Custom Directory for EBX5

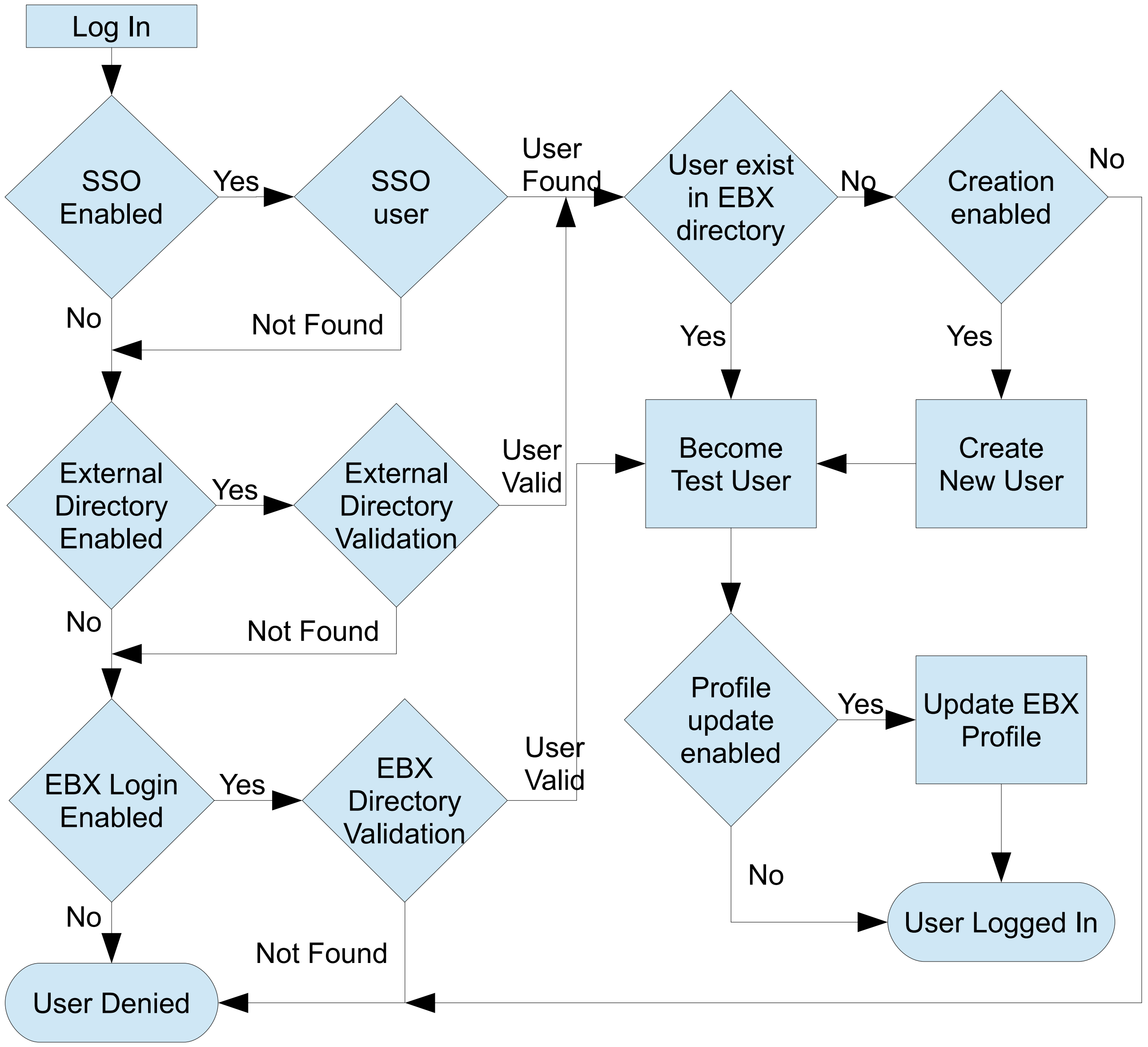
Table of Contents

Overview

The CustomDirectory framework for EBX allows for the default EBX directory to be replaced or augmented with an external means of authenticating users, determining their roles and populating their profiles. The CustomDirectory can be adapted for a a variety of external directory services.

Login Process

Login can be triggered by an HTTP request, by a login name and password, or by a SOAP data service request. Logging into to EBX involves determining who the user is and potentially adding or updating their profile in the EBX directory. These steps – Identification, Authenication, and Update can all be configured in the CustomDirectory.



Login Process Flowchart

Identification

In the identification process we determine if the identity of the user is known. This either a system provided value, part of an HTTP request, a parameter entered into EBX or the URL accessing EBX or part of the authentication header in a SOAP message.

Authentication

In the authentication process the user's identification is verified. If an external directory is defined it is first consulted for

Single Sign On (SSO)

In an HTTP request we may have information indicating the user is known to allow Single Sign On (SSO). Typically SSO is enabled when the user accessing the EBX service has presented a client certificate or another authentication method is enabled on the web server. The Custom Directory supports trusted user identification when the enabled in the `ebx.properties` file. If a trusted identification is found SSO will skip both the identification and authentication steps.

Most often the trusted user identity information is available from the server when the request's `getRemoteUser`() method is called. If this information is provided in another fashion a alternative SSO identification class can be provided by changing from the default.

Becoming Test User

As nearly all interaction with EBX can vary based on the user and role it is sometime necessary to take on a different user persona in order to verify or debug behavior. To do so an administrator can add a `become=<user>` to the EBX URL to access the system as the specified user without needing to change the users login credentials. Note this is only for users that are EBX administrators and not available for standard users.

Update

In the update phase information from an external directory is used to update the internal EBX directory. It is important to keep the internal directory up to date to accurately present user names and send email notifications.

Membership Process

The custom directory also allows for an external directory to determine EBX users role membership. The external directory As EBX consults role membership very frequently these results can be cached for a given time, reset at the start of each user session or the directory can be queried each time.

The specifics of the mapping between the directories varies with implementation, but typically there is special behavior for the built-in roles of Administrator and Read-Only.

Configuring the Custom Directory

Custom Directory behavior is configured with the following variables in the `ebx.properties` file.

Property Default	Description
ebx.directory.factory	Specifies the class implementing the custom directory
ebx.directory.enableSSO `false`	If true SSO will be attempted to provide identification and validation.
ebx.directory.SSOClass com.orchestranetworks.ps. customDirectory.SSOCheck	Specifies the class checking for Single Sign On, if enabled. Defaults to rely on the web server to set the remote user in the HTTP request.
ebx.directory.enableLogin `true`	If true login with credentials found only in the EBX directory are enabled.
ebx.directory.enableProfileUpdate `false`	If true the EBX directory will be updated with information from the external profile.
ebx.directory.enableUserCreation `false`	If true users validated externally or via SSO are added to the EBX directory
ebx.directory.userCreationAcct	Specifies the EBX administrator account that has permission to create and update users
ebx.directory.membershipCacheMs 0	Time to cache Role memberships after login. If 0 memberships cached until next login. If set less than 0 then memberships are not cached and will be checked each time.

Testing the Directory

To test the LDAP integration the directoryTest.jar will allow for connection, user validation and group membership to be tested with a given configuration file. To execute it

```
     java -jar directoryTest.jar [-p <properites file>] <user> <password>
[group[/desc]] ..]
```

For example, run

```
     java -jar directoryTest.jar -p myLDAP.prop joe secret dataEntry
```

to test if user joe has password "secret" and is a member of group dataEntry using properties file myLDAP.prop.

Implementing SSO Modules

Refer to the JavaDoc.

External Directory Modules

Refer to the JavaDoc.

LDAP Directory

The LDAPDirectory is an example of a Custom Directory implementation that uses an LDAP directory as the external directory. The LDAP directory performs login authentication, user profile update and role membership functions as an external directory.

LDAP Process

When connecting to the LDAP the bindDN and credential are used. Once connected a variety of queries are preformed.

User Identification: The ldap.search parameter is used to find the user in the directory.

User Validation: A connection is made to the directory as the identified user with the given credential.

Role Membership: For built in roles, the search can be specified specially (see the properties). Otherwise the role is found with the membershipRole and membershipBase. If not found an inconclusive result is returned. If the role is found the role membershipFilter is used to determine the result.

Profile Update: A list of EBX directory properties that differ from the existing record is returned to the Custom Directory to be updated.

LDAP Installation

Add ldapDirectory.jar as a shared library in your EBX environment. This may be by adding it to the application class loader, the server shared library folder or in the lib folder of the EBX ear you are deploying depending on your deployment.

LDAP Configuration

The LDAP directory is configured with the following settings in ebx.config.

Property Example	Description
ebx.directory.ldap.path ldap://ldap.testathon.net:389/	URL to directory
ebx.directory.ldap.baseDN DC=testathon,DC=net	Point in tree under which to search for users
ebx.directory.ldap.bindDN CN=stuart,OU=users,DC=testathon,DC=net	Login authentication principals If not set the connection will be tried anonymously
ebx.directory.ldap.credential stuart	See above
ebx.directory.ldap.search (sAMAccountName={0})	Profile DN search string, {0} is EBX userID

ebx.directory.ldap.membershipBase OU=groups,DC=testathon,DC=net	Group membership base Same as baseDN if not given
ebx.directory.ldap.membershipRole (cn={2})	Group role mapping string {0} will be replaced with EBX userID, {1} with user LDAP DN, {2} with roleID, {3} role description
ebx.directory.ldap.membershipFilter (&({2})(member={1}))	Group membership filter {0} is EBX userID, {1} is user LDAP DN, {2} is group DN
ebx.directory.ldap.adminGroup cn=cyclists	EBX administration user check - special case if defined
ebx.directory.ldap.readOnlyGroup cn=walkers	EBX read-only user check - special case if defined
ebx.directory.ldap.ldapAttrib mail,sn,givenName	LDAP attributes to copy into EBX profile
ebx.directory.ldap.userPaths ./email,./lastName,./firstName	Target paths for LDAP attributes