersProperties** properties.
defaultRolesProperties	String	**defaultRoles.properties**	The name of the properties resource containing the username-to-roles mappings that will be used as the default properties passed to the **usersProperties** properties.
roleGroupSeperator	String	.	The character used to separate the role group name from the user name, for example **jduke.CallerPrincipal=...**.

3.3. PROPERTIESUSERS LOGIN MODULE

Short name: PropertiesUsers

Full name: org.jboss.security.auth.spi.PropertiesUsersLoginModule

Parent: UsersRoles Login Module

The **PropertiesUsers** login module that uses a properties file to store user names and passwords for authentication. No authorization, role mapping, is provided. This module is only appropriate for testing.

3.4. SIMPLEUSERS LOGIN MODULE

Short name: SimpleUsers

Full name: org.jboss.security.auth.spi.SimpleUsersLoginModule

Parent: PropertiesUsers Login Module

The **SimpleUsers** login module that stores the user name and clear-text password using **module-option**. The **name** and **value** attributes of the **module-option** specifies a user name and password. It is included for testing only, and is not appropriate for a production environment.

3.5. SECUREIDENTITY LOGIN MODULE

Short name: SecureIdentity

Full name: org.picketbox.datasource.security.SecureIdentityLoginModule

Parent: AbstractPasswordCredential Login Module

The **SecurityIdentity** login module is a module that is provided for legacy purposes. It allows users to encrypt a password and then use the encrypted password with a static principal. If an application uses **SecureIdentity**, consider using a password vault mechanism instead.

Table 3.3. SecureIdentity Login Module Options

Option	Type	Default	Description
username	String	none	The user name for authentication.
password	encrypted String	""	The password to use for authentication. To encrypt the password, use the module directly at the command line, for example **java org.picketbox.datasource.security.SecureIdentityLoginModule password_to_encrypt**, and paste the result of this command into the module option's value field. The default value is an empty String.
managedConnectionFactoryName	JCA resource	none	The name of the JCA connection factory for your datasource.

3.6. CONFIGUREDIDENTITY LOGIN MODULE

Short name: ConfiguredIdentity

Full name: org.picketbox.datasource.security.ConfiguredIdentityLoginModule

Parent: AbstractPasswordCredential Login Module

The **ConfiguredIdentity** login module associates the principal specified in the module options with any subject authenticated against the module. The type of Principal class used is **org.jboss.security.SimplePrincipal**.

Table 3.4. ConfiguredIdentity Login Module Options

Option	Type	Default	Description
username	String	none	The user name for authentication.
password	encrypted String	""	The password to use for authentication, which can be encrypted via the vault mechanism. The default value is an empty String.
principal	Name of a principal	none	The principal which will be associated with any subject authenticated against the module.

3.7. SIMPLE LOGIN MODULE

Short name: Simple

Full name: org.jboss.security.auth.spi.SimpleServerLoginModule

Parent: UsernamePassword Login Module

The Simple login module is a module for quick setup of security for testing purposes. It implements the following simple algorithm:

- If the password is null, authenticate the user and assign an identity of **guest** and a role of **guest**.
- Otherwise, if the password is equal to the user, assign an identity equal to the **username** and both **user** and **guest** roles.
- Otherwise, authentication fails.

The Simple login module has no options.

3.8. DISABLED LOGIN MODULE

Short name: Disabled

Full name: org.jboss.security.auth.spi.DisabledLoginModule

A login module that always fails authentication. It is to be used for a security domain that needs to be disabled, for instance when we do not want JAAS to fall back to using the **other** security domain.

Table 3.5. Disabled Login Module Options

Option	Type	Default	Description
jboss.security.security_domain	String		Name of security domain to display in error message.

3.9. ANON LOGIN MODULE

Short name: Anon

Full name: org.jboss.security.auth.spi.AnonLoginModule

Parent: UsernamePassword Login Module

A simple login module that allows for the specification of the identity of unauthenticated users via the **unauthenticatedIdentity** property. This login module has no additional options beyond its inherited options from UsernamePassword Login Module.

3.10. RUNAS LOGIN MODULE

Short name: RunAs

Full name: org.jboss.security.auth.spi.RunAsLoginModule

The **RunAs** login module is a helper module that pushes a **run as** role onto the stack for the duration of the login phase of authentication, then pops the **run as** role from the stack in either the commit or abort phase. The purpose of this login module is to provide a role for other login modules that must access secured resources in order to perform their authentication, for example, a login module that accesses a secured EJB. The **RunAs** login module must be configured ahead of the login modules that require a **run as** role established.

Table 3.6. RunAs Login Module Options

Option	Type	Default	Description
roleName	role name	nobody	The name of the role to use as the **run as** role during the login phase.
principalName	principal name	nobody	Name of the principal to use as the **run as** principal during login phase. If not specified a default of nobody is used.

Option	Type	Default	Description
principalClass	A fully qualified classname.	org.jboss.security.SimplePrincipal	A Principal implementation class which contains a constructor that takes String arguments for the principal name.

3.11. ROLEMAPPING LOGIN MODULE

Short name: RoleMapping

Full name: org.jboss.security.auth.spi.RoleMappingLoginModule

Parent: AbstractServer Login Module

The **RoleMapping** login module is a login module that supports mapping roles, that are the end result of the authentication process, to one or more declarative roles. For example, if the authentication process has determined that the user **John** has the roles **ldapAdmin** and **testAdmin**, and the declarative role defined in the **web.xml** or **ejb-jar.xml** file for access is **admin**, then this login module maps the admin roles to **John**. The **RoleMapping** login module must be defined as an optional module to a login module configuration as it alters mapping of the previously mapped roles.

Table 3.7. RoleMapping Login Module Options

Option	Type	Default	Description
rolesProperties	The fully qualified file path and name of a properties file or resource	none	The fully qualified file path and name of a properties file or resource which maps roles to replacement roles. The format is **original_role=role1,role2,role3**.
replaceRole	true or false	false	Whether to add to the current roles, or replace the current roles with the mapped ones. Replaces if set to true.

3.12. REALMDIRECT LOGIN MODULE

Short name: RealmDirect

Full name: org.jboss.as.security.RealmDirectLoginModule

Parent: UsernamePassword Login Module

The **RealmDirect** login module allows for the use of an existing security realm to be used in making authentication and authorization decisions. When configured, this module will look up identity information using the referenced realm for making authentication decisions and delegate to that security realm for authorization decisions. For example, the pre-configured **other** security domain that ships with JBoss EAP has a **RealmDirect** login module. If no realm is referenced in this module, the **ApplicationRealm** security realm is used by default.

Table 3.8. RealmDirect Login Module Options

Option	Type	Default	Description
realm	String	ApplicationRealm	Name of the desired realm.

NOTE

The **RealmDirect** login module uses **realm** only for legacy security and not for Elytron.

3.13. REALMUSERSROLES LOGIN MODULE

Short name: RealmUsersRoles

Full name: org.jboss.as.security.RealmUsersRolesLoginModule

Parent: UsersRoles Login Module

A login module which can authenticate users from given realm. Used for remoting calls. Use of RealmDirect is recommended instead of **RealmUsersRoles**.

Table 3.9. RealmUsersRoles Login Module Options

Option	Type	Default	Description
realm	String	ApplicationRealm	Name of the desired realm.
hashAlgorithm	String	REALM	Static value set by login module for option from inherited UsernamePassword Login Module.
hashStorePassword	String	false	Static value set by login module for option from inherited UsernamePassword Login Module.

NOTE

The **RealmUsersRoles** login module uses **realm** only for legacy security and not for Elytron.

CHAPTER 4. LOGIN MODULES WITH EXTERNAL IDENTITY STORE

4.1. DATABASE LOGIN MODULE

Short name: Database

Full name: org.jboss.security.auth.spi.DatabaseServerLoginModule

Parent: UsernamePassword Login Module

The Database login module is a JDBC login module that supports authentication and role mapping. This login module is used if user name, password and role information are stored in a relational database. This works by providing a reference to logical tables containing Principals and Roles in the expected format.

Table 4.1. Database Login Module Options

Option	Type	Default	Description
dsJndiName	A JNDI resource	java:/DefaultDS	The name of the JNDI resource storing the authentication information.
principalsQuery	prepared SQL statement	select **Password** from **Principals** where **PrincipalID=?**	The prepared SQL query to obtain the information about the principal.
rolesQuery	prepared SQL statement	none	The prepared SQL query to obtain the information about the roles. It should be equivalent to query 'select **Role**, **RoleGroup** from **Roles** where **PrincipalID**=?', where **Role** is the role name and the **RoleGroup** column value should always be either **Roles** with a capital **R** or **CallerPrincipal**.
suspendResume	boolean	true	Whether any existing JTA transaction should be suspended during database operations.

Option	Type	Default	Description
transactionManagerJndiName	JNDI Resource	java:/TransactionManager	The JNDI name of the transaction manager used by the login module.

4.2. DATABASEUSERS LOGIN MODULE

Short name: DatabaseUsers

Full name: org.jboss.security.DatabaseUsers

Alias to Database Login Module for compatibility reasons.

4.3. LDAP LOGIN MODULE

Short name: Ldap

Full name: org.jboss.security.auth.spi.LdapLoginModule

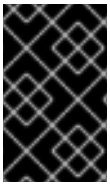
Parent: UsernamePassword Login Module

The **Ldap** login module is a login module implementation that authenticates against an LDAP server. The **security** subsystem connects to the LDAP server using connection information, a **java.naming.security.principal** that has permissions to search both the **baseCtxDN** and **rolesCtxDN** trees for the user and roles, provided using a JNDI initial context. When a user attempts to authenticate, the **Ldap** login module connects to the LDAP server, and passes the user's credentials to the LDAP server.

These credentials are formed by concatenating **principalDNPrefix**, the user input, and **principalDNSuffix**. For instance, consider the following scenario.

1. **principalDNPrefix** is set to **uid=**.
2. **principalDNSuffix** is set to **,ou=People,dc=jboss,dc=org**.

If the user input is set to **jduke**, then the search string is **uid=jduke,ou=People,dc=jboss,dc=org**. If the user input is instead **jduke,ou=Employees**, then the search string would be **uid=jduke,ou=Employees,ou=People,dc=jboss,dc=org**.



IMPORTANT

Any user input is transformed into a string before the search is performed. Due to this any special characters, such as commas, must be escaped for the search to function successfully.

Upon successful authentication, an **InitialLDAPContext** is created for that user within JBoss EAP, populated with the user's roles.

Table 4.2. Ldap Login Module Options

Option	Type	Default	Description
principalDNPrefix	String		Prefix added to the user name to form the user DN. You can prompt the user for a user name and build the fully qualified DN by using the **principalDNPrefix** and **principalDNSuffix**.
principalDNSuffix	String		Suffix added to the user name to form the user DN. You can prompt the user for a user name and build the fully qualified DN by using the **principalDNPrefix** and **principalDNSuffix**.
rolesCtxDN	fully qualified DN	none	The fully qualified DN for the context to search for user roles.
userRolesCtxDNAttributeName	attribute	none	The attribute in the user object that contains the DN for the context to search for user roles. This differs from **rolesCtxDN** in that the context to search for a user's roles may be unique for each user.
roleAttributeID	attribute	roles	Name of the attribute containing the user roles.
roleAttributeIsDN	true or false	false	Whether or not the **roleAttributeID** contains the fully qualified DN of a role object. If false, the role name is taken from the value of the **roleNameAttributeId** attribute of the context name. Certain directory schemas, such as Microsoft Active Directory, require this attribute to be set to true.
roleNameAttributeID	attribute	name	Name of the attribute within the **roleCtxDN** context which contains the role name. If the roleAttributeIsDN property is set to true, this property is used to find the role object's name attribute.
uidAttributeID	attribute	uid	Name of the attribute in the **UserRolesAttributeDN** that corresponds to the user ID. This is used to locate the user roles.

Option	Type	Default	Description
matchOnUserDN	true or false	false	Whether or not the search for user roles should match on the user's fully distinguished DN or the user name only. If true, the full user DN is used as the match value. If false, only the user name is used as the match value against the **uidAttributeName** attribute.
allowEmptyPasswords	true or false	false	Whether to allow empty passwords. Most LDAP servers treat empty passwords as anonymous login attempts. To reject empty passwords, set this to false.
searchTimeLimit	integer	10000, or 10 seconds	The timeout in milliseconds for user or role searches.
searchScope	One of: **OBJECT_SCOPE**, **ONELEVEL_SCOPE**, **SUBTREE_SCOPE**	**SUBTREE_SCOPE**	The search scope to use.
jaasSecurityDomain	String	none	The JMX ObjectName of the **JaasSecurityDomain** used to decrypt the **java.naming.security.credentials**. The encrypted form of the password is returned by the **decode64(String)** method which is called on the object passed in this option.

NOTE

For information about additional LDAP context properties related to connecting to an LDAP server and creating an initial context, see LDAP Connectivity Options.

NOTE

While this login module does inherit the **ignorePasswordCase** option from its parent, UsernamePassword Login Module, it is not used by this specific login module.

4.4. LDAPEXTENDED LOGIN MODULE

Short name: LdapExtended

Full name: org.jboss.security.auth.spi.LdapExtLoginModule

Parent: UsernamePassword Login Module

The **LdapExtended** login module searches for the user to bind, as well as the associated roles, for authentication. The roles query recursively follows DNs to navigate a hierarchical role structure. The login module options include whatever options are supported by the chosen LDAP JNDI provider supports.

The authentication happens in two steps:

1. An initial bind to the LDAP server is done using the bindDN and bindCredential options. The **bindDN** is an LDAP user with the ability to search both the **baseCtxDN** and **rolesCtxDN** trees for the user and roles. The user DN to authenticate against is queried using the filter specified by the **baseFilter** attribute.

2. The resulting user DN is authenticated by binding to the LDAP server using the user DN as a principal name and the password obtained by the callback handler as the principal's credentials.

Table 4.3. LdapExtended Login Module Options

Option	Type	Default	Description
baseCtxDN	fully qualified DN	none	The fixed DN of the top-level context to begin the user search.
bindCredential	String, optionally encrypted	none	Used to store the credentials for the DN.
bindDN	fully qualified DN	none	The DN used to bind against the LDAP server for the user and roles queries. This DN needs read and search permissions on the **baseCtxDN** and **rolesCtxDN** values.
baseFilter	LDAP filter String	none	A search filter used to locate the context of the user to authenticate. The input **username** or **userDN** obtained from the login module callback is substituted into the filter anywhere a **{0}** expression is used. A common example for the search filter is **(uid={0})**.

Option	Type	Default	Description
jaasSecurityDomain	String	none	The JMX ObjectName of the **JaasSecurityDomain** used to decrypt the password.
rolesCtxDN	fully qualified DN	none	The fixed DN of the context to search for user roles. This is not the DN where the actual roles are, but the DN where the objects containing the user roles are. For example, in a Microsoft Active Directory server, this is the DN where the user account is.
roleFilter	LDAP filter String	none	A search filter used to locate the roles associated with the authenticated user. The input **username** or **userDN** obtained from the login module callback is substituted into the filter anywhere a **{0}** expression is used. The authenticated **userDN** is substituted into the filter anywhere a **{1}** is used. An example search filter that matches on the input username is **(member={0})**. An alternative that matches on the authenticated userDN is **(member={1})**.
roleAttributeID	attribute	role	Name of the attribute containing the user roles.

Option	Type	Default	Description
roleAttributeIsDN	true or false	false	Whether or not the **roleAttributeID** contains the fully qualified DN of a role object. If false, the role name is taken from the value of the **roleNameAttributeId** attribute of the context name. Certain directory schemas, such as Microsoft Active Directory, require this attribute to be set to **true**.
defaultRole	Role name	none	A role included for all authenticated users
parseRoleNameFromDN	true or false	false	A flag indicating if the DN returned by a query contains the **roleNameAttributeID**. If set to **true**, the DN is checked for the **roleNameAttributeID**. If set to **false**, the DN is not checked for the **roleNameAttributeID**. This flag can improve the performance of LDAP queries.
parseUsername	true or false	false	A flag indicating if the DN is to be parsed for the **username**. If set to **true**, the DN is parsed for the user name. If set to **false** the DN is not parsed for the user name. This option is used together with **usernameBeginString** and **usernameEndString**.

Option	Type	Default	Description
usernameBeginString	String	none	Defines the String which is to be removed from the start of the DN to reveal the username. This option is used together with **usernameEndString** and only taken into account if **parseUsername** is set to **true**.
usernameEndString	String	none	Defines the String which is to be removed from the end of the DN to reveal the **username**. This option is used together with **usernameBeginString** and only taken into account if **parseUsername** is set to **true**.
roleNameAttributeID	attribute	name	Name of the attribute within the **roleCtxDN** context which contains the role name. If the **roleAttributeIsDN** property is set to true, this property is used to find the role object's name attribute.
distinguishedNameAttribute	attribute	distinguishedName	The name of the attribute in the user entry that contains the DN of the user. This may be necessary if the DN of the user itself contains special characters, backslash for example, that prevent correct user mapping. If the attribute does not exist, the entry's DN is used.

Option	Type	Default	Description
roleRecursion	integer	0	The numbers of levels of recursion the role search will go below a matching context. Disable recursion by setting this to 0.
searchTimeLimit	integer	10000, or 10 seconds	The timeout in milliseconds for user or role searches.
searchScope	One of: **OBJECT_SCOPE**, **ONELEVEL_SCOPE**, **SUBTREE_SCOPE**	**SUBTREE_SCOPE**	The search scope to use.
allowEmptyPasswords	true or false	false	Whether to allow empty passwords. Most LDAP servers treat empty passwords as anonymous login attempts. To reject empty passwords, set this to **false**.
referralUserAttributeIDToCheck	attribute	none	If you are not using referrals, this option can be ignored. When using referrals, this option denotes the attribute name which contains users defined for a certain role, for example **member**, if the role object is inside the referral. Users are checked against the content of this attribute name. If this option is not set, the check will always fail, so role objects cannot be stored in a referral tree.

NOTE

For information about additional LDAP context properties related to connecting to an LDAP server and creating an initial context, see LDAP Connectivity Options.

NOTE

While this login module does inherit the **ignorePasswordCase** option from its parent, UsernamePassword Login Module, it is not used by this specific login module.

NOTE

In cases when you are using Microsoft Active Directory with a **crossRef** object for creating referrals, you should take into account that LDAP Login Modules use only one value for **baseCtxDN** and only one value for **rolesCtxDN**. For that reason, initial users and roles should be stored in one Microsoft Active Directory domain to accommodate the possibility of using LDAP referrals.

4.5. ADVANCEDLDAP LOGIN MODULE

Short name: AdvancedLdap

Full name: org.jboss.security.negotiation.AdvancedLdapLoginModule

Parent: Common Login Module

The **AdvancedLdap** login module is a module which provides additional functionality, such as SASL and the use of a JAAS security domain. In cases where users wish to use LDAP with the SPNEGO authentication or skip some of the authentication phases while using an LDAP server, consider using the **AdvancedLdap** login module chained with the SPNEGO login module or only the **AdvancedLdap** login module.

AdvancedLdap login module differs from **LdapExtended** login module in the following ways:

- The top level role is queried only for **roleAttributeID** and not for **roleNameAttributeID**.
- When the **roleAttributeIsDN** module property is set to **false**, the recursive role search is disabled even if the **recurseRoles** module option is set to **true**.

Table 4.4. AdvancedLdap Login Module Options

Option	Type	Default	Description
bindDN	fully qualified DN	none	The DN used to bind against the LDAP server for the user and roles queries. This DN needs read and search permissions on the **baseCtxDN** and **rolesCtxDN** values.
bindCredential	String, optionally encrypted	none	Used to store the credentials for the DN.

Option	Type	Default	Description
jaasSecurityDomain	String	none	The JMX ObjectName of the **JaasSecurityDomain** used to decrypt the password.
java.naming.provider.url	String	If the value of **java.naming.security.protocol** is **SSL**, ldap://localhost:686, otherwise ldap://localhost:389	The URI of the directory server.
baseCtxDN	fully qualified DN	none	The distinguished name to use as the base for searches.
baseFilter	String representing an LDAP search filter.	none	The filter to use to narrow down search results.
searchTimeLimit	integer	10000, or 10 seconds	The timeout in milliseconds for user or role searches.
roleAttributeID	String value representing an LDAP attribute.	none	The LDAP attribute which contains the names of authorization roles.
roleAttributeIsDN	true or false	false	Whether the role attribute is a Distinguished Name, DN.
rolesCtxDN	fully qualified DN	none	The fully qualified DN for the context to search for user roles.

Option	Type	Default	Description
roleFilter	LDAP filter String	none	A search filter used to locate the roles associated with the authenticated user. The input user name or userDN obtained from the login module callback is substituted into the filter anywhere a **{0}** expression is used. The authenticated userDN is substituted into the filter anywhere a **{1}** is used. An example search filter that matches on the input user name is (**member={0}**). An alternative that matches on the authenticated userDN is (**member={1}**).
recurseRoles	true or false	false	Whether to recursively search the **roleAttributeID** for roles.
roleNameAttributeID	String representing an LDAP attribute.	none	The attribute contained within the **roleAttributeID** which contains the actual role attribute.

Option	Type	Default	Description
referralUserAttributeIDToCheck	attribute	none	If you are not using referrals, this option can be ignored. When using referrals, this option denotes the attribute name which contains users defined for a certain role, for example **member**, if the role object is inside the referral. Users are checked against the content of this attribute name. If this option is not set, the check will always fail, so role objects cannot be stored in a referral tree.
searchScope	One of: **OBJECT_SCOPE**, **ONELEVEL_SCOPE**, **SUBTREE_SCOPE**	**SUBTREE_SCOPE**	The search scope to use.
allowEmptyPassword	true or false	false	Whether to allow empty passwords. Most LDAP servers treat empty passwords as anonymous login attempts. To reject empty passwords, set this to false.
bindAuthentication	String	If the system property **java.naming.security.authentication** is set, it uses that value, otherwise it defaults to **simple**.	The type of SASL authentication to use for binding to the directory server.

NOTE

For information about additional LDAP context properties related to connecting to an LDAP server and creating an initial context, see LDAP Connectivity Options.

NOTE

In cases when you are using Microsoft Active Directory with a **crossRef** object for creating referrals, you should take into account that LDAP Login Modules use only one value for **baseCtxDN** and only one value for **rolesCtxDN**. For that reason, initial users and roles should be stored in one Microsoft Active Directory domain to accommodate the possibility of using LDAP referrals.

4.6. ADVANCEDADLDAP LOGIN MODULE

Short name: AdvancedAdLdap

Full name: org.jboss.security.negotiation.AdvancedADLoginModule

Parent: AdvancedLdap Login Module

The **AdvancedAdLdap** login module adds extra parameters that are relevant to Microsoft Active Directory but has no additional configurable options beyond the ones available in AdvancedLdap Login Module.

NOTE

For information about additional LDAP context properties related to connecting to an LDAP server and creating an initial context, see LDAP Connectivity Options.

4.7. LDAP CONNECTIVITY OPTIONS

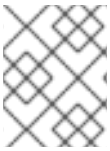
The LDAP connectivity information is provided as configuration options that are passed through to the environment object used to create JNDI initial context. These configuration options can be utilized by the Ldap Login Module, LdapExtended Login Module, AdvancedLdap Login Module, and AdvancedAdLdap Login Module.

The standard LDAP JNDI properties used include the following:

Option	Type	Default	Description
java.naming.factory.initial	class name	com.sun.jndi.ldap.LdapCtxFactory	InitialContextFactory implementation class name.
java.naming.provider.url	ldap:// URL	If the value of java.naming.security.protocol is SSL, ldap://localhost:636, otherwise ldap://localhost:389	URL for the LDAP server.
java.naming.security.authentication	none, simple, or the name of a SASL mechanism	The default is **simple**. If the property is explicitly undefined, the behavior is determined by the service provider.	The security level to use to bind to the LDAP server.

Option	Type	Default	Description
java.naming.security.protocol	transport protocol	If unspecified, determined by the provider.	The transport protocol to use for secure access, such as SSL.
java.naming.security.principal	String	none	The name of the principal for authenticating the caller to the service. This is built from other properties described below.
java.naming.security.credentials	credential type	none	The type of credential used by the authentication scheme. Some examples include hashed password, clear-text password, key, or certificate. If this property is unspecified, the behavior is determined by the service provider.

User authentication is performed by connecting to the LDAP server, based on the login module configuration options. Connecting to the LDAP server is done by creating an **InitialLdapContext** with an environment composed of the LDAP JNDI properties. The initial context implementation that is actually used depends on the initial context factory method configured. The initial context factory is defined using the **java.naming.factory.initial** property and gets its configuration from environment properties provided, for example **java.naming.provider.url**. This allows for arbitrary properties, as well as related login module options, to be used for custom initial context factories.



NOTE

Additional default and common options available for creating an initial context available in the javax.naming.Context interface javadoc.

4.8. LDAPUSERS LOGIN MODULE

Short name: LdapUsers

Full name: org.jboss.security.auth.spi.LdapUsersLoginModule

Parent: UsernamePassword Login Module

The **LdapUsers** module is superseded by the LdapExtended and AdvancedLdap modules.

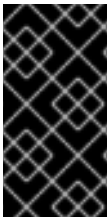
4.9. KERBEROS LOGIN MODULE

Short name: Kerberos

Full name: org.jboss.security.negotiation.KerberosLoginModule

The **Kerberos** login module performs Kerberos login authentication, using GSSAPI. This login module wraps the JDK supplied module, **com.sun.security.auth.module.Krb5LoginModule** for the Oracle JDK and **com.ibm.security.auth.module.Krb5LoginModule** for the IBM JDK, and provides additional logic for credential delegation and adding a **GSSCredential** to the populated Subject.

This module needs to be paired with another module which handles the authentication and roles mapping.



IMPORTANT

The below table lists the options available for **org.jboss.security.negotiation.KerberosLoginModule**, but options from the module supplied by the JDK can also be configured. For more details on each JDK's module options, please consult the Oracle and IBM Javadocs.

Table 4.5. Kerberos Login Module Options

Option	Type	Default	Description
delegationCredential	**IGNORE**, **REQUIRE**, or **USE**	**IGNORE**	Defines how this login module handles delegation. **IGNORE** specifies to not use the delegate credential and to perform normal Kerberos authentication. **USE** specifies to use a **GSSCredential** if available to populate a Subject, otherwise fall back to standard Kerberos authentication if unavailable. **REQUIRE** specifies to use a **GSSCredential** and fail authentication if one is not available.
addGSSCredential	boolean	false	Enables adding a **GSSCredential** to the private credentials of the populated Subject.
wrapGSSCredential	boolean	false	Specifies if any **GSSCredential** being added to the Subject should be wrapped to prevent disposal. This has no effect if a **GSSCredential** is not being added to the Subject.

Option	Type	Default	Description
credentialLifetime	integer	**GSSCredential.DEFAULT_LIFETIME**	The lifetime in seconds of the **GSSCredential**, a negative value will set this to **GSSCredential.INDEFINITE_LIFETIME**.

4.10. SPNEGO LOGIN MODULE

Short name: SPNEGO

Full name: org.jboss.security.negotiation.spnego.SPNEGOLoginModule

Parent: Common Login Module

The SPNEGO login module is an implementation of login module that establishes caller identity and credentials with a KDC. The module implements SPNEGO, Simple and Protected GSSAPI Negotiation mechanism, and is a part of the JBoss Negotiation project. This authentication can be used in the chained configuration with the **AdvancedLdap** login module to allow cooperation with an LDAP server.

Table 4.6. SPNEGO Login Module Options

Option	Type	Default	Description
serverSecurityDomain	String	null	Defines the domain that is used to retrieve the identity of the server service through the kerberos login module. This property must be set.
removeRealmFromPrincipal	boolean	false	Specifies that the Kerberos realm should be removed from the principal before further processing.
usernamePasswordDomain	String	null	Specifies another security domain within the configuration that should be used as a failover login when Kerberos fails.

CHAPTER 5. CERTIFICATE-BASED LOGIN MODULES

5.1. CERTIFICATE LOGIN MODULE

Short name: Certificate

Full name: org.jboss.security.auth.spi.BaseCertLoginModule

Parent: AbstractServer Login Module

Certificate login module authenticates users based on X509 certificates. A typical use case for this login module is **CLIENT-CERT** authentication in the web tier. This login module only performs authentication and must be combined with another login module capable of acquiring authorization roles to completely define access to a secured web or EJB components. Two subclasses of this login module, **CertRoles Login Module** and **DatabaseCert Login Module** extend the behavior to obtain the authorization roles from either a properties file or database.

Table 5.1. Certificate Login Module Options

Option	Type	Default	Description
securityDomain	String	other	Name of the security domain that has the JSSE configuration for the truststore holding the trusted certificates.
verifier	class	none	The class name of the **org.jboss.security.auth.certs.X509CertificateVerifier** to use for verification of the login certificate.

5.2. CERTIFICATEROLES LOGIN MODULE

Short name: CertificateRoles

Full name: org.jboss.security.auth.spi.CertRolesLoginModule

Parent: Certificate Login Module

The **CertificateRoles** login module adds role mapping capabilities from a properties file using the following options:

Table 5.2. CertificateRoles Login Module Options

Option	Type	Default	Description

Option	Type	Default	Description
rolesProperties	String	roles.properties	The name of the resource or file containing the roles to assign to each user. The role properties file must be in the format **username=role1,role2** where the user name is the DN of the certificate, escaping any equals and space characters. The following example is in the correct format: **CN\=unit-tests-client,\ OU\=Red\ Hat\ Inc.,\ O\=Red\ Hat\ Inc.,\ ST\=North\ Carolina,\ C\=US**
defaultRolesProperties	String	defaultRoles.properties	Name of the resource or file to fall back to if the **rolesProperties** file cannot be found.
roleGroupSeparator	A single character.	. (a single period)	Which character to use as the role group separator in the **rolesProperties** file.

5.3. DATABASECERTIFICATE LOGIN MODULE

Short name: DatabaseCertificate

Full name: org.jboss.security.auth.spi.DatabaseCertLoginModule

Parent: Certificate Login Module

The **DatabaseCertificate** login module adds mapping capabilities from a database table through these additional options:

Table 5.3. DatabaseCertificate Login Module Options

Option	Type	Default	Description
dsJndiName	A JNDI resource	java:/DefaultDS	The name of the JNDI resource storing the authentication information.

Option	Type	Default	Description
rolesQuery	prepared SQL statement	select **Role**,**RoleGroup** from **Roles** where **PrincipalID=?**	SQL prepared statement to be executed in order to map roles. It should be an equivalent to the query 'select **Role**, **RoleGroup** from **Roles** where **PrincipalID**=?', where **Role** is the role name and the **RoleGroup** column value should always be either **Roles** with a capital **R** or **CallerPrincipal**.
suspendResume	true or false	true	Whether any existing JTA transaction should be suspended during database operations.
transactionManagerJndiName	JNDI Resource	java:/TransactionManager	The JNDI name of the transaction manager used by the login module.

CHAPTER 6. LOGIN MODULES FOR EJBS AND REMOTING

6.1. REMOTING LOGIN MODULE

Short name: Remoting

Full name: org.jboss.as.security.remoting.RemotingLoginModule

Parent: AbstractServer Login Module

The **Remoting** login module allows remote EJB invocations, coming in over remoting, to perform a SASL-based authentication. This allows the remote user to establish their identity via SASL and have that identity be used for authentication and authorization when making that EJB invocation.

Table 6.1. Remoting Login Module Options

Option	Type	Default	Description
useClientCert	boolean	false	If **true**, the login module will obtain the **SSLSession** of the connection and substitute the peer's **X509Certificate** in place of the password.

6.2. CLIENT LOGIN MODULE

Short name: Client

Full name: org.jboss.security.ClientLoginModule

Client login module is an implementation of login module for use by JBoss EAP clients when establishing caller identity and credentials. This creates a new **SecurityContext**, assigns it a principal and a credential and sets the **SecurityContext** to the **ThreadLocal** security context. Client login module is the only supported mechanism for a client to establish the current thread's caller. Both standalone client applications, and server environments, acting as JBoss EAP EJB clients where the security environment has not been configured to use the JBoss EAP **security** subsystem transparently, must use Client login module.

WARNING

This login module does not perform any authentication. It merely copies the login information provided to it into the server EJB invocation layer for subsequent authentication on the server. Within JBoss EAP, this is only supported for the purpose of switching a user's identity for in-JVM calls. This is **NOT** supported for remote clients to establish an identity.

Table 6.2. Client Login Module Options

Option	Type	Default	Description
multi-threaded	true or false	true	Set to true if each thread has its own principal and credential storage. Set to false to indicate that all threads in the VM share the same identity and credential.
password-stacking	**useFirstPass** or false	false	Set to **useFirstPass** to indicate that this login module should look for information stored in the **LoginContext** to use as the identity. This option can be used when stacking other login modules with this one.
restore-login-identity	true or false	false	Set to true if the identity and credential seen at the start of the **login()** method should be restored after the **logout()** method is invoked.

CHAPTER 7. ABOUT PICKETLINK LOGIN MODULES

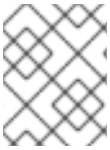
A PicketLink login module is typically configured as part of the security setup to use a Security Token Service (STS) or browser-based SSO with SAML for authenticating users. The STS may be collocated on the same container as the login module or be accessed remotely through web service calls or another technology. PicketLink STS login modules support non-PicketLink STS implementations through standard WS-Trust calls. For more details on the concepts behind Security Token Services as well as browser-based SSO with SAML, please see the JBoss EAP *Security Architecture* guide.

7.1. STSISSUINGLOGINMODULE

Full name: org.picketlink.identity.federation.core.wstrust.auth.STSIssuingLoginModule

The *STSIssuingLoginModule* uses a user name and password to authenticate the user against an STS by retrieving a token. The authentication happens as follows:

- Calls the configured STS and requests for a security token. Upon successfully receiving the *RequestedSecurityToken*, it marks the authentication as successful.
- A call to the STS typically requires authentication. This login module uses credentials from one of the following sources:
  - Its properties file, if the *useOptionsCredentials* module option is set to *true*.
  - Previous login module credentials if the *password-stacking* module option is set to *useFirstPass*.
  - From the configured *CallbackHandler* by supplying a Name and Password Callback.
- Upon successful authentication, the security token is stored in the login module's shared map with **org.picketlink.identity.federation.core.wstrust.lm.stsToken** key.



NOTE

This login module has no direct configurable attributes, but you may use module options to pass in configuration options.

Example STSIssuingLoginModule

```
<security-domain name="saml-issue-token">
  <authentication>
    <login-module code="org.picketlink.identity.federation.core.wstrust.auth.STSIssuingLoginModule"
flag="required">
      <module-option name="configFile">./picketlink-sts-client.properties</module-option>
      <module-option name="endpointURI">http://security_saml/endpoint</module-option>
    </login-module>
  </authentication>
  <mapping>
    <mapping-module
code="org.picketlink.identity.federation.bindings.jboss.auth.mapping.STSPrincipalMappingProvider"
type="principal"/>
    <mapping-module
code="org.picketlink.identity.federation.bindings.jboss.auth.mapping.STSGroupMappingProvider"
```

```
type="role" />
  </mapping>
</security-domain>
```

In the above example, the specified Principal mapping provider and the RoleGroup mapping provider results in an authenticated Subject being populated that enables coarse-grained and role-based authorization. After authentication, the Security Token is available and may be used to invoke other services by Single Sign-On.

7.2. STSVALIDATINGLOGINMODULE

Full name: org.picketlink.identity.federation.core.wstrust.auth.STSValidatingLoginModule

The *STSValidatingLoginModule* uses a TokenCallback to retrieve a security token from STS.

The authentication happens as follows:

- Calls the configured STS and validates an available security token.
- A call to STS typically requires authentication. This Login Module uses credentials from one of the following sources:
  - Its properties file, if the *useOptionsCredentials* module option is set to *true*.
  - Previous login module credentials if the *password-stacking* module option is set to *useFirstPass*.
  - From the configured *CallbackHandler* by supplying a Name and Password Callback.
- Upon successful authentication, the security token is stored in the login module's shared map with **org.picketlink.identity.federation.core.wstrust.lm.stsToken** key.

NOTE

This login module has no direct configurable attributes, but you may use module options to pass in configuration options.

Example STSValidatingLoginModule

```
<security-domain name="saml-validate-token">
  <authentication>
    <login-module
code="org.picketlink.identity.federation.core.wstrust.auth.STSValidatingLoginModule"
flag="required">
      <module-option name="configFile">./picketlink-sts-client.properties</module-option>
      <module-option name="endpointURI">http://security_saml/endpoint</module-option>
    </login-module>
  </authentication>
  <mapping>
    <mapping-module
code="org.picketlink.identity.federation.bindings.jboss.auth.mapping.STSPrincipalMappingProvider"
type="principal"/>
    <mapping-module
code="org.picketlink.identity.federation.bindings.jboss.auth.mapping.STSGroupMappingProvider"
```

```
type="role"/>
  </mapping>
</security-domain>
```

The above example shows how to enable validation for an issued token, either directly by contacting the STS or through a token-issuing login module, to be used to authenticate against multiple applications and services. Providing a Principal mapping provider and a RoleGroup mapping provider results in an authenticated Subject being populated that enables coarse-grained and role-based authorization. After authentication, the Security Token is available and can be used to invoke other services by Single Sign-On.

7.3. SAML2STSLOGINMODULE

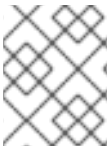
Full name: org.picketlink.identity.federation.bindings.jboss.auth.SAML2STSLoginModule

The authentication happens as follows:

- This Login Module supplies an *ObjectCallback* to the configured *CallbackHandler* and expects a *SamlCredential* object back. The Assertion is validated against the configured STS.
- Upon successful authentication, the *SamlCredential* is inspected for a *NameIDType*.
- If a user ID and SAML token are shared, this Login Module bypasses validation when stacked on top of another Login Module that is successfully authenticated.

Example SAML2STSLoginModule

```
<security-domain name="saml-sts" cache-type="default">
  <authentication>
    <login-module
code="org.picketlink.identity.federation.bindings.jboss.auth.SAML2STSLoginModule" flag="required"
module="org.picketlink">
      <module-option name="configFile" value="${jboss.server.config.dir}/sts-config.properties"/>
      <module-option name="password-stacking" value="useFirstPass"/>
    </login-module>
  </authentication>
</security-domain>
```



NOTE

This login module has no direct configurable attributes, but you may use module options to pass in configuration options.

7.4. SAML2LOGINMODULE

Full name: org.picketlink.identity.federation.bindings.jboss.auth.SAML2LoginModule

The authentication happens as follows:

- This login module is used in conjunction with other components for SAML authentication and performs no authentication itself.
- The SAML authenticator, which is installed by the PicketLink Service Provider Undertow ServletExtension

(**org.picketlink.identity.federation.bindings.wildfly.sp.SPServletExtension**), uses this login module to authenticate users based on a SAML assertion previously issued by an identity provider.

- If the user does not have a SAML assertion for the service provider, the user is redirected to the identity provider to obtain a SAML assertion.
- This login module is used to pass the user ID and roles to the security framework to be populated in the JAAS subject.

Example SAML2LoginModule

```
<security-domain name="sp" cache-type="default">
  <authentication>
    <login-module code="org.picketlink.identity.federation.bindings.jboss.auth.SAML2LoginModule"
flag="required"/>
  </authentication>
</security-domain>
```

NOTE

This login module has no direct configurable attributes.

WARNING

The **SAML2LoginModule** is intended for use with applications using PicketLink with SAML and should not be used without the PicketLink Service Provider Undertow ServletExtension (**org.picketlink.identity.federation.bindings.wildfly.sp.SPServletExtension**). Doing so presents a possible security risk since the **SAML2LoginModule** or **SAML2CommonLoginModule** will always accept the default password of **EMPTY_STR**. For example, this can also occur if the PicketLink Service Provider Undertow ServletExtension is not installed in the SP application. The PicketLink Service Provider Undertow ServletExtension is installed automatically when configuring the SP application for JBoss EAP . This can also occur if the **SAML2LoginModule** is stacked with other login modules:

```
<security-domain name="sp" cache-type="default">
  <authentication>
    <login-module
code="org.picketlink.identity.federation.bindings.jboss.auth.SAML2LoginModule"
flag="optional">
      <module-option name="password-stacking" value="useFirstPass"/>
    </login-module>
    <login-module code="UsersRoles" flag="required">
      <module-option name="usersProperties" value="users.properties"/>
      <module-option name="rolesProperties" value="roles.properties"/>
      <module-option name="password-stacking" value="useFirstPass"/>
    </login-module>
  </authentication>
</security-domain>
```

7.5. REGEXUSERNAMELOGINMODULE

Full name: org.picketlink.identity.federation.bindings.jboss.auth.RegExUserNameLoginModule

This login module can be used after any Certificate Login Module to extract a username, UID or other field from the principal name so that roles can be obtained from LDAP. The module has an option named **regex** which specifies the regular expression to be applied to the principal name, the result of which is passed on to the subsequent login module.

Example RegExUserNameLoginModule

```
<login-module
code="org.picketlink.identity.federation.bindings.jboss.auth.RegExUserNameLoginModule"
flag="required">
  <module-option name="password-stacking" value="useFirstPass"/>
  <module-option name="regex" value="UID=(.*?),"/>
</login-module>
```

For example, an input principal name of **UID=007, EMAILADDRESS=something@something, CN=James Bond, O=SpyAgency** would result in the output **007** using the above login module.

For more information on regular expressions, see the **java.util.regex.Pattern** class documentation.

CHAPTER 8. CUSTOM LOGIN MODULES

In cases where the login modules bundled with the JBoss EAP security framework do not meet the needs of the security environment, a custom login module implementation may be written. The **org.jboss.security.AuthenticationManager** requires a particular usage pattern of the Subject principals set. A full understanding of the JAAS Subject class's information storage features and the expected usage of these features are required to write a login module that works with the **org.jboss.security.AuthenticationManager**. Custom login modules must be implementations of **javax.security.auth.spi.LoginModule**. Refer to the API documentation for more information about creating a custom authentication module.

CHAPTER 9. AUTHORIZATION MODULES

The following modules provide authorization services:

Code	Class
DenyAll	org.jboss.security.authorization.modules.AllDenyAuthorizationModule
PermitAll	org.jboss.security.authorization.modules.AllPermitAuthorizationModule
Delegating	org.jboss.security.authorization.modules.DelegatingAuthorizationModule
Web	org.jboss.security.authorization.modules.web.WebAuthorizationModule
JACC	org.jboss.security.authorization.modules.JACCAuthorizationModule
XACML	org.jboss.security.authorization.modules.XACMLAuthorizationModule

AbstractAuthorizationModule

This is the base authorization module which has to be overridden and provides a facility for delegating to other authorization modules. This base authorization module also provides a **delegateMap** property to the overriding class, which allows for delegation modules to be declared for specific components. This enables more specialized classes to handle the authorization for each layer, for example **web**, **ejb**, etc, since the information used to authorize a user may vary between the resources being accessed. For instance, an authorization module may be based on permissions, yet have different permission types for the **web** and **ejb** resources. By default, the authorization module would be forced to deal with all possible resource and permission types, but configuring the **delegateMap** option allows the module to delegate to specific classes for different resource types. The **delegateMap** option takes a comma-separated list of modules, each of which is prefixed by the component it relates to, for example **<module-option name="delegateMap">web=xxx.yyy.MyWebDelegate,ejb=xxx.yyy.MyEJBDelegate</module-option>**.

IMPORTANT

When configuring the **delegateMap** option, every delegate must implement the **authorize(Resource)** method and have it call the **invokeDelegate(Resource)** method in same way the provided authorization modules do. Failure to do so will result in the delegate not getting called.

AllDenyAuthorizationModule

This is a simple authorization module that always denies an authorization request. No configuration options are available.

AllPermitAuthorizationModule

This is a simple authorization module that always permits an authorization request. No configuration options are available.

DelegatingAuthorizationModule

This is the default authorization module that delegates decision making to the configured delegates. This module also supports the **delegateMap** option.

WebAuthorizationModule

This is the default web authorization module with the default Tomcat authorization logic, permit all.

JACCAuthorizationModule

This module enforces JACC semantics using two delegates, **WebJACCPolicyModuleDelegate** for web container authorization requests and **EJBJACCPolicyModuleDelegate** for EJB container requests. This module also supports the **delegateMap** option.

XACMLAuthorizationModule

This module enforces XACML authorization using two delegates for web and EJB containers, **WebXACMLPolicyModuleDelegate** and **EJBXACMLPolicyModuleDelegate**. It creates a PDP object based on registered policies and evaluates web or EJB requests against it. This module also supports the **delegateMap** option.

CHAPTER 10. SECURITY MAPPING MODULES

The following security mapping modules are provided in JBoss EAP.

Class	Code	Type
org.jboss.security.mapping.providers.role.PropertiesRolesMappingProvider	PropertiesRoles	role
org.jboss.security.mapping.providers.role.SimpleRolesMappingProvider	SimpleRoles	role
org.jboss.security.mapping.providers.DeploymentRolesMappingProvider	DeploymentRoles	role
org.jboss.security.mapping.providers.role.DatabaseRolesMappingProvider	DatabaseRoles	role
org.jboss.security.mapping.providers.role.LdapRolesMappingProvider	LdapRoles	role
org.jboss.security.mapping.providers.attribute.LdapAttributeMappingProvider	LdapAttributes	attribute
org.jboss.security.mapping.providers.DeploymentRoleToRolesMappingProvider		role
org.jboss.security.mapping.providers.attribute.DefaultAttributeMappingProvider		attribute

NOTE

The mapping module functionality is only invoked for role type mapping modules. To invoke other mapping module types, the mapping functionality needs to be invoked in the application or in a custom login module.

10.1. PROPERTIESROLESMAPPINGPROVIDER

Code: PropertiesRoles

Class: org.jboss.security.mapping.providers.role.PropertiesRolesMappingProvider

Type: role

A MappingProvider that reads roles from a properties file in the following format: **username=role1,role2,...**

Option	Type	Default	Description
rolesProperties	String	roles.properties	Properties formatted file name. Expansion of JBoss EAP variables can be used in form of **${jboss.variable}**.

10.2. SIMPLEROLESMAPPINGPROVIDER

Code: SimpleRoles

Class: org.jboss.security.mapping.providers.role.SimpleRolesMappingProvider

Type: role

A simple **MappingProvider** that reads roles from the options map. The option attribute name is the name of principal to assign roles to and the attribute value is the comma-separated role names to assign to the principal.

Example

```
<module-option name="JavaDuke" value="JBossAdmin,Admin"/>
<module-option name="joe" value="Users"/>
```

10.3. DEPLOYMENTROLESMAPPINGPROVIDER

Code: DeploymentRoles

Class: org.jboss.security.mapping.providers.DeploymentRolesMappingProvider

Type: role

A Role Mapping Module that takes into consideration a principal to roles mapping that can be done in **jboss-web.xml** and **jboss-app.xml** deployment descriptors.

Example

```
<jboss-web>
...
  <security-role>
     <role-name>Support</role-name>
     <principal-name>Mark</principal-name>
     <principal-name>Tom</principal-name>
  </security-role>
...
</jboss-web>
```

10.4. DATABASEROLESMAPPINGPROVIDER

Code: DatabaseRoles

Class: org.jboss.security.mapping.providers.role.DatabaseRolesMappingProvider

Type: role

A **MappingProvider** that reads roles from a database.

Option	Type	Default	Description
dsJndiName	String		JNDI name of data source used to map roles to the user.
rolesQuery	String		This option should be a prepared statement equivalent to **select RoleName from Roles where User=?**. **?** is substituted with current principal name.
suspendResume	boolean	true	If **true**, will suspend and later resume transaction associated with current thread while performing search for roles.
transactionManagerJndiName	String	java:/TransactionManager	JNDI name of transaction manager.

10.5. LDAPROLESMAPPINGPROVIDER

Code: LdapRoles

Class: org.jboss.security.mapping.providers.role.LdapRolesMappingProvider

Type: role

A mapping provider that assigns roles to a user using an LDAP server to search for the roles.

Option	Type	Default	Description
bindDN	String		The DN used to bind against the LDAP server for the user and roles queries. This DN needs read and search permissions on the **baseCtxDN** and **rolesCtxDN** values.
bindCredential	String		The password for the **bindDN**. This can be encrypted via the vault mechanism.
rolesCtxDN	String		The fixed DN of the context to search for user roles. This is not the DN where the actual roles are, but the DN where the objects containing the user roles are. For example, in a Microsoft Active Directory server, this is the DN where the user account is.

Option	Type	Default	Description
roleAttributeID	String	role	The LDAP attribute which contains the names of authorization roles.
roleAttributeIsDN	boolean	false	Whether or not the **roleAttributeID** contains the fully qualified DN of a role object. If **false**, the role name is taken from the value of the **roleNameAttributeId** attribute of the context name. Certain directory schemas, such as Microsoft Active Directory, require this attribute to be set to **true**.
roleNameAttributeID	String	name	Name of the attribute within the **roleCtxDN** context which contains the role name. If the **roleAttributeIsDN** property is set to **true**, this property is used to find the role object's name attribute.
parseRoleNameFromDN	boolean	false	A flag indicating if the DN returned by a query contains the **roleNameAttributeID**. If set to **true**, the DN is checked for the **roleNameATtributeID**. If set to **false**, the DN is not checked for the **roleNameAttributeID**. This flag can improve the performance of LDAP queries.
roleFilter	String		A search filter used to locate the roles associated with the authenticated user. The input **username** or **userDN** obtained from the login module callback is substituted into the filter anywhere a **{0}** expression is used. An example search filter that matches on the input **username** is **(member={0})**.
roleRecursion	number	0	The numbers of levels of recursion the role search will go below a matching context. Disable recursion by setting this to 0.
searchTimeLimit	number	10000	The timeout in milliseconds for the user/role searches.
searchScope	String	SUBTREE_SCOPE	The search scope to use.

10.6. LDAPATTRIBUTEMAPPINGPROVIDER

Code: LdapAttributes

Class: org.jboss.security.mapping.providers.attribute.LdapAttributeMappingProvider

Type: attribute

Maps attributes from LDAP to the subject. The options include whatever options your LDAP JNDI provider supports.

Examples of Standard Property Names

```
Context.INITIAL_CONTEXT_FACTORY = "java.naming.factory.initial"
Context.SECURITY_PROTOCOL = "java.naming.security.protocol"
Context.PROVIDER_URL = "java.naming.provider.url"
Context.SECURITY_AUTHENTICATION = "java.naming.security.authentication"
```

Option	Type	Default	Description
bindDN	String		The DN used to bind against the LDAP server for the user and roles queries. This DN needs read and search permissions on the **baseCtxDN** and **rolesCtxDN** values.
bindCredential	String		The password for the bindDN. This can be encrypted if the **jaasSecurityDomain** is specified.
baseCtxDN	String		The fixed DN of the context to start the user search from.
baseFilter	String		A search filter used to locate the context of the user to authenticate. The input **username** or **userDN** as obtained from the login module callback is substituted into the filter anywhere a **{0}** expression is used. This substitution behavior comes from the standard ***DirContext.search(Name, String, Object[], SearchControls cons)*** method. A common example search filter is **(uid={0})**.
searchTimeLimit	number	10000	The timeout in milliseconds for the user/role searches.
attributeList	String		A comma-separated list of attributes for the user. For example, **mail,cn,sn,employeeType,employeeNumber**.

Option	Type	Default	Description
jaasSecurityDomain	String		The **JaasSecurityDomain** to use to decrypt the **java.naming.security.credentials**. The encrypted form of the password is that returned by the **JaasSecurityDomain#decode64(String)** method. The **org.jboss.security.plugins.PBEUtils** can also be used to generate the encrypted form.

10.7. DEPLOYMENTROLETOROLESMAPPINGPROVIDER

Class: org.jboss.security.mapping.providers.DeploymentRoleToRolesMappingProvider

Type: role

A Role to Roles Mapping Module that takes into consideration a role to roles mapping. This can be defined in the deployment descriptors **jboss-web.xml** and **jboss-app.xml**. In this case, all the **principal-name** elements denote the roles that will replace the given role in **role-name**.

Example

```
<jboss-web>
...
   <security-role>
     <role-name>Employee</role-name>
     <principal-name>Support</principal-name>
     <principal-name>Sales</principal-name>
   </security-role>
...
</jboss-web>
```

In the above example, each principal having the role **Employee** will have this role replaced with **Support** and **Sales**. If it is desired for the principal to retain the **Employee** role as well as gain the **Support** and **Sales** roles, **<principal-name>Employee</principal-name>** should be added.

NOTE

This mapping provider does not have a code associated with it so the full class name must be in the **code** field when configuring.

10.8. DEFAULTATTRIBUTEMAPPINGPROVIDER

Class: org.jboss.security.mapping.providers.attribute.DefaultAttributeMappingProvider

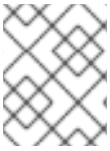
Type: attribute

Checks module and locates principal name from mapping context to create attribute email address from module option named **principalName** + **.email** and maps it to the given principal.

Example

```
<module-option name="admin.email" value="jduke@redhat.com"/>
```

In the above example, the attribute **email** with a value **jduke@redhat.com** is added for the principal **admin**.



NOTE

This mapping provider does not have a code associated with it so the full class name must be in the **code** field when configuring.

Revised on 2021-08-27 13:42:46 UTC