LDAP Authentication and User Import/Export



Michael C. Han
Actualización hace 4 años

This article provides an overview of LDAP integration with Liferay DXP 7.0. This content on authentication, user Import/Export, configurations, upgrade considerations and what has changed from previous Liferay Portal versions, is intended to help readers successfully integrate LDAP with DXP 7.0 in production.

Table of Contents:

1. LDAP Authentication
2. LDAP User Import
3. LDAP User Export
4. Configurations
5. Upgrades
6. Production Considerations

Resolution

As a best practice, please download the latest fix pack of DXP.

The Liferay platform provides three primary levels of integration for LDAP:

1. LDAP Authentication
2. LDAP User Import
3. LDAP User Export

These services are encapsulated in the portal-ldap module (com.liferay.portal.ldap.jar).

LDAP Authentication

LDAP authentication is unlike other SSO implementations within the portal (e.g. Token, CAS, OpenSSO, etc). LDAP authentication uses a lower level Liferay concept of an Authenticator. Authenticators are injected into the Liferay `AuthPipeline` and used as part of the portal's own authentication process. Other SSO implementations use a bypass mechanism (AutoLogin) which bypasses the portal's authentication process.

Implementation

`LDAPAuth` is configured via 3 sets of configuration properties:

1. `LDAPAuthConfiguration`
2. `LDAPServerConfiguration`
3. `SystemLDAPConfiguration`

Assuming successful authentication, the authenticated user is imported from LDAP to the Liferay user store.

LDAP User Import

Liferay has a generic user import service which imports users from any external system into Liferay's user store. The LDAP user import is a LDAP specific implementation of this service.

There are two paths during which users can be imported from LDAP:

1. When LDAP authentication is enabled, any user that successfully authenticates into the portal will be automatically imported into the Liferay user store.
2. When LDAP scheduled import is enabled (via `LDAPImportConfiguration.enabled`), then on a regularly configured interval (`LDAPImportConfiguration.importInterval`), users are imported from LDAP into the Liferay user store.

Implementation

`com.liferay.portal.ldap.internal.exportimport.LDAPUserImportImpl` provides the bulk of the implementation for the service. It communicates with LDAP to retrieve user and group information. `LDAPUserImportImpl` uses a LDAPToPortalConverter to convert from LDAP objects and attributes to Liferay objects (User, Contact, Group, Role).

`LDAPUserImportImpl` is configured via

1. `LDAPImportConfiguration`
2. `LDAPServerConfiguration`

Default User Password

This feature fulfills the technical requirement that the Password field cannot have a NULL value and that all users must have the same password. The use case is where the password was not imported from the LDAP server or it was not autogenerated by the Liferay platform, this function ensures that this requirement is met. You can also create a file called `com.liferay.portal.security.ldap.exportimport.configuration.LDAPImportConfiguration-default.config` and enter `importUserPasswordDefault="test"`. Deploy this in the `/osgi/module` folder.

LDAP User Export

Liferay has a generic user export service that compliments the user import service. The user export service is capable of exporting users to any external system from Liferay's user store. The LDAP user export is a LDAP specific implementation of this service.

There are a few conditions under which users would be exported:

1. When users are added to Liferay's user store
2. When user profiles are modified in Liferay's user store
3. When users are added to user groups
4. When users are removed from user groups
5. When users are added to roles
6. When users are removed from roles

Implementation

`com.liferay.portal.ldap.internal.exportimport.LDAPUserExportImpl` provides the bulk of the implementation for the service. It communicates with LDAP to store user profile information. `LDAPUserExportImpl` uses a `PortalToLDAPConverter` to convert from Liferay objects (User, Contact) to LDAP objects.

`LDAPExportUserImpl` configured via

1. `LDAPExportConfiguration`
2. `LDAPServerConfiguration`

To trigger export, the implementation uses ModelListeners:

1. Exporting user information when user and contact information changes:
    a. `com.liferay.portal.ldap.internal.exportimport.model.listener.UserModelListener`
    b. `com.liferay.portal.ldap.internal.exportimport.model.listener.ContactModelListener`
2. Exporting user information when UserGroup memberships change for users
    a. `com.liferay.portal.ldap.internal.exportimport.model.listener.UserGroupModelListener`
3. Exporting user information when Role membership changes for users
    a. `com.liferay.portal.ldap.internal.exportimport.model.listener.UserModelListener`

Configurations

Prior to DXP 7.0, LDAP related configurations were present in a couple of places:

Certain values were configurable through the platform's portal settings user interface (*Authentication -> LDAP*). This led to a lot of inconsistent behavior and attempts to decipher which values should be changed where.

After DXP 7.0, that confusion moves away. All LDAP configurations will be configured using *ConfigurationAdmin*, specifically the following configurations:

`LDAPAuthConfiguration`

- LDAPAuth specific configurations.
- Each portal instance (companyId) may have one.

`LDAPExportConfiguration`

- `LDAPUserExportImpl` specific configurations.
- Each portal instance (companyId) may have one.

`LDAPImportConfiguration`

- `LDAPUserImporttImpl` specific configurations.
- Each portal instance (companyId) may have one.

`LDAPServerConfiguration`

- Configurations for individual ldap server configurations.
- Each portal instance (companyId) may have multiple identified by a `ldapServerId` .

`SystemLDAPConfiguration`

- General LDAP connection properties like range sizes, etc.
- Each portal instance (companyId) may have one.

There are some configurations used by various LDAP components which are configured in portal.properties, specifically:

`DefaultLDAPToPortalConverter`

- `user.screen.name.always.autogenerate`
- `user.email.address.required`

`DefaultPortalLDAP`

- `company.security.auth.type`
- `user.screen.name.always.autogenerate`

- `auth.pipeline.enable.liferay.check`

`LDAPUserImporterImpl`

- `company.security.auth.type`

These properties may be converted to using *ConfigurationAdmin* in the future.

LDAPAuthConfiguration

Link to Javadoc for LDAPAuthConfiguration

Notes for specific properties:

`method` (“bind” or “password-compare”)

- Set either "bind" or "password-compare" for the LDAP authentication method. Bind is preferred by most vendors so that you don't have to worry about encryption strategies.

`passwordEncryptionAlgorithm`

- Set the password encryption to use for comparing passwords during import and to use for encrypting passwords during export. Comparing passwords during import will only be used when the property "method" is set to password-compare. If the encryption is set to NONE, which is the default value, passwords are considered as plain text. The SHA-512 algorithm is currently unsupported.

`passwordPolicyEnabled`

- Set this to true to use LDAP's password policy instead of the portal password policy. If set to true, it is possible that portal generated passwords will not match the LDAP policy. See the "passwords.regexptoolkit.*" properties in portal,properties for details on configuring `RegExpToolkit` in generating these passwords.

LDAPExportConfiguration

Link to Javadoc for LDAPExportConfiguration

> Note: None of the values for LDAPExportConfiguration will be used unless LDAPAuthConfiguration.enabled has been set to true

LDAPImportConfiguration

Link to Javadoc for LDAPImportConfiguration

LDAPServerConfiguration

Link to Javadoc for LDAPServerConfiguration

Notes for specific properties:

`authSearchFilter`

- Active Directory stores information about the user account as a series of bit fields in the `UserAccountControl` attribute. If you want to prevent disabled accounts from logging into the portal you need to use a search filter similar to the following: (&(objectclass=person)(userprincipalname=@email_address@)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))

See the following links:

- How to use the UserAccountControl flags to manipulate user account properties
- http://support.microsoft.com/?kbid=269181

SystemLDAPConfiguration

Link to Javadoc for SystemLDAPConfiguration

Notes for specific properties:

`pageSize`

- Set the page size for directory servers that support paging. This value needs to be 1000 or less for the Microsoft Active Directory Server.

`rangeSize`

- Set the number of values to return in each query to a multivalued attribute for directory servers that support range retrieval. The range size must be 1000 or less for Windows 2000 and 1500 or less for Windows Server 2003.

`connectionProperties`

- To enable LDAP connection pooling, you have to set " `com.sun.jndi.ldap.connect.pool` " to true and provide additional JVM system properties via the JVM start-up options via:
  - java ... `-Dcom.sun.jndi.ldap.connect.pool.maxsize=50 -Dcom.sun.jndi.ldap.connect.pool.timeout=10000`

- JNDI/LDAP Service Provider

Upgrades

6.x to 7.0

While there have been many structural changes from 6.x to 7.0, there is very little you must do to upgrade. The primary upgrade task is migrating configurations from values stored in portal.properties and `PrefProps` (6.x) to `ConfigurationAdmin` (7.0). To automate this process, we provide the `LDAPPropertiesVerifyProcess`.

`LDAPPropertiesVerifyProcess` triggers as part of the portal's `VerifyProcess` and automatically converts from the legacy LDAP configurations to the new *`ConfigurationAdmin`* based configurations for LDAP. This includes upgrading:

1. `LDAPAuthConfiguration` for each portal instance
2. `LDAPExportConfiguration` for each portal instance
3. `LDAPImportConfiguration` for each portal instance
4. `LDAPServerConfiguration` for each portal instance and configured LDAP server
5. `SystemLDAPConfiguration` for each portal instance

Breaking Change

The one property that cannot be automatically migrated is: `ldap.attrs.transformer.impl`. This property allowed users to specify their own class that implements `com.liferay.portal.security.ldap.AttributesTransformer` to transform the LDAP attributes before a user or group is imported to the LDAP store.

With the OSGi infrastructure in 7.0, there's no need set this property. Instead, you would simply publish your implementation of the AttributeTransformer to the registry.

For instance, if you were using Declarative Services:

```
@Component(immediate = true, service = AttributesTransformer.class)
public class MyAttributesTransformer implements AttributesTransformer {
 @Override
 public Attributes transformGroup(Attributes attributes) {
  ....
```

```
    }

    @Override

    public Attributes transformUser(Attributes attributes) {

       …..

       return attributes;

    }

  }
```

Production Considerations

The primary production consideration for the LDAP features is promoting configurations to production. Prior to 7.0, customers that used portal.properties do not have a way to manage configurations using the Portal Settings user interface. With 7.0, this process is no longer an issue.

Customers may place configuration values in cfg files for deployment to specific environments. The deployed values may be modified in the Portal Settings user interface or via the System Settings UI.

To configure using files:

1. Create a configuration file named `com.liferay.portal.ldap.authenticator.configuration.LDAPAuthConfiguration-<companyId>.cfg`

The contents should contain the properties defined in the LDAPAuthConfiguration interface. For instance, inside of `com.liferay.portal.ldap.authenticator.configuration.LDAPAuthConfiguration-20202.cfg`

```
companyId=20202

enabled=true

method=bind

passwordEncryptionAlgorithm=NONE
```

```
required=false
```

2. Deploy this file to `$liferay_home/osgi/configs`

This approach allows for configuring LDAP authentication without using the Control Panel.

Other LDAP related configuration files should follow the appropriate naming conventions:

- `com.liferay.portal.ldap.exportimport.configuration.LDAPExportConfiguration-<companyId>.cfg`
- `com.liferay.portal.ldap.exportimport.configuration.LDAPImportConfiguration<companyId>.cfg`
- `com.liferay.portal.ldap.exportimport.configuration.LDAPServerConfiguration-<companyId>-<ldapServerId>.cfg`
- `com.liferay.portal.ldap.exportimport.configuration.SystemLDAPConfiguration<companyId>.cfg`

Additional Information

Related Links:

- Authentication Verifiers
- Configuring Liferay's LDAP Settings

¿Fue útil este artículo?

Sí No

Usuarios a los que les pareció útil: 0 de 0

¿Necesitas más ayuda?

Abre un ticket

Pregunta en la Comunidad

Liferay University
Forma a tu equipo en los puntos esenciales del uso de Liferay con videos detallados y ejercicios prácticos diseñados para simular una formación presencial

Liferay Global Services
Aumenta tu conocimiento sobre el producto y el desarrollo gracias a expertos que te ayudarán a maximizar tu ROI

© 2023 LIFERAY INC. ALL RIGHTS RESERVED