TIBCO JasperReports® Server Authentication Cookbook

Software Release 7.5

TIBCO®

Important Information

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

TIBCO JasperReports® Server builds on TIBCO JasperReports® Library as a comprehensive family of Business Intelligence (BI) products, providing robust static and interactive reporting, report server, and data analysis capabilities. These capabilities are available as either stand-alone products, or as part of an integrated end-to-end BI suite utilizing common metadata and provide shared services, such as security, a repository, and scheduling. The server exposes comprehensive public interfaces enabling seamless integration with other applications and the capability to easily add custom functionality.

This section describes functionality that can be restricted by the software license for JasperReports Server. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

The heart of the TIBCO Jaspersoft® BI Suite is the server, which provides the ability to:

- Easily create new reports based on views designed in an intuitive, web-based, drag and drop Ad Hoc Editor.
- Efficiently and securely manage many reports.
- Interact with reports, including sorting, changing formatting, entering parameters, and drilling on data.
- Schedule reports for distribution through email and storage in the repository.
- Arrange reports and web content to create appealing, data-rich Jaspersoft Dashboards that quickly convey business trends.

For users interested in multi-dimensional modeling, we offer Jaspersoft® OLAP, which runs as part of the server.

While the Ad Hoc Editor lets users create simple reports, more complex reports can be created outside of the server. You can either use Jaspersoft® Studio or manually write JRXML code to create a report that can be run in the server. We recommend that you use Jaspersoft Studio unless you have a thorough understanding of the JasperReports file structure.

You can use the following sources of information to learn about JasperReports Server:

- Our core documentation describes how to install, administer, and use JasperReports Server and Jaspersoft Studio. Core documentation is available as PDFs in the doc subdirectory of your JasperReports Server installation. You can also access PDF and HTML versions of these guides online from the Documentation section of the Jaspersoft Community website.
- Our Ultimate Guides document advanced features and configuration. They also include best practice recommendations and numerous examples. You can access PDF and HTML versions of these guides online from the Documentation section of the Jaspersoft Community website.

- Our Online Learning Portal lets you learn at your own pace, and covers topics for developers, system administrators, business users, and data integration users. The Portal is available online from the Professional Services section of our website.
- Our free samples, which are installed with JasperReports Library, Jaspersoft Studio, and JasperReports Server, are available and documented online. Please visit our GitHub repository.
- If you have a subscription to our professional support offerings, please contact our Technical Support team when you have questions or run into difficulties. They're available on the web at and through email at http://support.tibco.com and js-support@tibco.com.

JasperReports Server is a component of both a community project and commercial offerings. Each integrates the standard features such as security, scheduling, a web services interface, and much more for running and sharing reports. Commercial editions provide additional features, including Ad Hoc views and reports, advanced charts, dashboards, Domains, auditing, and a multi-organization architecture for hosting large BI deployments.

This chapter contains the following sections:

1.1 JasperReports Server Version Supported

This guide applies to the community and commercial editions of JasperReports Server 7.5, the most recent version at publication time. If you're running an earlier version of JasperReports Server, refer to the corresponding version of the Authentication Cookbook.

1.2 Spring Security

JasperReports Server currently uses Spring Security 4.2. If you are upgrading from an earlier version of JasperReports Server, you may need to migrate your configuration files. See the *JasperReports Server External Authentication Cookbook*, available from the support portal or the community website.

JasperReports Server relies on Spring Security 4.2 to provide the mechanisms that authenticate and authorize users. If you plan to make extensive customizations, we recommend that you delve more deeply into Spring Security by visiting its project pages and participating in its community. For more information, see the Spring Security website Spring Security website and refer to the documentation for Spring Security 4.2:

https://docs.spring.io/spring-security/site/docs/4.2.4.RELEASE/reference/htmlsingle/

1.3 Terminology

This guide uses the following terms in very specific ways.

- Users – JasperReports Server users are either people who access the web-based interface or applications that access the same content through web services. Authentication is slightly different for each type of access but operates on the same principles. For simplicity, this guide considers users to be people accessing the web-based interface.
- Internal database – User accounts are created by administrators and stored in a private, internal database. For example, when you install JasperReports Server with the default settings, it requires a PostgreSQL database

server where it stores the internal database tables containing user information. The internal database is independent of databases that store your operational data, although it may be on the same database server.

- Authentication – Authentication is the verification of a user's identity to allow access to JasperReports Server. By default, anonymous access is disabled, and all users must present valid credentials to log in: a user ID, a password, and in certain cases an organization ID. Authentication is more than gathering the user credentials, it's a process that ensures that every page is secure — either displayed to a verified user or denied until valid credentials are provided.
- External authentication – External authentication is the process of gathering and verifying user credentials through a third-party application, for example, a corporate LDAP directory. The external application is called an authority because it's trusted to contain valid user information. The process of external authentication depends on the nature of the external authority, and the various configurations support different use scenarios. For example, with simple external authentication, users log into the JasperReports Server page, but their credentials are verified externally; in a single sign-on configuration, the user may log into another application, then navigate to JasperReports Server without seeing the server's login page.
- Principal object – Authentication happens only once at the beginning of the user's session. After authentication, the user's session is represented by an in-memory instance referred to as the principal object. The existence of the principal object determines that the user is logged on and can access pages throughout the application. The principal object also stores the user's roles and organization ID, which are required for authorization within JasperReports Server.
- Authorization – Authorization is the verification of a user's roles and organization ID to access features of the server, resources in the repository, and, in some cases, data. For every page the user requests, the server determines which menu items, resources, and report contents the user can access, based on the principal object. Authorization happens every time a user accesses a resource.

  In the JasperReports Server architecture, which is based on the Spring Framework and Spring Security, authentication may be configured through an external authority, but authorization is always performed by internal mechanisms. Part of configuring external authentication is to define a mapping of external roles and organization IDs into the principal object so authorization can proceed internally. Profile attributes can also be mapped from the external authority; however, you need to write a custom processor to do so. See **"Creating a Custom Processor" on page 102**.
- Synchronization – When an external session is established, the user's current organization and roles are mapped into the principal object. The first time an external user logs in, the synchronization mechanism creates the user's organization folder, roles and user account in the internal database. The server uses these structures to enforce authorization for the external user just as for internally-defined users. The synchronization mechanism updates the roles every time the external user logs in, so that the internal database reflects the contents of the external authority.

CHAPTER 2 AUTHENTICATION IN JASPERREPORTS SERVER

This cookbook describes how to configure JasperReports Server to use external authentication in place of the built-in user authentication. The benefits of external authentication include:

- Centralized identity management within your enterprise.
- Single sign-on capabilities if the authentication mechanism supports it.

For deployments that include the Jaspersoft OLAP component within JasperReports Server, external authentication applies transparently to Jaspersoft OLAP users.

This guide covers the following authentication mechanisms:

- Lightweight Directory Access Protocol (LDAP). See **"LDAP Authentication" on page 25**.
- Central Authentication Service (CAS). See **"CAS Authentication" on page 55**.
- Authentication via an external database. See **"External Database Authentication" on page 69**.
- Authentication when the user has already been reliably authenticated by another external system. See **"Token-based Authentication" on page 85**.

You can also create custom code to run on the server after the user has been authenticated, or use custom authentication providers. See **"Advanced Topics" on page 99** for an overview of these topics. Details are beyond the scope of this guide.

The procedures in this guide assume you're familiar with JasperReports Server installation, deployment, and administration. You must have system administrator privileges within JasperReports Server and its application server and read and write access to their files on the host.

If you're setting up external authentication, you may need to understand how JasperReports Server performs internal authentication, or how external roles and organizations in JasperReports Server are created when external authorization has been set up. This chapter gives background information that can help you configure external authentication correctly.

This chapter contains the following sections:

2.1 Locating and Working With Sample Files

All Spring Security configuration files are located in the JasperReports Server web application deployed in an application server. In general, all of the sample files for external authentication are located in the <js-install>/samples/externalAuth-sample-config directory. Unless otherwise specified, all file names mentioned in this guide are located in this directory.

2.1.1 Deploying Configuration Files

To configure JasperReports Server to work with external authentication, you need to create and deploy an external configuration file as follows:

1. Create or copy a file and name it in the form applicationContext-<customName>.xml, for example, applicationContext-externalAuth-LDAP.xml. JasperReports Server includes sample files for some implementations, for example, LDAP, CAS, and an external JDBC database, in the <js-install>/samples/externalAuth-sample-config/ directory.
2. Edit the file and create and configure the bean properties correctly for your deployment, as described in the following sections.
3. Place the correctly configured applicationContext-<customName>.xml file in the <js-webapp>/WEB-INF directory.

2.1.2 WEB-INF Directory Location

Depending on your deployment and your needs, there are several ways to work with configuration files:

Deployment	<js-webapp>/WEB-INF File Location
Installed server	Once you've installed your server, either through a platform installer or any other deployment, the configuration files are deployed in the application server. The location depends on the application server where you installed JasperReports Server. If you used the bundled Apache Tomcat application server, the modified configuration files should be placed in: <js-webapp>/WEB-INF = <js-install>/apache-tomcat/webapps/jasperserver[-pro]/WEB-INF After modifying the configuration files, restart the application server to use the settings.
WAR file distribution	When you download the WAR file distribution, you can customize your deployment of JasperReports Server and possibly install it on several machines. You can find the WAR file in the following location: <js-webapp> = <js-install>/jasperserver[-pro].war After modifying the WAR file distribution, you need to redeploy it to your application server, as described in the *JasperReports Server Installation Guide*. But every time you redeploy your modified WAR file, external authentication is pre-configured.

Deployment	<js-webapp>/WEB-INF File Location
Source code	The JasperReports Server source code contains the XML source of the configuration files. If you maintain other customizations in the source code, you can modify the configuration files for external authentication. The modified configuration files for source code are placed in: <js-webapp>/WEB-INF = <js-src>/jasperserver/jasperserver-war/src/main/webapp/WEB-INF When building the source, these files are copied into the WAR file that you must then deploy into a running application server. See the *JasperReports Server Source Build Guide* for more information.

When working with the WAR file distribution or source code, you usually modify the files in an installed server for testing. But after testing, you copy the changes into your WAR file or source code.

When working with the WAR file distribution or servers installed in application servers other than Apache Tomcat, the WAR file is kept as a single archive file from which you must extract, modify and replace the files. The following code sample shows one way to do this from the command line.

```
cd <js-webapp>
"%JAVA_HOME%\bin\jar" xf jasperserver[-pro].war <path/filename>
<edit> <path\filename>
"%JAVA_HOME%\bin\jar" uf jasperserver[-pro].war <path/filename>
delete <path\filename>
```

In this sample:

- <path/filename> refers to the relative path and name of the file to modify within the WAR file
- -pro is part of the WAR file name if you installed a commercial edition of JasperReports Server.

2.2 Configuring Logging for Debugging

If your connection is failing, for example, with an “Invalid credentials supplied” error, and you cannot find information in the JasperReports Server logs, you may want to enable logging for Spring Security or JasperReports Server external authentication.

To enable logging, add the corresponding line to the <js-webapp>/WEB-INF/log4j.properties file, in the form:

```
log4j.logger.<logger-classname> = <log-level>, <output-type>
```

For example:

```
log4j.logger.org.springframework.security=DEBUG, stdout, fileout
```

You must restart the server for your changes to take effect. For more information about log configuration, see the *JasperReports Server Administrator Guide*.

You can reduce impact on performance by restricting logging to functionality that you are interested in, for example, to a subset of Spring Security instead of all of Spring Security. **Table 2-1** shows some logger classnames commonly used when debugging authentication.

Table 2-1 Useful Logger Classnames for External Authentication

Functionality to Log	Logger Classname
Spring Security	`org.springframework.security`
Subset of Spring Security related to LDAP	`org.springframework.security.ldap`

Useful subsets of Spring Security's LDAP logging	`org.springframework.security.ldap.userdetails.LdapAuthoritiesPopulator`
	`org.springframework.security.ldap.userdetails.LdapUserDetailsService`
	`org.springframework.security.ldap.search.LdapUserSearch`
	`org.springframework.security.ldap.search.FilterBasedLdapUserSearch`
	`org.springframework.security.ldap.SpringSecurityLdapTemplate`

Other LDAP-related Spring Security loggers	`org.springframework.security.providers.ldap.LdapAuthenticationProvider`
	`org.springframework.security.userdetails.ldap.LdapUserDetailsMapper`
	`org.springframework.security.providers.ldap.authenticator.BindAuthenticator`

Subset of Spring related to LDAP	`org.springframework.ldap`
CAS	`org.jasig.cas`
JasperReports Server external authentication API	`com.jaspersoft.jasperserver.multipleTenancy.security.externalAuth`
Subset of external authentication API related to single tenancy	`com.jaspersoft.jasperserver.api.security.externalAuth`
Subset of external authentication API related to token-based authentication	`com.jaspersoft.jasperserver.api.security.externalAuth.preauth`
Subset of external authentication API related to external database authentication	`com.jaspersoft.jasperserver.api.security.externalAuth`

Enabling a logger and a sublogger together creates duplicate entries in the logs. For example, `org.springframework.security` and `org.springframework.security.ldap` creates duplicate entries.

2.3 Default Internal Authentication

The following diagram shows the general steps involved in JasperReports Server's default internal authentication:

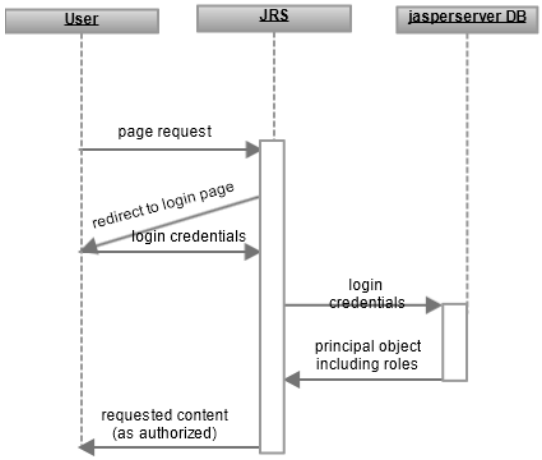

Figure 2-1 Steps of Internal Authentication

The interaction between the user's browser and JasperReports Server includes these general steps:

1. An unauthenticated user requests any page in JasperReports Server.

   Often, users bookmark the login page and begin directly at **step 3**, but this step covers the general case and secures every possible access to the server. For example, this step applies when a user clicks the page of an expired session or if a user enters the direct URL to a report in the repository.
2. JasperReports Server detects that the user is not logged in and replies with a redirect to the login page.

   For convenience, the server includes the original URL in the login screen request so that the user goes directly to the requested page after logging in.
3. The user enters a username, password, and possibly an organization ID.

   JasperReports Server compares these credentials with the existing user accounts in the internal user database, and if they are valid, creates a principal object. The user is now authenticated, and the principal object represents the user session, including any roles found in the user database.
4. JasperReports Server sends the requested content to the user, or if none was specified, the home page.

   Content that is sent to the user is subject to authorization. For example the home page has different options for administrators than for end-users, as determined by the roles of the user in the principal object. If the user is viewing the repository, the folders and objects returned depend on the organization ID and roles in the principal object.

2.4 Organizations and Users in JasperReports Server

When performing external authentication, JasperReports Server obtains all the information about a user from the external authority. In order to create and enforce permissions, JasperReports Server must store the information about users, roles, and organizations in the internal database. So for each externally-defined user, role, and

organization, the server creates a local user account, role definitions, and organization folders. This process is called synchronization.

The users and roles defined through external authentication appear in the management interface, but are labeled as "external." Administrators can view and delete the external users and roles, but the synchronization will re-create the necessary users and roles as long as external authentication is configured. External user accounts and roles are placeholders for use by the synchronization and permissions mechanisms. Administrators can disable specific external users in JasperReports Server to prevent those external users from logging into JasperReports Server.

There are three important aspects to managing external users, roles, and organizations:

- The synchronization of external users, roles, and organizations with the internal database is automatic once external authentication is configured. This is done by the `ExternalDataSynchronizer` bean. For more information, see **"Advanced Topics" on page 99**.
- Permissions in the repository must be initialized manually for the external roles after their creation.
- Maintenance of the external users is necessary only when creating or deleting roles, when creating new organizations from an external authority, or when disabling external users in JasperReports Server.

When you deploy JasperReports Server in a production environment, you need to set up role permissions before your users access the server. However, you cannot create external roles or organizations directly; you can create them only by logging in as an external user with the desired roles and permissions. To set up role permissions, you must understand the synchronization process. For maintenance, you must be aware of how changes in the external authority impact permissions in JasperReports Server. These processes are explained in the following sections.

2.4.1 Multiple Organizations in JasperReports Server

This section describes functionality that can be restricted by the software license for JasperReports Server. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

Some commercial editions allow several distinct organizations to coexist within the same server instance. Each organization, also known as a tenant, has its own users, roles, and possibly a hierarchy of sub-organizations, each of which is invisible to other organizations. For information about deploying multiple organizations, see the *JasperReports Server Administrator Guide*.

When you deploy JasperReports Server, there are three distinct cases with respect to the organization architecture:

Edition and Usage	Impact on External Authentication Configuration
Commercial edition with multiple organizations	Configuring external authentication for multiple organizations requires extra steps. In the chapter for each authentication mechanism, look for the additional section on mapping the organization. All other configurations for external authentication remain the same.

Edition and Usage	Impact on External Authentication Configuration
Commercial edition with no organizations	When JasperReports Server has the organization architecture, but only implements a single default organization, the organization ID is mapped automatically. You can skip any section that refers to mapping the organization ID.
Community Project	JasperReports Server Community Project does not use the organization architecture. You can skip any section that refers to mapping the organization ID.

Another name for multiple organizations is multi-tenancy, sometimes abbreviated `mt` in file and bean names. However, the `mt` prefix appears in both community and commercial editions.

2.4.1.1 Default Admins in Organizations

You can set one or more default admins that are created when you create a new organization using external authentication. The sample files are set up to create a `jasperadmin` in each new organization by default. For information on how to customize default admin users, see the section on organizations in the chapter corresponding to your authentication mechanism.

Default admins for new organizations can be customized only when using external authentication. When you create organizations manually, the `jasperadmin` user is created.

2.4.2 Synchronization of External Users

When a user is authenticated by an external authority, JasperReports Server initializes its session principal object, which contains the username, role names, and organization ID, if applicable. The `ExternalDataSynchronizer` uses this information to automatically update or create corresponding structures in the internal database:

- If the internal database has no organization with the user's organization ID, JasperReports Server creates it with the templates currently defined in the repository. For LDAP authentication, organization hierarchies are created for users.
- Roles can be mapped to external or internal roles.
  - When mapping a role to an external role, the role name is compared to existing external roles. If the role doesn't exist, it's created as an external role. See **2.4.3, "Synchronization of Roles," on page 18** for details about role creation.
  - When mapping a role to an internal role, the role name is compared to existing internal roles. If the role doesn't exist, it's created as an internal role. You can also create a role at the root level, which gives administrative permissions, or at the organization level, which restricts access to the organization. See **2.4.3, "Synchronization of Roles," on page 18** for details about role creation.
- The user ID is compared to existing user accounts in the internal database. If an organization ID is specified, only the user IDs in that organization are checked.
  - If the user ID matches an account in the internal database, its list of assigned roles is synchronized as described in **2.4.3, "Synchronization of Roles," on page 18**. The user ID can match either a previously synchronized external user or an internal user created by an administrator. If the external user ID matches an existing internal user, authentication fails; an administrative user has to resolve the situation manually.

- If the user ID does not match an account in the internal database, an external user account is created. If an organization ID is specified, the account is created within that organization. Finally, all of the external roles along with any configurable default internal roles are assigned to the new user account.

For more information about organizations, roles, and user accounts see the *JasperReports Server Administrator Guide*.

A user account created for an external user has the same structure as an internal user account but differs in the following ways:

- A database flag marks it as externally defined.
- The full name of the user is the same as the user ID, which is always the same as the login name entered by the user.
- The external user account does not store the password.
- It does not have any values for optional user properties, such as the user's email or profile attributes. The default implementation of external authentication does not include these properties. An administrator can manually include these properties.

An external authority such as LDAP contains information like the user's full name, email address, and profile attributes, which can be mapped into the external user account. However, this requires customizing the mapping and synchronization beans. See **Chapter 7, "Advanced Topics," on page 99**.

After synchronization, the external user fits in cohesively with all the structures and mechanisms of JasperReports Server, especially those required for authorization. But the JasperReports Server administrator's management of an external account is limited to the ability to disable the account and prevent the external user from logging in.

An external user cannot log in when the external authority is offline; external accounts do not store the password and are not meant for failover. Once external authentication is configured, only the information in the external authority determines who can log in and what roles they have. However, administrators may view external organizations, users, and roles to determine if all mappings from the external authority are correct.

2.4.3 Synchronization of Roles

Role synchronization is a complex process because roles need to be updated every time an external user logs in. Like users and organizations, roles are synchronized in two phases: mapping the roles from the external authority to roles in JasperReports Server, then assigning them to the user.

External authentication in JasperReports Server involves three role types:

- External — The synchronization process assigns and removes external roles in users' accounts according to the roles defined in the external authority. These roles are flagged in the UI, because they're not meant to be managed by administrators.
- Internal — JasperReports Server administrators create internal roles. To assign internal roles to external users, set up synchronization to map external roles to internal roles. Do not manually assign internal roles to external users. External role mapping takes precedence over manually-assigned roles. So synchronization may remove manually-assigned roles from users' accounts.
- System — JasperReports Server creates system roles like `ROLE_DEMO` during installation. System roles are handled exactly like internal roles above.

In the first phase of synchronization, the principal object has a set of role names derived from role definitions in the external authority. The goal is a JasperReports Server role for each of the mapped role names.

- If you're using organizations, all target roles are created within the user's mapped organization.
- If a role with the target name is already defined in JasperReports Server, that role is assigned to the user. Otherwise a new external role with this name is created and assigned.
- By default, roles are mapped to external roles, even if an internal role has the same name. If an external role name conflicts with an existing internal role in the target organization, a suffix like `_EXT` is added to the role name.
- You must explicitly map external roles to internal roles. If you're mapping a role to an internal role, you can specify whether to assign the internal role at the organization level or at the system (root) level. Roles mapped at the organization level have no administrative privileges. Roles at the system administrative privileges and access to the repositories of all organizations. To map to an internal role at the organization level, append |* to the name of the internal role. To map to an internal role at the system level, do not modify the internal role name.

The goal of the second phase of synchronization is to update the user with the set of roles mapped from the external authority. The origin of a role determines how the synchronizer assigns and removes the role from an external user account:

- All of roles identified or created in the first phase — internal, external, and system — are assigned to the external user. Any role not yet in the user's account, is assigned by the synchronization process.
- External roles that are assigned to the user — but not among those mapped and identified in the first phase — are removed from the user. This way, roles removed from the external authority are also removed from the user's JasperReports Server account.
- Internal and system roles created by synchronization during a previous login — but no longer mapped and identified from the external authority — are removed from the user. If an internal role that was assigned or removed by an administrator appears in the mapping configuration, the mapping from the external role takes precedence. This means that a manually assigned role may be added or removed based on the state of the external database. Therefore, to ensure that synchronization automatically makes all roles reflect those in the external authority, you should not manually assign internal roles.

2.4.4 Initialization of JasperReports Server for External Users

Knowing how external users are synchronized, the system administrator must initialize JasperReports Server and set the permissions in the repository to provide needed authorization.

Your deployment procedure must include the following steps:

1. Configure the mapping of usernames, roles, and possibly organization IDs from your external authority to JasperReports Server. The mapping depends on external authentication, as described in the chapter for each process. Optionally configure an admin user to be created in each external organization. The sample configuration files create a `jasperadmin` user by default.
2. Create test users in the external authority for every organization and with every role that you expect in your production environment.
3. If you're mapping external users to multiple organizations, log into JasperReports Server as the system administrator and prepare your organization templates and, possibly, themes for those organizations.
4. Log into JasperReports Server as each of the test users. Doing so validates your configuration of the external authentication and mapping beans. When successful, it also creates the external roles and organizations you need.

5. Log into JasperReports Server as the system administrator and:
    a. Ensure that mapping and synchronization created the external users, roles, and organizations you expect.
    b. In every organization, change the password of each automatically created administrator.
6. Initialize your repository:
    a. If you're using organizations, create additional repository resources, like data sources and shared reports, within each organization folder.
    b. Define all your repository permissions using the external roles that were created.

This procedure is necessary because the external roles must exist in the internal database before you can create permissions for them. If you're using organizations, roles must be defined within organizations, so the organizations must exist as well. Optionally, you can use the administrative pages to create all the organizations your externally authenticated users need. This allows you some additional control over the creation of the organizations, but you must ensure that their IDs exactly match the values and hierarchies determined by the mapping.

When your JasperReports Server is in production, the external user accounts will be populated by the synchronization process as users log in. When the mapping is correct and consistent, the user population will have the same roles and organizations as those in your external authority, and you won't need to manually import users or roles into the server. And as role membership is updated in the external authority, external users are automatically synchronized in JasperReports Server.

2.4.5 Maintenance of External Users

The advantage of an external authority is that it provides a single place to manage user accounts across applications. So JasperReports Server can use the same login information you maintain for your other applications. When a user logs in, the server receives the username then maps the roles and organization. If these have changed from the previous login, the server can synchronize the locally stored external user information.

However, external authentication does not actively replicate the contents of the external authority in JasperReports Server. If users are deleted from the external authority, or role definitions change, JasperReports Server's local information is not updated with these changes. The external user or role definitions remain in the server's internal database, but authentication is still secure because they can't be used to log in.

In general, the mapping and synchronization maintain the external users and roles in JasperReports Server, but administrators must handle new roles, and you may want to cleanup deleted users, as explained in the following sections.

2.4.5.1 Managing External Users

The following table describes the impact on JasperReports Server when managing users in the external authority:

Action in External Authority	Impact on JasperReports Server
Creating a new user	JasperReports Server automatically creates the new external user account when the user first accesses the server. As long as the user relies on existing roles in an existing organization, no server changes are required. If the user is associated with new roles, see **2.4.5.2, "Managing External Role Definitions," on page 21**.
Updating a password	JasperReports Server doesn't store the passwords of external users, and is not affected by changes to user passwords or policies on the external authority.
Updating personal information	With the default mapping of external users, only the login name is stored in an external user account. If you configure role mapping based on personal information in the external user entry, you need to account for any possible change in the content or structure of the external authority. For example, if a role is based on a user's `department` attribute, make sure the user can't modify this attribute in the external authority. Otherwise, the user could inadvertently or maliciously change his or her roles within JasperReports Server.
Changing group or role membership	Described in **2.4.5.2, "Managing External Role Definitions," on page 21**.
Changing organization membership	When organizations represent departments within a company, a user may change organizations, as mapped from the external authority. From JasperReports Server's point of view, this is the same as deleting a user in the old organization and creating a user in another organization.
Disabling or deleting a user	When the user can no longer authenticate with the external authority, he can't access JasperReports Server. Even though the external user account remains in the internal database, it can't be used for logging in. The defunct user account has no impact on the server; you can safely delete it. An external user can be disabled in JasperReports Server.

2.4.5.2 Managing External Role Definitions

Roles can be mapped from a variety of structures that depend on the external authority: LDAP authentication maps roles dynamically from groups, and CAS authentication extracts roles from an external data source or specifies them statically in the configuration file. Because each external authority may define roles differently, this guide refers to those structures collectively as role definitions.

In practice, you'll find that only a subset of the role definitions in your external authority are applicable to JasperReports Server. Of those, some may also be used by other applications, and others may be created specifically for managing users in JasperReports Server. You should identify the maintenance procedures on your enterprise-wide user authority that impact JasperReports Server and document additional procedures for keeping JasperReports Server in sync.

The following table describes the impact on JasperReports Server when modifying role definitions in the external authority:

Action in External Authority	Impact on JasperReports Server
Creating a new role	Role definitions are not directly mapped to JasperReports Server; only roles that are assigned to users who log in are mapped. When you create a new role and assign it to a user who accesses JasperReports Server, determine which case applies: • The role is significant to access control within JasperReports Server. You must initialize this role in the server with a test user and define all necessary repository authorization rules to secure your data before you deploy this role to real users, as described in **2.4.3, "Synchronization of Roles," on page 18**. • The role is not significant to users within JasperReports Server. Synchronization automatically creates the role and assigns it to users according to their mapping, but with no authorization rules based on the role, it has no impact.
Modifying role membership	Changes in role membership are reflected the next time a role member starts a new session in JasperReports Server, as described in **2.4.2, "Synchronization of External Users," on page 17**. Roles that were previously unknown to the server are treated as new roles as described above, and roles that are no longer assigned to a user are deleted as described below.
Deleting a role	External users no longer have the role, and it is removed from each external user by synchronization upon the next login. The role remains in the internal database, and permissions that reference the role remain in the repository. The role may still be assigned to external users who have not logged in since the role was removed. • If the role definition in the external authority was mapped to an external role in JasperReports Server, it has no impact on the server and you can safely delete it. • If the role definition is mapped to an internally defined role in JasperReports Server, you can delete the role or modify the configuration file to remove the mapping. If you remove the mapping, the internal role can be assigned manually by an administrator. If you do not modify the configuration file and you attempt to assign the internal role manually to a user in JasperReports Server, the role is automatically removed during synchronization.

2.4.5.3 Modifying Role Mappings

Once you've set up external authentication with your JasperReports Server instance, you add new role mappings by editing the applicationContext-externalAuth-*.xml file. You need to restart the server for these changes to take effect.

Be careful changing or removing role mappings. When a role mapping is removed or changed, synchronization no longer updates the target role in JasperReports Server. This means users assigned a deleted external role still have that role in JasperReports Server. You can work around this by creating a mapping from a non-existing role definition in the external authority to the target role you want to remove.

When you want to change the target role for an existing role mapping, you should create a dummy mapping that maps a non-existent role definition to the JasperReports Server role you no longer want to use.

For example, suppose you have **Sales Manager** as a role in your external authority, and you initially map it to ROLE_ADMINISTRATOR in JasperReports Server.

Sales Manager Mandy logs into JasperReports Server and is assigned ROLE_ADMINISTRATOR.

You then create a new role in JasperReports Server, ROLE_SALES_MANAGER, and modify your role mapping so **Sales Manager** in the external authority is now mapped to ROLE_SALES_MANAGER in JasperReports Server. You then restart the server.

By default, the next time Mandy logs in, she's assigned ROLE_SALES_MANAGER. But because ROLE_ADMINISTRATOR no longer appears in your application context file, synchronization doesn't check for it and remove it. Mandy now has two roles: ROLE_ADMINISTRATOR and ROLE_SALES_MANAGER.

You can remove ROLE_ADMINISTRATOR from Mandy's account by creating a dummy mapping with ROLE_ADMINISTRATOR as the target. For example, if no one in your external authority has the role definition **No Such Role**, you can add a mapping in your application context file from **No Such Role** to ROLE_ADMINISTRATOR then restart the server. The next time Mandy logs in, the synchronizer finds that she doesn't have the **No Such Role** role definition and removes ROLE_ADMINISTRATOR.

It is possible for a role in JasperReports Server to be the target of more than one role mapping. If multiple role definitions map to the same role in JasperReports Server, users who have any one of the role definitions will receive the role in JasperReports Server.

2.4.5.4 Managing External Organizations

The following table describes the impact on JasperReports Server when modifying organizations defined in the external authority:

Action in External Authority	Impact on JasperReports Server
Adding an organization	Organizations are not directly mapped to JasperReports Server, rather a new organization ID is mapped when synchronizing the first user in the organization that accesses the server. At that time, synchronization creates the organization and the user within it, along with any roles assigned to the user. Determine which of the following cases applies: • Your organization definitions and mappings create the same role names in every organization. You should configure the organization folder templates so the default contents and permissions work with the known role names. Your external organization definitions should then map to organizations that work as soon as the first user logs in. • Each of your externally defined organizations has different role names or requires specific repository contents. You should create test users in the new organizations first, so you can configure the new organization folder, synchronize external roles, and assign repository permissions before actual users have access. See the procedure in **"Initialization of JasperReports Server for External Users" on page 19**.

Action in External Authority	Impact on JasperReports Server
Modifying an organization	Changing the users or roles in organizations defined in the external authority is the same as adding users or roles to one organization and removing them from the other. See the corresponding actions in **2.4.5.1, "Managing External Users," on page 21** and **2.4.5.2, "Managing External Role Definitions," on page 21**.
Deleting an organization	Because organization definitions are not mapped directly, deleting an organization has the same effect as removing each of its users. The organization remains in the internal database and repository, along with the external roles and users who last accessed it. The unused organization has no impact on the server. You can safely delete it.
Changing the default admin users of organizations	Default admin users are created only when the organization is created. Therefore, changes to the default admin users appear only in organizations created after the changes were made. In particular, if you add or delete default admin users, your changes affect only new organizations.

2.4.6 Internal Users

Even when you enable external authentication, JasperReports Server supports internal users, especially administrative users. For example, you can still define `superuser` and `jasperadmin` internally. Internal users cannot login through a separate external login screen. They can log in only through the JasperReports Server login screen, for example at http://localhost:8080/jasperserver-pro/login.html. Internal administrators such as `superuser` may have access at the root organization level. They can also set internal permissions for external users, if necessary.

An external user can have the same username as an internal user in a different organization. But if this happens within an organization, the external user won't be able to log into JasperReports Server.

CHAPTER 3 LDAP AUTHENTICATION

Lightweight Directory Access Protocol (LDAP) is one of the most popular architectures for enterprise directories. By centralizing all user management in an LDAP directory, applications across the enterprise can share the same user database, and administrators don't need to duplicate user accounts.

This chapter shows how JasperReports Server can be configured to perform external authentication with LDAP. As part of the authentication process, JasperReports Server also synchronizes the external user information, such as roles and organization ID, between LDAP and the JasperReports Server internal database.

LDAP authentication does not provide single sign-on (SSO) functionality. You must implement that separately and configure it for use within JasperReports Server. Enabling SSO with LDAP is beyond the scope of this guide. For more information, see **Chapter 7, "Advanced Topics," on page 99**.

This chapter assumes you're familiar with LDAP servers and the structure of the data they contain, in particular the format of distinguished names (DNs) and relative distinguished names (RDNs) that create structure and identify entries in LDAP. For more information about LDAP in Spring Security, see the LDAP sample in the Spring Security reference documentation for 3.2.x at http://docs.spring.io/spring-security/site/docs/3.2.5.RELEASE/reference/htmlsingle/.

This chapter contains the following sections:

3.1 Overview of External LDAP Authentication

This section explains how JasperReports Server performs external authentication with an LDAP server, highlighting the differences with **"Default Internal Authentication" on page 15**.

The following diagram shows the general steps involved in external LDAP authentication:

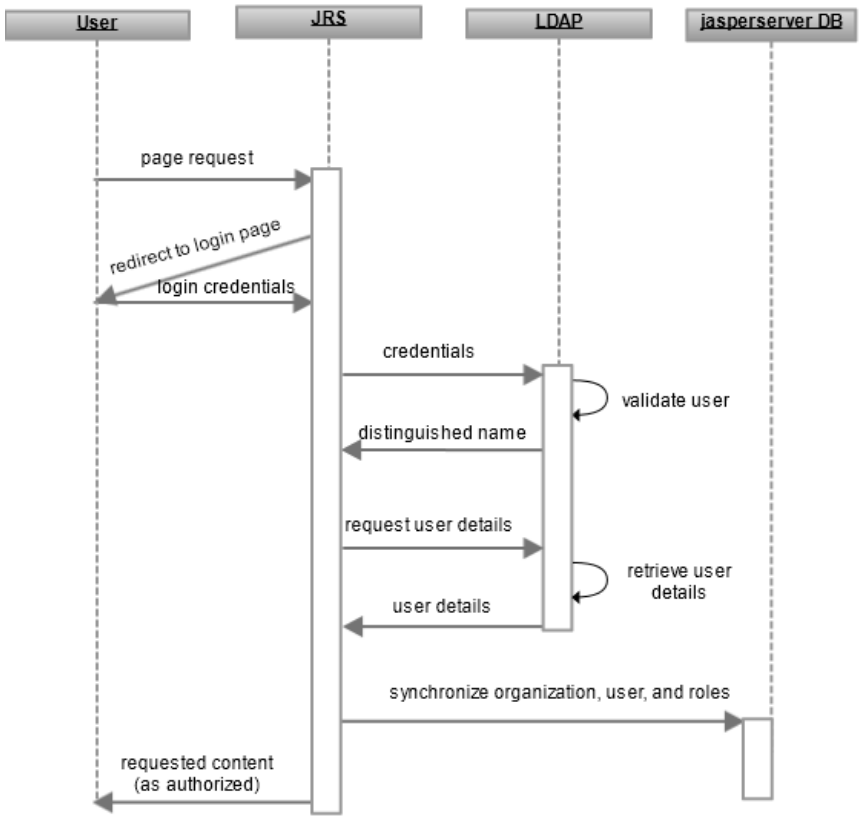

Figure 3-1 General Steps of External LDAP Authentication

The following process explains the interaction of the user's browser, JasperReports Server, and the LDAP server:

1. An unauthenticated user requests any page in JasperReports Server.

   Often, users bookmark the login page and begin directly at **step 3**, but this step covers the general case and secures every possible access to the server. For example, this step applies when a user clicks the web interface of an expired session or if a user is given the direct URL to a report within the server.
2. JasperReports Server detects that the user is not logged in and redirects to the JasperReports Server login page.
3. The user submits a username and password through the login page, even though the user credentials are not verified internally.

   In servers with multiple organizations, the organization ID must be left blank because it is supplied by the external LDAP authority, except in the case of an internal login (such as an administrator), then the organization ID must be provided.
4. JasperReports Server performs a search on the LDAP server with the given credentials. If they are valid, the server creates a principal object to represent the user's session in memory. In multi-organization environments, the user's organization ID is mapped from the LDAP entry. The server also performs a second search to map the user's LDAP groups to server roles.

The beans that perform LDAP authentication do not map information like the user's full name, email address, or profile attributes that may exist in the LDAP directory. This requires customizing the `JSFilterBasedLdapUserSearch` bean, as described in **Chapter 7, "Advanced Topics," on page 99**.

The username, roles, and organization information are also synchronized with the internal database, where the user account is marked as an external user. The user is now authenticated, the principal object represents the user session, and the JasperReports Server environment reflects the user's roles and organization defined in LDAP. For more information about synchronization, see **"Synchronization of External Users" on page 17**.

5. As with the default internal authorization, JasperReports Server now sends the requested content to the user or, if none was specified, the home page appropriate for the user.

   Content sent to the user is subject to authorization. For example the home page has different options for administrators than for regular users, as determined by the roles of the user in the principal object. Or if the user is viewing the repository, the folders and objects returned are determined by the organization ID and roles in the principal object.

When comparing these steps with those in **2.3, "Default Internal Authentication," on page 15**, there are three significant differences, all in **step 3**:

- JasperReports Server verifies the credentials through LDAP instead of using its internal user database.
- The roles and organization ID in the user's principal object are mapped from the LDAP response.
- The internal database must be synchronized with any new information in the user's principal object.

3.2 Configuring JasperReports Server for LDAP Authentication

To use LDAP with your JasperReports Server, configure the LDAP connection parameters in default_master.properties before installing JasperReports Server. You can set up encryption for the password to your LDAP server at that time. See the *JasperReports Server Security Guide* for more information.

A sample file for configuring JasperReports Server for external LDAP authentication is included in the JasperReports Server distribution. Sample files are located in the <js-install>/samples/externalAuth-sample-config directory of your JasperReports Server. The file included depends on your version of JasperReports Server:

- sample-applicationContext-externalAuth-LDAP.xml: Sample file for integrating LDAP with JasperReports Server with a single organization. This file is included in the community edition.
- sample-applicationContext-externalAuth-LDAP-mt.xml: Sample file for integrating LDAP with JasperReports Server with multiple organizations. This file is included in commercial editions of JasperReports Server. To use external authentication with a commercial version of JasperReports Server with a single organization, you need to modify the sample file as described in **3.7, "Mapping the User Organization," on page 40**.

To configure JasperReports Server to work with your implementation of LDAP, modify and deploy the sample configuration file as follows:

1. Make a copy of the LDAP sample file in the <js-install>/samples/externalAuth-sample-config/ directory and rename it to remove the sample- prefix.
2. Edit the file you created and configure the beans correctly for your deployment, as described in the following sections.
3. Place the modified file in the <js-webapp>/WEB-INF directory.

<js-webapp> is the location of the JasperReports Server web application in your application server, or where you're modifying the configuration files, as explained in . The rest of this chapter refers to file names alone.

3.3 Overview of LDAP Beans

The sample-applicationContext-externalAuth-LDAP[-mt].xml file contains the beans needed to enable and perform LDAP authentication. This section summarizes the most important beans in this file, including the beans you need to modify to configure JasperReports Server to work with your external database.

- `proxyAuthenticationProcessingFilter`: Bean that enables external authentication for direct access. When this proxy bean definition is present in the application context, that is, when it appears in an applicationContext-<customName>.xml file in the <js-webapp>/WEB-INF directory, the Spring Security filter chain processes the authentication with the proxy definitions instead of the default internal filter. You do not need to configure this bean.
- `ldapAuthenticationManager`: Lists the available authentication providers. The providers in the list are invoked in the order they appear in the configuration file until one of them authenticates the user. The rest of the providers are then skipped. The final provider in the list, `${bean.daoAuthenticationProvider}` authenticates against the `jasperserver` internal database. You can customize authentication by adding more providers to this bean.
- `ldapContextSource` – Helper bean that defines the LDAP server used by the `ldapAuthenticationProvider` bean. Configure your LDAP connection using this bean, as described in **3.4, "Setting the LDAP Connection Parameters," on page 29**.
- `ldapAuthenticationProvider`: Custom authentication provider for LDAP. This bean has two inline sub-beans:
  - Bean of class `JSBindAuthenticator` – `JSBindAuthenticator` is a wrapper class for the Spring Security `BindAuthenticator` class. Configure this bean and its sub-bean `userSearch` to specify the rules for finding user entries in your LDAP directory, as described in **3.5, "Performing LDAP User Search," on page 31**.
  - Bean of class `JSDefaultLdapAuthoritiesPopulator` – `JSDefaultLdapAuthoritiesPopulator` is a wrapper class for the Spring Security `DefaultLdapAuthoritiesPopulator` class. Configure this bean to specify the location of group definitions in LDAP, how to find the groups to which the user belongs, and any transformation of group names in LDAP to role names in JasperReports Server, as described in **3.6, "Mapping the User Roles," on page 34**.
- `externalDataSynchronizer` – Bean whose class creates a mirror image of the external user in the internal `jasperserver` database. The sample includes the following processors:
  - `ldapExternalTenantProcessor` (commercial editions only) – Bean that maps externally defined tenants to JasperReports Server organizations. For single-organization JasperReports Server deployments, configure this bean to specify the default organization, as described in **3.7.2, "Mapping to a Single Organization," on page 45**. For multi-organization JasperReports Server deployments, configure this bean to specify the mapping between LDAP RDNs and JasperReports Server organizations, as described in **3.7.1, "Mapping to Multiple Organizations," on page 41**.
  - `mtExternalUserSetupProcessor` or `externalUserSetupProcessor` – Bean that creates and configures the internal user corresponding to a successfully authenticated external user. Configure this bean to specify the default internal role given to the external users in JasperReports Server and to map external LDAP roles to internal JasperReports Server roles, as described in **3.6.2, "Mapping Roles to System Roles," on page 36**.

The following figure shows the beans used in LDAP authentication:

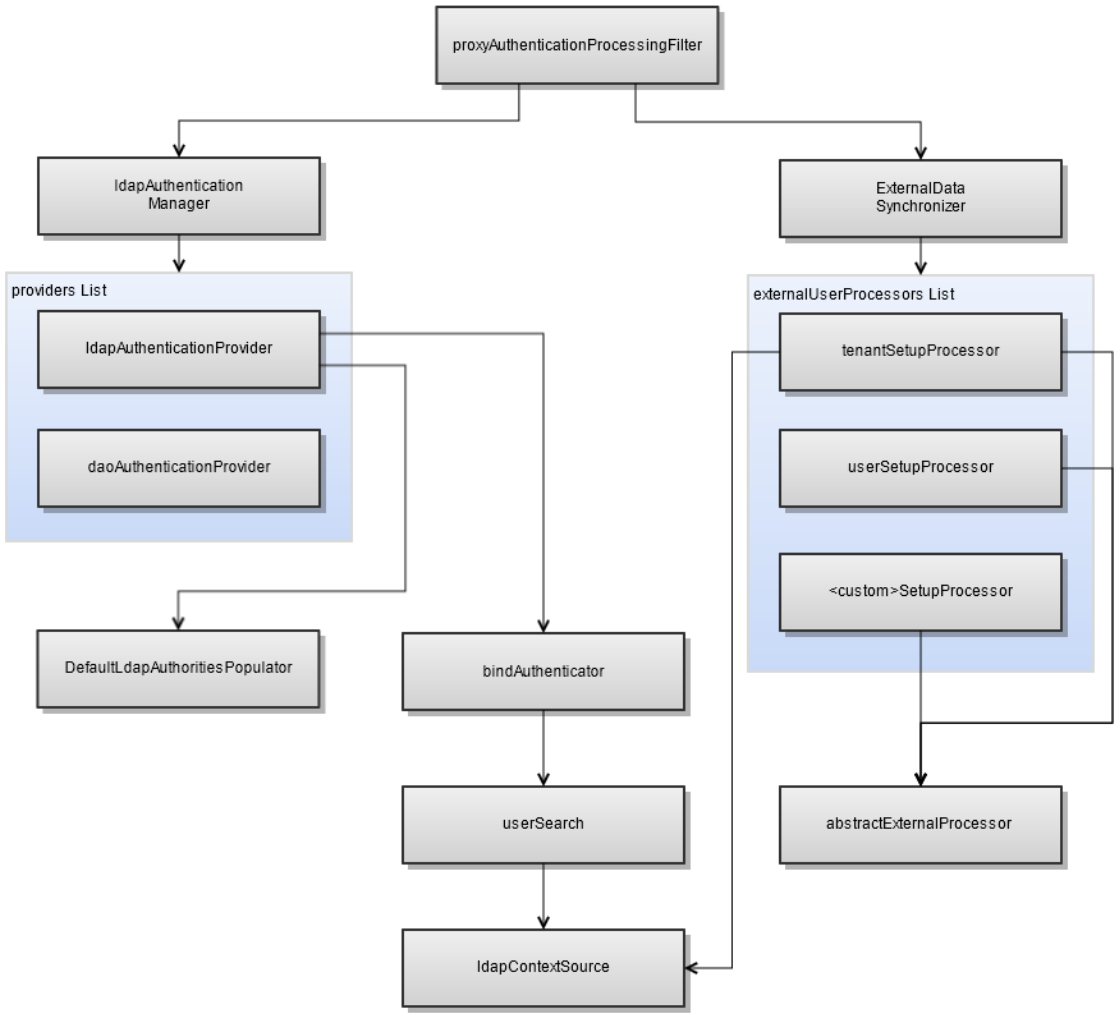

Figure 3-2 LDAP Beans

3.4 Setting the LDAP Connection Parameters

You can configure your connection to the LDAP server in one of two ways:

- Configure the connection by LDAP properties in the default_master properties file before installation or upgrade. You can choose to encrypt any of the LDAP connection parameters. This is the preferred method for setting the LDAP connection parameters.
- For an existing JasperReports Server, you can configure the connection properties directly in your sample-applicationContext-externalAuth-LDAP[-mt].xml file. In this case, the properties, including the password, cannot be encrypted.

3.4.1 Setting LDAP Connection Parameters in default_master.properties

The preferred approach is to configure the `external.ldapUrl`, `external.ldapDn`, and `external.ldapPassword` properties in the default_master.properties file before installation or upgrade. The default configuration of the `ldapContextSource` bean in sample-applicationContext-externalAuth-LDAP[-mt].xml uses context properties for the LDAP connection properties:

```
<bean id="ldapContextSource"
    class="com.jaspersoft.jasperserver.api.security.externalAuth.ldap.JSLdapContextSource">
  <constructor-arg value="${external.ldap.url}" />
  <property name="userDn" value="${external.ldap.username}" />
  <property name="password" value="${external.ldap.password}"/>
</bean>
```

To configure these properties using default_master.properties, follow these steps:

1. Open default_master.properties in a text editor.
2. Locate the following properties and set them for your LDAP server as follows:
   - `external.ldapUrl` property – The URL of your LDAP server, including the base DN.
   - `external.ldapDn` property – The distinguished name (DN) of your LDAP administrator.
   - `external.ldapPassword` property – The password of your LDAP administrator.
3. You can choose to encrypt any of the LDAP connection parameters.

The following example shows the syntax of the properties in the default_master.properties file:

```
external.ldapUrl=ldap://hostname:389/dc=example,dc=com
external.ldapDn=cn=Administrator,dc=example,dc=com
external.ldapPassword=password
```

To encrypt the password property, also set the following:

```
encrypt=true
propsToEncrypt=dbPassword,external.ldapPassword
```

See the *JasperReports Server Security Guide* for more information on encrypting passwords using buildomatic.

3.4.2 Setting LDAP Connection Parameters Manually

To set the connection parameters for the LDAP server directly in the application context file, configure the `ldapContextSource` helper bean as follows:

If you configured your LDAP connection during JasperReports Server installation or upgrade, do not set the parameters using `ldapContextSource`. You can verify whether the parameters are set by looking at the default_master.properties file.

1. In sample-applicationContext-externalAuth-LDAP[-mt].xml, locate the `ldapContextSource` bean.
2. Specify the following information:
   - `constructor-arg` value – The URL of your LDAP server, including the base DN.
   - `userDn` property – The distinguished name (DN) of your LDAP administrator.

- `password` property – The password of your LDAP administrator.

If your LDAP server is configured to allow anonymous user lookup, you don't need to specify the `userDn` and `password` properties.

Here's an example shows the syntax of the bean's constructor and properties when manually configured:

```
<bean id="ldapContextSource"
    class="com.jaspersoft.jasperserver.api.security.externalAuth.ldap.JSLdapContextSource">
  <constructor-arg value="ldap://hostname:389/dc=example,dc=com" />
  <property name="userDn"><value>cn=Administrator,dc=example,dc=com</value></property>
  <property name="password"><value>password</value></property>
</bean>
```

3.5 Performing LDAP User Search

You need to set up the search parameters for locating your users in the LDAP directory. The goal is to locate a single user entry that validates the password given during the login process. The LDAP entry located by the user search is later used to map roles and organizations.

One of the most common problems in configuring LDAP for JasperReports Server is setting up the correct search parameters to locate the users you want to map. LDAP is a rich and complex structure, with many possible variations, and LDAP directories tend to grow in complexity over time. To successfully map your users from LDAP to JasperReports Server, you need to understand the directory server tree structure of your LDAP server. Be aware that branches can be password protected and a single keyword can be used in different ways in different contexts. It can be helpful to use an open-source LDAP browser, like Apache Directory Server/Studio or JXplorer, to view and navigate your LDAP directory while troubleshooting LDAP user search problems.

Each time a user logs in, their roles and status are updated via your chosen method and synchronized with the internal jasperserver database. If you want to disable an external user or modify their external roles, you must do so in your LDAP directory.

3.5.1 Configuring JSBindAuthenticator

The sample files use an unnamed bean of the `JSBindAuthenticator` class to encapsulate search parameters for finding users in the LDAP directory.

There are two ways to configure `JSBindAuthenticator` to locate users:

- Configure the `userDnPatterns` property in the `JSBindAuthenticator` bean to match RDN patterns based on the login name provided by the user. Use this method if the login name appears in the DN of your user entries and your user entries are in a fixed branch of your LDAP directory. See **3.5.3, "Specifying userDnPatterns Parameters," on page 32** for more information.

Matching patterns is faster because it checks for a DN only in the LDAP directory, instead of a searching all users. However, it's less flexible. `userDnPatterns` is not included in the sample files by default.

- Configure the `userSearch` helper bean to perform a search for the login name provided by the user. Use this method if the login name is the value of an attribute that doesn't appear in the RDN, or if your user

entries are located in a more complex structure. See **3.5.4, "Specifying userSearch Parameters," on page 33** for more information.

You can configure pattern matching and login name search at the same time. Patterns are matched first, and login name search is done only if no match is found.

To find a user, `JSBindAuthenticator` takes the login name entered into JasperReports Server and attempts to find the correct user in the LDAP directory using bind authentication, as follows:

1. Using the specified pattern matching or search for the login name, find a candidate user entry.

The LDAP username for this candidate does not have to be the JasperReports Server login name. If they are different, the user in JasperReports Server is assigned the login name given during the login process, and not the LDAP username.

2. Attempt to log into the LDAP server using the candidate LDAP username with the login password.
3. A successful bind indicates that the right user was found.

3.5.2 Alternative to Bind Authentication

Bind authentication with the `JSBindAuthenticator` bean is the default behavior when configuring Spring Security for LDAP authentication. But Spring Security provides an alternate authentication method based on password comparison:

1. Use the administrator credentials in the `ldapContextSource` bean to log into the LDAP server.
2. Find a candidate user entry.
3. Retrieve the candidate's password attribute and compare it to the login password, or send the login password for comparison by the LDAP server.

The alternate authentication method is implemented by Spring Security in the `PasswordComparisonAuthenticator` class. Configuring Spring Security with this class is beyond the scope of this guide. For more information, see the Spring Security documentation and Javadoc.

3.5.3 Specifying userDnPatterns Parameters

If you have a fixed structure of user entries and the login name of the user appears in the RDN of your user entries, you can configure the `JSBindAuthenticator` bean with patterns to match them. The patterns are not included in the sample file, but can easily be added:

1. In sample-applicationContext-externalAuth-LDAP[-mt].xml, locate the `ldapAuthenticationProvider` bean. The unnamed bean of class `JSBindAuthenticator` is the first constructor argument.
2. Add the `userDnPatterns` property in `JSBindAuthenticator`.
3. Configure one or more patterns for matching the RDNs of user entries. For each value in the list, the server substitutes the login name entered by the user for the `{0}` placeholder, then creates a DN by appending the base DN from the LDAP URL. The LDAP URL is specified in **3.4, "Setting the LDAP Connection Parameters," on page 29**. JasperReports Server attempts to bind to the LDAP directory with the DN created with each pattern in the order they are given.

When you enter a pattern for RDN matching, make sure to use only the relative DN. Do not include the base DN that you set up when creating the LDAP connection parameters.

In the example below, JasperReports Server looks for a user whose given login name appears in the `uid` attribute of the RDN in the `ou=users` branch of the LDAP directory:

```
<bean id="ldapAuthenticationProvider" class="com.jaspersoft.jasperserver.api.security.
    externalAuth.wrappers.spring.ldap.JSLdapAuthenticationProvider">
  <constructor-arg>
    <bean class="com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.
          spring.ldap.JSBindAuthenticator">
      <constructor-arg><ref bean="ldapContextSource"/></constructor-arg>
      <property name="userDnPatterns">
        <list>
          <value>uid={0},ou=users</value>
        </list>
      </property>
    </bean>
  </constructor-arg>
  ...
</bean>
```

Notice that the domain name value only specifies ou=users. This is combined with the base DN defined by the `external.ldapUrl` property in the default_master.properties file or the `constructor-arg` value in the `ldapContextSource` bean to create the full DN.

3.5.4 Specifying userSearch Parameters

Use the `userSearch` bean to find users if they don't match a simple pattern. In particular, if you're authenticating users for one or more organizations, it is likely user entries are in multiple branches of your directory.

To search for user entries, locate the helper bean `userSearch` in sample-applicationContext-externalAuth-LDAP [-mt].xml and specify the following information:

- An optional branch RDN where user entries are located. If not specified, the search includes your entire LDAP directory starting from the base DN of the LDAP URL specified in **3.4, "Setting the LDAP Connection Parameters," on page 29**.
- An LDAP filter expression to compare any attribute or combination of attributes with the login name. JasperReports Server substitutes the login name entered by the user for the `{0}` placeholder to perform the search.
- Whether or not the search should extend to all subtrees beneath the branch DN or, when no branch DN is specified, beneath the base DN.

When you enter a location for user search, make sure to use only the relative DN. Do not include the base DN that you set up when creating the LDAP connection parameters.

The following example shows the syntax of the bean's constructor and property:

```
<bean id="userSearch" class="com.jaspersoft.jasperserver.api.security.
             externalAuth.wrappers.spring.ldap.JSFilterBasedLdapUserSearch">
  <constructor-arg index="0"><value>ou=users</value></constructor-arg>
  <constructor-arg index="1"><value>(uid={0})</value></constructor-arg>
  <constructor-arg index="2"><ref bean="ldapContextSource" /></constructor-arg>
  <property name="searchSubtree"><value>true</value></property>
</bean>
```

The combination of these three parameters lets you optimize the search for your user entries and reduce the load on your LDAP directory. For example, if your users are located in a dedicated branch of your LDAP structure, specify it in the first constructor argument to avoid searching the entire tree.

3.5.5 LDAP Search for Multiple Organizations

When you're authenticating users for one or more organizations in a commercial edition of JasperReports Server, the search parameters must be able to locate all users for all organizations for these reasons:

- The mapping from LDAP user entries to organizations in the server requires the user entries to be in a hierarchical tree structure that mimics the intended organization hierarchy. You can't use attribute values or group membership in LDAP to define organizations.
- External authorization doesn't allow the user to enter an organization name. So the search must find the username among all organizations.

This has two implications:

1. Your choice of pattern matching or search depends on the structure of user entries in LDAP. For example, if you have a small fixed number of organizations, you could match them with a pattern for each one, as follows:

```
      <property name="userDnPatterns"><list>
        <value>uid={0},ou=users,o=Finance</value>
        <value>uid={0},ou=users,o=HR</value>
        <value>uid={0},ou=users,o=Executive</value></list>
      </property>
```

   But if you have a large number of organizations, or if the number or names of organizations can change, you need to search for every potential user. Depending on your LDAP structure, you may be able to specify a search base in `constructor-arg index="0"`; the example below doesn't have one.

```
<bean id="userSearch" class="com.jaspersoft.jasperserver.api.security.
      externalAuth.wrappers.spring.ldap.JSFilterBasedLdapUserSearch">
  <constructor-arg index="0"><value></value></constructor-arg>
  <constructor-arg index="1"><value>(uid={0})</value></constructor-arg>
  <constructor-arg index="2"><ref bean="ldapContextSource" /></constructor-arg>
  <property name="searchSubtree"><value>true</value></property>
</bean>
```

2. You cannot implement external authentication for two users with the same login name in different organizations. LDAP supports this as long as the two users have distinct DNs, and JasperReports Server supports this for the default internal authentication. But during external authentication, organization mapping happens after user search, so the user search must return a single LDAP entry:
   - Pattern matching stops at the first match based on the login name. As a result, only the user whose LDAP entry pattern is listed higher in the list can log in.
   - Search returns more than one entry. As a result, login fails for both users with the same login name.

3.6 Mapping the User Roles

An external user's roles in JasperReports Server are based on the groups to which that user belongs in LDAP. The server does a second search in the LDAP directory to determine any user roles. The mapping defines the

location of the group definitions in LDAP, how to find the user's groups, and any transformation of the group name for use in the server as a role name.

3.6.1 Configuring the User Role Mapping

The mapping for user roles is configured in a bean of the `JSDefaultLdapAuthoritiesPopulator` class, a Jaspersoft wrapper class that is itself part of the configuration of the `ldapAuthenticationProvider` bean.

Some LDAP servers support other user-grouping mechanisms, like `nsrole` in the Sun Directory Server. These can be mapped into JasperReports Server roles through the configuration parameters below, by extending the `JSDefaultLdapAuthoritiesPopulator` class, or a combination of both. Such configurations are beyond the scope of this guide.

To configure the mapping for user roles in sample-applicationContext-externalAuth-LDAP[-mt].xml, locate the bean of the `JSDefaultLdapAuthoritiesPopulator` class, the second constructor argument of `ldapAuthenticationProvider`, and specify the following information:

- `constructor-arg index="1"` – An optional branch DN where group entries are located. If not specified, the search covers your entire LDAP directory starting from the base DN.
- `groupRoleAttribute` property – The attribute whose value is mapped to the name of the JasperReports Server role. Often, this is the `cn` attribute that gives the name of the role in the RDN of the group entry. But it can be any attribute, for example a custom attribute named `Jaspersoft Role Name` defined by a custom LDAP schema.
- `groupSearchFilter` property – A group search filter that locates entries representing groups to which the user belongs. For static groups, this filter should detect entries with the `groupofuniquenames` object class and with a `uniqueMember` value that matches the DN found by the user search. You can use the following parameters:
  - `{0}` represents the full DN of the user entry.
  - `{1}` represents the username.
- `searchSubtree` property – Whether or not the search should extend to all subtrees beneath the branch DN, or beneath the base DN when no branch DN is specified.

`JSDefaultLdapAuthoritiesPopulator` is a wrapper class of the Spring Security `DefaultLdapAuthoritiesPopulator` class. Spring Security supports additional properties; see the Spring Security 3.2 documentation for more information.

All internal and external users are assigned `ROLE_USER` by default. So you never need to create or map this role in your LDAP directory.

The following shows an example syntax of the constructor arguments and properties that uses `groupofuniquenames`:

```
<bean id="ldapAuthenticationProvider"  class="com.jaspersoft.jasperserver.api.security.
      externalAuth.wrappers.spring.ldap.JSLdapAuthenticationProvider">
  <constructor-arg> ...
  </constructor-arg>
  <constructor-arg>
    <bean class="com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.
                spring.ldap.JSDefaultLdapAuthoritiesPopulator">
      <constructor-arg index="0"><ref bean="ldapContextSource"/></constructor-arg>
```

```
      <!-- optional branch DN for roles -->
      <constructor-arg index="1"><value></value></constructor-arg>
      <property name="groupRoleAttribute"><value>cn</value></property>
      <property name="groupSearchFilter"><value>
        (&amp;(uniqueMember={0})(objectclass=groupofuniquenames))</value></property>
      <property name="searchSubtree"><value>true</value></property>
    </bean>
  </constructor-arg>
</bean>
```

Be careful when defining the properties for mapping user roles. The search for groups in the LDAP directory must not cause an error, otherwise the entire login will fail. For example, if you specify a branch DN that doesn't exist, the search will cause an error, and users will be unable to log in. A successful search that returns no results will allow users to log in, but without having the intended roles.

After the mapping has determined the role names given to the external user in JasperReports Server:

- In the community edition, which doesn't have the organization architecture, the roles are synchronized with existing roles and assigned to the user.
- In commercial editions, which have the organization architecture, the external user and roles are assigned to an organization that's either the default single organization or an organization mapped from the DN of the LDAP user. Organization mapping is described in **3.7, "Mapping the User Organization," on page 40**. If you intend for one of the mapped roles to provide administrator privileges, you must explicitly map it to the system roles, as described in **3.6.2, "Mapping Roles to System Roles," on page 36**. Otherwise, all mapped roles are created in the mapped organization.

Synchronization creates roles in JasperReports Server if they don't exist, as described in **2.4.3, "Synchronization of Roles," on page 18**.

3.6.2 Mapping Roles to System Roles

When the organization mapping is complete, synchronization invokes `mtExternalUserSetupProcessor` commercial editions) or `externalUserSetupProcessor` (community edition) to create the external user and roles in that organization. JasperReports Server includes an additional mapping of roles to system roles so you can grant administrator privileges to your external users. Using this feature, LDAP entries belonging to custom groups can be granted system or organization admin roles in JasperReports Server.

Depending on your deployment, you can map roles to system roles in one of two ways:

- Configure the `mtExternalUserSetupProcessor` or `externalUserSetupProcessor` bean with `organizationRoleMap` to map between external and internal roles. The processor checks if the user has an external role as a map entry key. If the user has the role, the processor assigns the user the internal role in the map entry value instead of the external role in the key.
- Map user roles statically using the `mtExternalUserSetupProcessor` or `externalUserSetupProcessor` bean.

One practical consequence of external administrator role mapping is that external authentication can be used exclusively. When properly set up, you can have external users who are system or organization administrators. Then you don't need to have the superuser and jasperadmin users. However, you must ensure that every organization has an external user mapped to the organization with the correct attributes to have organization admin privileges.

Administrators of your LDAP server cannot log into JasperReports Server using their LDAP administrator credentials.

In most LDAP servers, users and administrators are stored in different base DNs. For example, you might store user entries in dc=example,dc=com, but administrators are stored under cn=Administrators,cn=config or ou=system. The mechanism for locating users during authentication can only search in a single base DN, so administrators in a different one cannot be found.

3.6.2.1 organizationRoleMap

System and organization admin privileges are determined by the `ROLE_SUPERUSER` and `ROLE_ADMINISTRATOR` system roles at the root level. Using the `organizationRoleMap` property, you can assign these system roles to LDAP entries based on custom group membership. This property can be used in addition to the properties that map group names to organization roles.

Whether you map users and roles to a single organization or multiple organizations, you can define this additional mapping between any role name that your mapping creates and any system role. You specify role mapping via the `organizationRoleMap` property of the `mtExternalUserSetupProcessor` bean (commercial editions) or `externalUserSetupProcessor` (community edition).

- `organizationRoleMap` property – A list of key/value pairs that maps external role names to internal ones. The key should be a role name that your mapping creates, after adding the prefix and capitalization as configured in `JSDefaultLdapAuthoritiesPopulator`. For commercial JasperReports Server deployments, you need to choose the level at which the role is assigned:
  - To map to an internal role at the organization level, append `|*` to the name of the internal role, for example, `ROLE_EXTERNAL_USER|*`. Roles mapped at the organization level do not have administrative privileges.
  - To map to an internal role at the system (null) level, do not modify the internal role name, for example, ROLE_EXTERNAL_ADMINISTRATOR. Roles at the system level are usually reserved for special users like the system administrator and allow access to the repository folder of all other organizations.

For example, if your LDAP user belongs to a group named `jrsadmin` that's mapped to the name `ROLE_ADMIN_EXTERNAL_ORGANIZATION`, the following code example would assign that user the `ROLE_ADMINISTRATOR` system role that makes the user an organization admin. This example shows how to create this system role mapping in a single organization configuration for commercial editions:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
        externalAuth.processors.MTExternalUserSetupProcessor" parent="abstractExternalProcessor">
  <property name="userAuthorityService">
    <ref bean="${bean.internalUserAuthorityService}"/>
  </property>
  <property name="defaultInternalRoles">
    <list>
      <value>ROLE_USER</value>
    </list>
  </property>
```

```
  <property name="organizationRoleMap">
    <map>
      <entry>
        <key>
          <value>ROLE_ADMIN_EXTERNAL_ORGANIZATION</value>
        </key>
        <value>ROLE_ADMINISTRATOR</value>
      </entry>
    </map>
  </property>
</bean>
```

If the value `ROLE_ADMINISTRATOR` in the key value pair had ended with |* (`ROLE_ADMINISTRATOR|*`), the user would have been assigned `ROLE_ADMINISTRATOR` at the organization level.

Roles not mapped to system roles are created and synchronized in the mapped organization, as described in **2.4.3, "Synchronization of Roles," on page 18**. In particular, if the name `ROLE_ADMINISTRATOR` or `ROLE_SUPERUSER` are mapped from the LDAP groups, but not mapped to system roles, they're created as organization roles and assigned to the user. As organization roles, they don't grant any access permissions, which can be very confusing for administrators. Avoid LDAP groups and role mappings that create these names as organization roles.

3.6.2.2 Defining User Roles Statically

If you're mapping all your external users to a single organization, you can assign static roles to users. This lets you specify a list of administrative users and roles, and a list of roles for non-administrative users. To define static roles, use the `externalUserSetupProcessor` or `mtExternalUserSetupProcessor` bean. To set up static roles, locate the version of the bean used in your sample file and configure the following properties:

- `adminUserNames` property — A list of usernames granted internal administrator privileges in JasperReports Server. The username values must exactly match the usernames authenticated and returned by the external authority.
- `defaultAdminRoles` property — A list of JasperReports Server internal roles. These are assigned to every user in the list of administrators.
- `defaultInternalRoles` property — A list of JasperReports Server roles assigned to every user not in the list of administrators.

The following example shows how to use the `mtExternalUserSetupProcessor` bean to define static roles. The configuration for `externalUserSetupProcessor` is similar:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
    externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
       ...
  <property name="adminUsernames">
    <list>
      <value>myorgadmin</value>
    </list>
  </property>

  <property name="defaultAdminRoles">
    <list>
      <value>ROLE_USER</value>
      <value>ROLE_ADMINISTRATOR</value>
    </list>
  </property>
```

```
  <property name="defaultInternalRoles">
    <list>
      <value>ROLE_USER</value>
    </list>
  </property>
...
```

3.6.3 Setting Default Roles

You can assign roles to all users using the `defaultInternalRoles` property of `externalUserSetupProcessor` or `mtExternalUserSetupProcessor`. The following example shows how to use this property in `externalUserSetupProcessor` to assign `ROLE_USER` to all users, in addition to the roles assigned by mapping:

```
  <property name="defaultInternalRoles">
    <list>
      <value>ROLE_USER</value>
    </list>
  </property>
```

3.6.4 Avoiding Role Collisions

If an external role has the same name as an internal role at the same organization level, JasperReports Server adds a suffix such as _EXT to the external role name to avoid collisions. For example, a user with the externally defined role ROLE_ADMINISTRATOR is assigned the role ROLE_ADMINISTRATOR_EXT in the JasperReports Server database. This ensures that internal administrator accounts like jasperadmin and superuser can still log in as internal administrators with the associated permissions.

You can set the extension in the `conflictingExternalInternalRoleNameSuffix` property in the `externalUserSetupProcessor` or `mtExternalUserSetupProcessor` bean. If the property doesn't appear in the bean, the extension is still implemented but defaults to _EXT. The following example shows how to configure this property:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
    externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
  <property name="conflictingExternalInternalRoleNameSuffix"
        value="_EXTERNAL"/>
  <property name="organizationRoleMap">
      ...
     <!-- Example of mapping customer roles to JRS roles -->
      ...
  </property>
```

3.6.5 Restricting the Mapping to Whitelisted Roles

You may not want every role in your external authority to appear as a role in JasperReports Server. Use the `permittedRolesRegex` property of the `externalUserSetupProcessor` bean or `mtExternalUserSetupProcessor` bean to specify which external roles become roles in JasperReports Server. You can use regular expressions to specify multiple roles that match the expression.

For example, to restrict the roles you create in JasperReports Server to roles that begin with JRS_ or EXT_ in your external authority, you would configure `permittedRolesRegex` in a way similar to the following:

```
<property name="permittedRolesRegex">
    <list>
        <value>JRS_.*</value>
        <value>EXT_.*</value>
    </list>
</property>
```

To allow all roles, use .* or comment out the property. If the property is omitted, all roles in the external authority are synchronized with roles in JasperReports Server.

3.6.6 Supporting Additional Characters in Role Names

The default mapping from attributes in your external authentication server to roles in JasperReports Server supports only alphanumeric characters and underscores. If a role in your external authority contains unsupported characters, each sequence of unsupported characters is replaced with a single underscore. For example, ROLE$-DEMO)EXT maps to ROLE_DEMO_EXT.

You can extend the supported character set by modifying the `permittedExternalRoleNameRegex` property of the `externalUserSetupProcessor` bean or `mtExternalUserSetupProcessor` bean. Check the sample configuration file for your deployment to determine which bean to modify.

The default value of the `permittedExternalRoleNameRegex` property is the regular expression [A-Za-z0-9_]+. Edit this expression to add supported characters. For example, the following syntax allows alphanumeric characters, underscores, and the Cyrillic letter Я (Unicode 042F):

```
<bean id="mtExternalUserSetupProcessor"  class="com.jaspersoft.jasperserver.api.security.
        externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
 <property name="userAuthorityService">
   <ref bean="${bean.internalUserAuthorityService}"/>
 </property>
  .....
 <property name="permittedExternalRoleNameRegex"
     value="[A-Za-z0-9_\u042F]+">
</bean>
```

Do not allow the following in role names: spaces, periods or |, [ ], `, ", ', ~, !, #, $, %, ^, &, [,], *, +, =, ;, :, ?, <, >, }, {, ), (, ], [, /, or \. Adding these characters in the `permittedExternalRoleNameRegex` property may cause unexpected behavior, such as the inability to delete or edit roles containing those characters.

3.7 Mapping the User Organization

Organizations are a feature of JasperReports Server commercial editions. Skip this section if you have JasperReports Server community edition.

In implementation that supports multiple organizations, all users and roles except for system administrators and system roles belong to organizations. In turn, each organization determines the folders that its users can access

in the repository. So the final part of mapping is to determine an organization ID for the external user and roles, based on the user's RDN in the directory.

If your JasperReports Server supports multiple organizations, you have two ways to set the user organization for external users:

- Create a mapping from RDNs in your LDAP server to organizations in JasperReports Server, as described in **3.7.1, "Mapping to Multiple Organizations," on page 41**.
- If you want all users to be in just one of your organizations, use the `externalTenantSetupProcessor` bean to specify the organization, as described in **3.7.2, "Mapping to a Single Organization," on page 45**.

If your JasperReports Server deployment supports only a single organization (all community deployments and some professional editions), you do not need to set organization information.

3.7.1 Mapping to Multiple Organizations

LDAP is well suited to mapping users into organizations, because LDAP itself has a hierarchical structure of user entries that's often used to represent separate organizations, such as the internal departments of a company. The LDAP tree structure is reflected in the elements of the RDN of each user entry, and the server maps this RDN into an organization or hierarchy of organizations for the external user. For example, the users `uid=jack,ou=audit,ou=finance,dc=example,dc=com` and `uid=jill,ou=accounting,ou=finance,dc=example,dc=com` could be mapped to the organizations `audit` and `accounting`, respectively, both of which are sub-organizations of `finance`.

In order to ensure consistency, the server must create the organization of any external user if the organization does not already exist. The server also creates any organization that does not exist in the hierarchy of organizations mapped from the user RDN. To avoid "stray" organizations outside of your intended hierarchy, test your mapping against all potential user DNs in your LDAP directory.

Organizations created during external user login have an administrator with a default password. The admin username and password is configurable. See **3.7.1.2, "Setting Up Default Admins for Organizations," on page 42** for more information. For security reasons, you should change the default password of any organization admin created. See **2.4.4, "Initialization of JasperReports Server for External Users," on page 19** for a process to initialize the server, including organization admins, before going into production with external authentication.

3.7.1.1 Setting Up Organization Mapping

Specify the following information in the `ldapExternalTenantProcessor` bean to map the RDN of the user to a hierarchy of organizations in JasperReports Server:

- `excludeRootDn` property – Property that specifies whether the base DN, also called root DN, should be mapped along with the RDN. For example, if the property list for `organizationRDNs` contains `dc` and you don't exclude the base DN of `dc=example,dc=com`, the base DN maps to the following: the organization ID `example` nested inside the organization ID `com` nested inside the specified root organization. The base DN is part of the LDAP URL specified in **3.4, "Setting the LDAP Connection Parameters," on page 29**.
- `organizationRDNs` property – A list of attribute names that determines which RDN values should be mapped to organization names. The names in this list determine the RDNs that creates a hierarchy of organizations in JasperReports Server. For example, if you specify the value `ou`, each RDN with `ou=<name>` creates a level in the hierarchy of mapped organizations. If this list is blank or none of the attributes match the RDN of the user entry, the `defaultOrganization` property determines the organization name.
- `rootOrganizationId` property – The ID of an organization under which any mapped organizations are created as sub-organizations. If the root organization ID is absent or blank (`""`), the server creates the

organization(s) mapped in `organizationRDNs` as children of the default organization shipped with JasperReports Server.

- `defaultOrganization` property (optional) – The ID of an organization assigned to users that would otherwise be mapped to a null organization ID.

If `excludeRootDn = true`, `defaultOrganization = ""` or is absent, and no `organizationRDNs` match in the DN of the user, the user will have a null organization ID. The null organization ID is usually reserved for special users like the system administrator and allows access the repository folder of all other organizations. To avoid this mapping, specify a value for `defaultOrganization` or ensure that every user has one of the `organizationRDNs`.

The following example shows the syntax of the `ldapExternalTenantProcessor` bean and its properties:

```
<bean id="ldapExternalTenantProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
      externalAuth.processors.ldap.LdapExternalTenantProcessor" parent="abstractExternalProcessor">
  <property name="ldapContextSource" ref="ldapContextSource" />
  <property name="multiTenancyService">
    <ref bean="internalMultiTenancyService"/></property>
  <property name="excludeRootDn" value="false"/>
  <!-- only following RDNs matter in creating the organization hierarchy -->
  <property name="organizationRDNs">
    <list>
      <value>o</value>
      <value>ou</value>
    </list>
  </property>
  <property name="rootOrganizationId" value="organization_1"/>
  <property name="defaultOrganization" value="#{configurationBean.tenantIdNotSupportedSymbols}"/>

...
</bean>
```

For example, given the `ldapExternalTenantProcessor` bean configuration above, an LDAP user with the DN `uid=jack,ou=audit,ou=finance, dc=example,dc=com` is placed in a organization named `audit` which is a child of an organization named `finance`, which in turn is a child of `organization_1`. This example illustrates that it's not possible to map only one of the two RDN components if they have the same attribute. In other words, the mapping mechanism does not let you choose to create only the `audit` or the `finance` organization; both are created if you specify `ou` in the list of `organizationRDNs`.

By default, the sample-applicationContext-externalAuth-LDAP-mt.xml file maps users to multiple organizations. If you want to map all users to a single organization, see **3.7.2, "Mapping to a Single Organization," on page 45**

3.7.1.2 Setting Up Default Admins for Organizations

In a multi-organization deployment, JasperReports Server creates a `jasperadmin` user whenever you create a new organization. The `jasperadmin` user is also given a standard default password. When creating multiple organizations using external authentication, you can set a different default password for jasperadmin, remove the jasperadmin user, and/or create additional default users in each new organization created by external authentication. Optionally, you can encrypt the password in the configuration files. See the *JasperReports Server Security Guide* for more information on default users in every organization.

For security reasons, you should change the default password of any organization admin. See **2.4.4, "Initialization of JasperReports Server for External Users," on page 19** for a process to initialize the server, including organization admins, before going into production with external authentication.

To set up admin users:

1. Open your sample-applicationContext-xxx-externalAuth.xml file in a text editor.
2. Locate the `externalTenantSetupUsers` property in the `ldapExternalTenantProcessor` bean.
3. The sample contains a bean of class `ExternalTenantSetupUser` already configured for `jasperadmin`.

```
<property name="externalTenantSetupUsers">
  <list>
    <bean class="com.jaspersoft.jasperserver.multipleTenancy.security.
                externalAuth.processors.MTAbstractExternalProcessor.ExternalTenantSetupUser">
      <property name="username" value="${new.tenant.user.name.1}"/>
      <property name="fullName" value="${new.tenant.user.fullname.1}"/>
      <property name="password" value="${new.tenant.user.password.1}"/>
      <property name="emailAddress" value="${new.tenant.user.email.1}"/>
      <property name="roleSet">
        <set>
          <value>ROLE_ADMINISTRATOR</value>
          <value>ROLE_USER</value>
        </set>
      </property>
    </bean>
  </list>
</property>
```

4. To create additional admin users for each external organization, create a bean of class `ExternalTenantSetupUser` for each admin user you want.

```
    <bean class="com.jaspersoft.jasperserver.multipleTenancy.security.
                externalAuth.processors.MTAbstractExternalProcessor.ExternalTenantSetupUser">
      <property name="username" value="${new.tenant.user.name.2}"/>
      <property name="fullName" value="${new.tenant.user.fullname.2}"/>
      <property name="password" value="${new.tenant.user.password.2}"/>
      <property name="emailAddress" value="${new.tenant.user.email.2}"/>

      <property name="roleSet">
        <set>
          <value>ROLE_ADMINISTRATOR</value>
          <value>ROLE_USER</value>
        </set>
      </property>
    </bean>
```

5. The ${...} syntax above references values configured in the following file:
   <js-install>\buildomatic\conf_source\iePro\js.config.properties file.
   To set these values, open <js-install>\buildomatic\conf_source\iePro\js.config.properties and edit the entries there.

```
new.tenant.user.name.1=jasperadmin
new.tenant.user.fullname.1=jasperadmin
new.tenant.user.password.1=mynewpassword
new.tenant.user.email.1=

new.tenant.user.name.2=anotheradmin
new.tenant.user.fullname.2=Another Admin
new.tenant.user.password.2=anotherpassword
new.tenant.user.email.2=
```

Note: The property names, for example, `new.tenant.user.name.1`, are arbitrary. You can use any name for each property as long as the name in the applicationContext-externalAuth-xxx.xml file matches the name in the js.config.properties file.

6. If you want to obfuscate the default passwords in the js.config.properties files, encrypt them as described in the *JasperReports Server Security Guide*. Obfuscation must be implemented before you install the server.
7. If you don't want to obfuscate default passwords, you can eliminate the reference to js.config.properties and instead configure the values directly in the `externalTenantSetupUsers` property in the applicationContext-externalAuth-xxx.xml file. For example:

```
    <property name="username" value="anotheradmin"/>
    <property name="fullName" value="Another Admin"/>
    <property name="password" value="anotherpassword"/>
    <property name="emailAddress" value=""/>
```

3.7.1.3 Mapping Organization Names

You have the option to use the `organizationMap` property in the `ldapExternalTenantProcessor` bean to map organization names extracted from your external authority to organization names in JasperReports Server. To do this, create a key/value pair for each organization you want to map, specifying the external organization name as the key and the organization name in JasperReports Server as the value. When mapping organizations, the server determines the mapped name and uses it as the name, ID, and description of the organization.

For example, the following would map users in External_Org_1 in the external authority to JRS_Org_1 in JasperReports Server and users in External_Org_2 in the external authority to JRS_Org_2 in JasperReports Server:

```
<property name="organizationMap">
    <map>
        <entry key="External_Org_1" value="JRS_Org_1" />
        <entry key="External_Org_2" value="JRS_Org_2" />
    </map>
</property>
```

The `organizationMap` property is optional. Any organization in your external authority that is not listed in `organizationMap` is mapped to an organization of the same name in JasperReports Server. However, if an organization in your external authority contains unsupported characters, each sequence of unsupported characters is replaced with a single underscore. For example, Human Resources maps to Human_Resources.

The `tenantIdNotSupportedSymbols` property of the `configurationBean` bean in the applicationContext.xml file lists the unsupported characters, including spaces and the following characters: `|`, `&`, `*`, `?`, `<`, `>`, `/`, `\`, `~`, `!`, `#`, `$`, `%`, `^`, `[`, `]`, or a space. If you want to list additional characters that should be replaced with an underscore, you can add them in this bean. However, we do not recommend removing any of the pre-defined characters, as JasperReports Server may not handle them correctly.

3.7.2 Mapping to a Single Organization

If you have a commercial version of JasperReports Server, you can choose to map all external users to a single organization, for example, in the following cases:

- You have a commercial JasperReports Server deployment that does not implement multiple organizations, but instead uses the default organization. This includes commercial versions licensed for a single organization. In this case, externally authenticated users must be mapped to the default organization.
- You have multiple organizations in JasperReports Server, but still want all external users to be placed in a single organization.

The following steps show how to map all external users to a single organization using the sample-applicationContext-externalAuth-LDAP-mt.xml file:

8. In sample-applicationContext-externalAuth-LDAP-mt.xml, locate the first instance of the `ldapExternalTenantProcessor` bean and comment the bean out.
9. Locate the second instance of the `ldapExternalTenantProcessor` bean and uncomment it.
10. Set the `defaultOrganization` property to the organization you want assigned to all external LDAP users.

The `ldapExternalTenantProcessor` bean is not available in the community edition. You don't need to set the organization in the community edition.

`ldapExternalTenantProcessor` is an example of a processor. For more information about processors, see **7.3, "Creating a Custom Processor," on page 102**.

The following example places all external users in the default organization, `organization_1`.

```
<bean id="ldapExternalTenantProcessor"
        class="com.jaspersoft.jasperserver.multipleTenancy.security.
              externalAuth.processors.ldap.LdapExternalTenantProcessor"
        parent="abstractExternalProcessor">
  <property name="ldapContextSource" ref="ldapContextSource"/>
  <property name="multiTenancyService"><ref bean="internalMultiTenancyService"/></property>
  <property name="excludeRootDn" value="true"/>
  <property name="defaultOrganization" value="organization_1"/>
</bean>
```

Make sure you specify a value for the `defaultOrganization`. If `defaultOrganization` is left empty, users may be mapped to the null organization id. This is usually reserved for special users like the system administrator and allows access to the repository folder of all other organizations.

3.8 Setting Up Multiple Providers

The file sample-applicationContext-externalAuth-LDAP[-mt].xml contains an LDAP-specific authentication manager, `ldapAuthenticationManager`, configured as follows.

```
<bean id="ldapAuthenticationManager" class="com.jaspersoft.jasperserver.api.security.
      externalAuth.wrappers.spring.JSProviderManager">
  <constructor-arg index="0">
    <list>
      <ref bean="ldapAuthenticationProvider"/>
      <ref bean="${bean.daoAuthenticationProvider}"/>
    </list>
  <constructor-arg>
</bean>
```

The `ldapAuthenticationManager` bean attempts to authenticate a user session with each provider in the list in the order they appear. When one of the providers successfully authenticates the user, the rest of the providers are skipped.

As shown in the example above, you can list other authentication providers with LDAP.

The `daoAuthenticationProvider` is the default internal authentication using the internal database. You may keep this provider in the list to access the `jasperadmin` and `superuser` accounts, or any other administrator accounts you've created. Or, if you have configured system role mapping as described in **3.6.2, "Mapping Roles to System Roles," on page 36**, you can remove the `daoAuthenticationProvider` from the list.

The internal database contains accounts for all of the external users that have previously logged into the server. However, these external accounts do not contain passwords and cannot be used for authentication with the default internal authentication, for example when the LDAP server is unavailable.

3.9 Authentication with Microsoft Active Directory

Microsoft Active Directory can be used to authenticate users through the `ldapAuthenticationProvider` provided by Spring Security. When setting up an LDAP provider for Active Directory, you should be aware of the following:

- You must use configure user search to work with the `sAMAccountName` attribute containing the user's login name. See **3.9.1, "Configuring User Search for Active Directory," on page 46** for more information.
- You may need to set the Spring `referral` property in `LdapContextSource` to `follow`. See **3.9.2, "Configuring the Spring Referral Property," on page 47** for more information.

In addition, the structure of an Active Directory instance can become quite complex and this can be a challenge when setting up user search.

3.9.1 Configuring User Search for Active Directory

The most important difference in configuration between Active Directory and a standard LDAP server is the need to search for the `sAMAccountName` attribute containing the user's login name. Because of this requirement, you must use the `JSBindAuthenticator` bean, along with the `userSearch` bean and corresponding property in `JSBindAuthenticator`.

The following example shows how to configure the `userSearch` bean for LDAP authentication with the special syntax for Active Directory. This configuration is only an example; you need to configure the `JSBindAuthenticator` and `ldapContextSource` beans correctly for your LDAP server, as described earlier in this chapter.

The following example shows how you might set the `sAMAccountName` attribute.

```
<bean id="userSearch"  class="com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.
       spring.ldap.JSFilterBasedLdapUserSearch">
  <constructor-arg index="0"><value>cn=Users</value></constructor-arg>
  <constructor-arg index="1"><value>(&amp;(sAMAccountName={0}))</value></constructor-arg>
  <constructor-arg index="2"><ref bean="ldapContextSource"/></constructor-arg>
  <property name="searchSubtree"><value>true</value></property>
</bean>
```

You must also include a role mapping for any roles you want to import to JasperReports Server, and you must include an organization mapping if you implement multiple organizations. For more information, see **3.6, "Mapping the User Roles," on page 34**.

Note that for Active Directory, `sAMAccountName` must be in `constructor-arg index="1"` of the `userSearch` bean.

3.9.2 Configuring the Spring Referral Property

Some Active Directory servers are unable to automatically follow referrals, which leads to a `PartialResultException` being thrown in searches. To handle this, set the Spring `referral` property in `LdapContextSource` to follow, for example:

```
<bean id="ldapContextSource"
        class="com.jaspersoft.jasperserver.api.security.externalAuth.ldap.JSLdapContextSource">
  <constructor-arg value="ldap://hostname:389/dc=example, dc=com" />
  <property name="userDn">
    <value>cn=Administrator, dc=example, dc=com</value>
  </property>
  <property name="password"><value>password</value></property>
  <property name="referral" value="follow"/>
</bean>
```

3.10 Troubleshooting LDAP Configurations

This section describes how to diagnose and resolve some common problems with configuring external authentication for LDAP.

3.10.1 Planning for Troubleshooting

If you have problems configuring external authentication for LDAP, first make sure that you have configured logging and located the files you need to diagnose and resolve the issue. If you need to contact Jaspersoft technical support, they will ask you for this information:

1. Enable logging, including detailed LDAP logging, as described in **2.2, "Configuring Logging for Debugging," on page 13**. For more information about logging in JasperReports Server, see the *JasperReports Server Administrator Guide*.
2. Find your application context file for external authentication. The deployed file is named applicationContext-externalAuth.xml; the sample file is named applicationContext-externalAuth-LDAP-mt.xml or applicationContext-externalAuth-LDAP.xml. This file can be modified in any text editor.
3. Export the LDIF file for your LDAP server. You can view the exported file in an LDIF editor. Most LDAP browsers and/or LDIF editors support export. If you do not have an LDIF editor, you can find a free open-

source editor on the web, such as Apache Directory Studio. Many common IDEs, such as Eclipse, also support LDIF plugins.

3.10.2 "Invalid Credentials Supplied" Errors

One common error you will see when trying to log in to JasperReports Server is an invalid credential error, with the following message:

```
Invalid credentials supplied.
Could not login to JasperReports Server.
```

This error can be misleading, because it can come from a wide range of root causes, including problems that are not directly related to the credentials used. These include:

- Communication issues
- User search issues

"Invalid credentials supplied" errors can be misleading, as they are not always related to the credentials used.

3.10.2.1 Problems Communicating with the LDAP Server

If you receive an invalid credentials error, the first thing to check for is errors connecting to the LDAP server. Search the jasperserver.log file for any stack trace containing the following:

```
javax.naming.CommunicationException
```

If you see this in the logs, you need to dig a little further to find the cause of the communication exception.

3.10.2.1.1 Incorrect Connection URL

Problem

If the URL for the LDAP server is incorrect in applicationContext-externalAuth.xml, you will see an error such as the following:

```
ERROR EncryptionAuthenticationProcessingFilter,http-apr-8630-exec-6:218 - An internal error occurred
while trying to authenticate the user.

org.springframework.security.authentication.InternalAuthenticationServiceException: localhost:10399;
nested exception is javax.naming.CommunicationException: localhost:10399 [Root exception is
java.net.ConnectException: Connection refused: connect]
```

Solution

To fix this error, locate the following lines in applicationContext-externalAuth.xml and verify that `myLDAPServer` is correct hostname for your LDAP server and that and `port` is your LDAP server port:

```
<bean id="ldapContextSource"
              class="com.jaspersoft.jasperserver.api.security.externalAuth.ldap.JSLdapContextSource">
    <constructor-arg value="ldap://myLDAPServer:port"/>
```

Edit this information to point to the correct server and port.

3.10.2.1.2 Timeout Errors

Problem

If the connection is timing out while trying to talk to the LDAP server, you will see a "Connection timed out" error in the log, such as:

```
ERROR EncryptionAuthenticationProcessingFilter,http-apr-8630-exec-3:218 - An internal error occurred
while trying to authenticate the user.
org.springframework.security.authentication.InternalAuthenticationServiceException:
172.17.10.63:10390; nested exception is javax.naming.CommunicationException: 172.17.10.63:10390 [Root
exception is java.net.ConnectException: Connection timed out: connect]
```

Solution

To fix this error, ensure that the LDAP server is reachable from the server that is hosting JasperReports Server. Possible causes for connectivity problems include (but are not limited to): firewalls, anti-virus software, or an incorrectly configured DMZ.

3.10.2.2 Problems with User Search

"Invalid credentials supplied" errors are frequently caused by problems with the way user search is configured.

3.10.2.2.1 Unable to Find an LDAP Branch

Problem

If JasperReports Server can't find the user because you have not configured the server to communicate with an existing branch within LDAP, you may see an "Invalid search base" error in jasperserver.log. For example:

```
org.apache.directory.api.ldap.model.exception.LdapNoSuchObjectException: ERR_648 Invalid search base
ou=users,dc=example,dc=com
```

Solution

To resolve this, check with your LDAP admin that the search base you are using exists within your LDAP directory. The search base is usually specified in the `<constructor-arg index="0">` parameter in the `userSearch` bean in applicationContext-externalAuth.xml.

3.10.2.2.2 Incorrect or Missing Partition

Problem

If JasperReports Server can't find the user because the search is not configured to look in the correct partition, you may see a "Cannot find a partition" error in jasperserver.log, for example:

```
ERR_268 Cannot find a partition for ou=users,o=mojo55:
org.apache.directory.api.ldap.model.exception.LdapNoSuchObjectException: ERR_268 Cannot find a par-
tition for ou=users,o=mojo55
```

Solution

This error can arise if you are importing your LDIF file. In this case, the partitions may not be created automatically, and therefore are not searchable. Check the LDAP connection URL in the `ldapContextSource`

bean in applicationContext-externalAuth.xml. If the partition is not present, you can append it to the bean.

3.10.2.2.3 Invalid Search Filter

Problem

If the search filter is not constructed correctly in your applicationContext-externalAuth.xml file your connection will fail. This error can be tricky because it does not always give an error code in the log. Instead, the query is valid, but when it searches your LDAP directory, it simply returns nothing:

```
DEBUG FilterBasedLdapUserSearch,http-apr-8630-exec-8:107 - Searching for user 'hwilliams', with user
search [ searchFilter: '(sAMAccountName={0})', searchBase: 'ou=users', scope: subtree,
searchTimeLimit: 0, derefLinkFlag: false ]
DEBUG SpringSecurityLdapTemplate,http-apr-8630-exec-8:211 - Searching for entry under DN 'o=mojo',
base = 'ou=users', filter = '(sAMAccountName={0})'
```

An invalid search filter is a filter that is malformed or cannot be loaded. You can also have a correctly formed search filter that returns incorrect results. See **3.10.2.2.5, "User Not Found By Valid Search Filter," on page 50** for more information.

Solution

If you suspect the search filter might be invalid, run the query in a third-party LDAP client and see if it returns any users. If the query does not return any users, correct the query to retrieve the users you want, then update your applicationContext-externalAuth.xml file with the correct query.

3.10.2.2.4 Failure to Bind the User

Problem

In some cases, the user is found, but the bind process fails. You might see an error in the logs such as the following:

```
Failed to bind as uid=myUser,OU=Users,OU=MTC-Users: org.springframework.ldap.AuthenticationException:
[LDAP: error code 49 - 80090308: LdapErr: DSID-0C09042A, comment: AcceptSecurityContext error, data
52e, v3839
```

Solution

A common reason for this is because of a mismatch in the DN format between what you specify in your search query versus what is acceptable for your specific version of LDAP. The precise solution depends on the implementation and configuration of your LDAP server. For more information, please consult documentation for your LDAP solution.

3.10.2.2.5 User Not Found By Valid Search Filter

Problem

A valid search filter that runs and returns results may not find all intended users. In this case you will see the same failed authentication message in the log as you would for an unauthorized user:

```
FilterBasedLdapUserSearch,http-apr-8630-exec-8:107 - Searching for user 'myUser', with user search [
```

```
searchFilter: '(uid={0})', searchBase: 'ou=users, o=org1', scope: subtree, searchTimeLimit: 0, dere-
fLinkFlag: false ]
DEBUG SpringSecurityLdapTemplate,http-apr-8630-exec-8:211 - Searching for entry under DN '', base =
'ou=users,o=org1', filter = '(uid={0})'
DEBUG ProviderManager,http-apr-8630-exec-8:152 - Authentication attempt using com.-
jaspersoft.jasperserver.multipleTenancy.MTDaoAuthenticationProvider
DEBUG SimpleUrlAuthenticationFailureHandler,http-apr-8630-exec-8:67 - Redirecting to
/login.html?error=1
DEBUG DefaultRedirectStrategy,http-apr-8630-exec-8:36 - Redirecting to '/jasperserver-pro/-
login.html?error=1'
```

Solution

This can happen for a number of causes. One of the most common is that the search filter is valid but is not returning the set of users you want. You could have misconfigured your query or you could be pointing to the wrong query in your applicationContext-externalAuth.xml file. Test the query shown in the log by running it in a third-party LDAP client and see if it returns the missing user.

To fix an incorrect query, look for the search filter in the applicationContext-externalAuth.xml file. Search filters are declared in the `ldapAuthenticationManager` bean, and the filter definition containing the query string is defined later in the same file. Check to make sure the search string is correct and that `ldapAuthenticationManager` is pointing to the correct search filter.

3.10.3 Login Page Not Loading

Another common symptom with multiple causes is failure to load the login page.

A blank page is shown instead.

3.10.3.1 Invalid Application Context File

Problem

If your application context file, applicationContext-externalAuth.xml, is not a valid XML file, the login page does not load. You may see a stack trace in the logs, specifying the invalid file:

```
Context initialization failed
org.springframework.beans.factory.xml.XmlBeanDefinitionStoreException: Line <lineNumber> in XML doc-
ument from ServletContext resource [/WEB-INF/applicationContext-externalAuth-LDAP-mt.xml] is invalid;
nested exception is org.xml.sax.SAXParseException; lineNumber: <lineNumber>; columnNumber: <colum-
nNumber>; The element type "property" must be terminated by the matching end-tag "</property>"
```

Solution

Usually the stack trace shows the name of the invalid file, the location in the file that is causing the problem, and the error that triggered the stack trace. The resolution depends on the error. In some cases, the location in the stack trace will not be the location of the root problem in the file.

In the example above, there is a missing ending tag for `property`. To fix this, add the tag at the location specified by `<lineNumber>` and `<columnNumber>`.

3.10.3.2 XML Special Characters in Role Names

Problem

Role names can't contain certain special characters, including the XML reserved characters <, >, &, ', ", and \. If you use a reserved character in a role name in your XML file, JasperReports Server attempts to interpret it as XML, which results in a stack trace. The precise error depends on the special character. For example, if you use an ampersand (&) in a role name, you see an error like this:

```
context initialization failed org.springframework.beans.factory.xml.XmlBeanDefinitionStoreException:
Line <lineNumber> in XML document from ServletContext resource [/WEB-INF/applicationContext-
externalAuth-LDAP-mt.xml] is invalid; nested exception is org.xml.sax.SAXParseException; lineNumber:
<lineNumber>; columnNumber: <columnNumber>; The entity name must immediately follow the '&' in the
entity reference. at
org.springframework.beans.factory.xml.XmlBeanDefinitionReader.doLoadBeanDefinitions
(XmlBeanDefinitionReader.java:397)
```

Solution

In general, it is safest to restrict role names to alpha-numeric characters. If extended characters are necessary for your naming convention, choose non-reserved characters.

3.10.3.3 Missing Bean Definition

Problem

Jasper is trying to read a bean definition that doesn't exist. In the error message, you see reference to the <type of bean>, which typically refers to the actual java class name for that bean definition. The bean name in the applicationContext-externalAuth.xml file may appear later in the error:

```
Caused by: org.springframework.beans.factory.BeanCreationException: Error creating bean with name
'<type of bean>' defined in ServletContext resource [/WEB-INF/applicationContext-externalAuth-LDAP-
mt.xml]: Cannot resolve reference to bean '<myBean>' while setting bean property 'userSearch'; nested
exception is org.springframework.beans.factory.NoSuchBeanDefinitionException: No bean named '<myBean>'
is defined
```

Solution

Check the following:

1. Make sure the bean is defined in your application context XML file.
2. Verify that the name of the bean is correct in your applicationContext-externalAuth.xml. If the bean name is spelled wrong, it won't be found.

3.10.3.4 Missing Java Class

Problem

JasperReports Server can't find a Java class referenced in the XML file. The stack trace shows the name of the class:

```
Caused by: org.springframework.beans.factory.CannotLoadBeanClassException: Cannot find class
```

```
[className] for bean with name 'ldapAuthenticationProvider' defined in ServletContext resource [/WEB-
INF/applicationContext-externalAuth-LDAP-mt.xml]; nested exception is java.lang.ClassNotFoundEx-
ception: <className>
```

Solution

Check the following:

1. Make sure the jar containing the specified class can be found in the classpath, (for example, \jasperserver-pro\WEB-INF\lib)
2. Verify that the name of the class is correct in your XML file. If the class name is spelled wrong, it won't be found as the name won't match, even if the class is present in a jar in the classpath.

3.10.4 Login Displays Security Check Page

Problem

JasperReports Server displays the j_spring_security_check page:

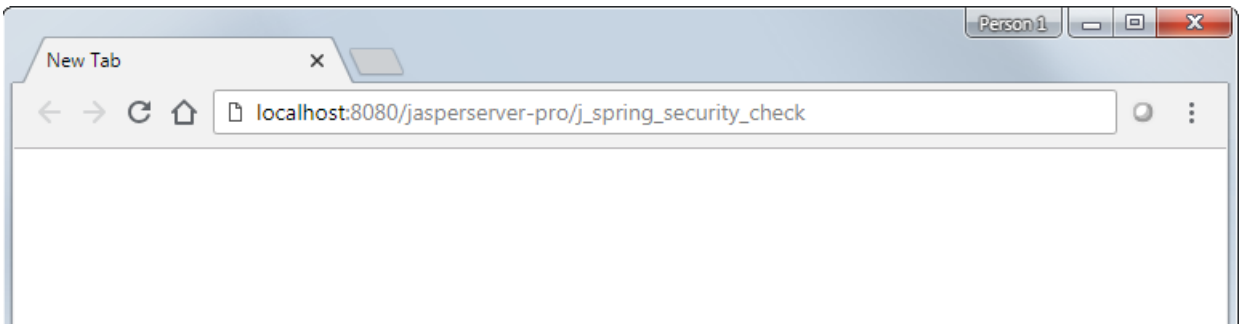

Figure 3-3 j_spring_security_check Page

In the jasperserver.log, look for `DefaultLdapAuthoritiesPopulator` and problems locating roles:

```
DefaultLdapAuthoritiesPopulator,http-apr-8630-exec-9:211 - Searching for roles for user 'guest01', DN
= 'cn=guest 01,cn=Users,dc=test,dc=com', with filter (objectClass=group) in search base 'DC=test,D-
DC=com'
SpringSecurityLdapTemplate,http-apr-8630-exec-9:150 - Using filter: (objectClass=group)
HttpSessionSecurityContextRepository,http-apr-8630-exec-9:304 - SecurityContext is empty or contents
are anonymous - context will not be stored in HttpSession.
SecurityContextPersistenceFilter,http-apr-8630-exec-9:97 - SecurityContextHolder now cleared, as
request processing completed
```

Solution

Errors with `DefaultLdapAuthoritiesPopulator` indicate that no roles could be found for this user. As part of the login process, the user is both authenticated and authorized. JasperReports Server uses LDAP and `DefaultLdapAuthoritiesPopulator` to determine which roles to assign to the user. A user needs at least ROLE_USER to log in. If no roles are assigned to the user, the login fails as above.

Ensure that you've configured role search correctly in the `JSDefaultLdapAuthoritiesPopulator` bean in your LDAP file. Make sure that it is using the correct branch in your LDAP.

3.11 Adding a Custom Processor

To create custom code to run on the server after the user has been authenticated, you can create a custom processor and add it to the processors list for the `externalUserProcessors` property of the `externalDataSynchronizer` bean. For example, you can add a processor that calls code to automatically create a user home folder for the authenticated user. See **7.3, "Creating a Custom Processor," on page 102**.

3.12 Restarting JasperReports Server

When you've configured all the beans in the appropriate files, restart JasperReports Server to make the changes take effect.

To test your configuration, navigate to the JasperReports Server login page. If the server and LDAP are configured correctly, you can log into JasperReports Server with credentials stored in your LDAP directory. Try several users with different roles or organizations to verify your configuration.

CHAPTER 4 CAS AUTHENTICATION

Central Authentication Service (CAS) is an open source, Java-based authentication server that supports for single sign-on (SSO) across web applications, including those running on different application servers. When a user requests a page from a CAS-enabled web application, the application redirects the user to the CAS server login page. Thereafter, logged-in users can navigate among all participating applications without needing to log in again. Each application communicates with the CAS server in the background to verify that the user is valid before providing access to its resources.

With the CAS protocol, the client application (such as JasperReports Server) never receives or transmits the user's password. As a result, the client application doesn't need to apply any encryption to protect passwords. However, unlike LDAP, CAS does not provide any user context, such as roles or organizations, that can be mapped to JasperReports Server. Instead, you can configure any organization and static roles that apply to each CAS-authenticated user, or pull user details from an external data source.

This chapter shows how JasperReports Server's default authentication mechanism using Spring Security can be configured to perform external authentication with CAS. The JasperReports Server deployment includes several sample files that provide the beans for CAS integration. The implementation of these beans is sufficient to enable CAS authentication but may not provide enough functionality in a complex deployment. Further customization of these beans is beyond the scope of this guide.

This chapter assumes that you're familiar with security concepts such as certificates, tokens, and cookies. The first section explains how to install a CAS server for testing. All examples refer to the test server and assume you're using the Apache Tomcat application server. When configuring JasperReports Server to use CAS in production, you must take into account any differences between application servers and the contents of certificates.

This chapter contains the following sections:

4.1 Overview of External CAS Authentication

This section describes how JasperReports Server integrates Spring Security to perform external CAS authentication, including SSO.

The following figure shows the general protocol during external CAS authentication:

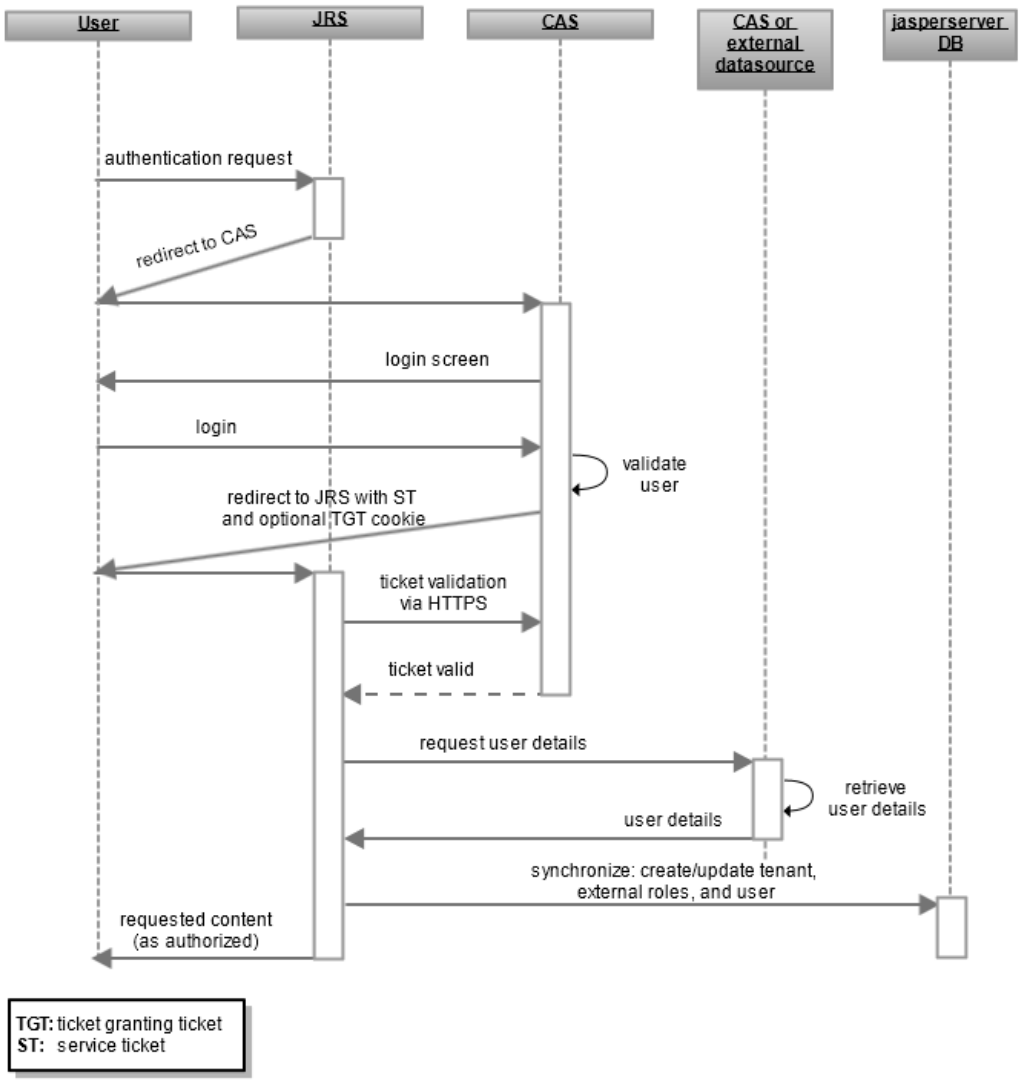

Figure 4-1 General Steps of External CAS Authentication

The overview in this section explains the major steps involved in the protocol between the CAS server and JasperReports Server, as well as the Spring Security beans involved. This chapter does not explain CAS proxies, but it does cover the Spring Security beans used to configure JasperReports Server's response to CAS proxies. The sample configuration has been verified with CAS 3.

See https://www.apereo.org/projects/cas and https://apereo.github.io/cas/4.2.x/index.html for more information.

The interaction between the user's browser, JasperReports Server, and the CAS server includes these steps:

1. An unauthenticated user requests any page in JasperReports Server.

   With SSO, JasperReports Server does not have a login page for users to bookmark. Instead, users can bookmark their home page or any page that allows it in JasperReports Server. As a result, every user goes through this step for every page they request from JasperReports Server, thereby securing every possible access to JasperReports Server.

   

   Internal users, such as jasperadmin or superuser, log in by going directly to the login page, for example, http://host1:8080/jasperserver[-pro]/login.html.

2. JasperReports Server detects that the user is not logged in and replies with a redirect to the CAS login page.

   The redirect URL contains the service parameter that tells CAS where to redirect the user after successful authentication. This URL activates JasperReports Server's user authentication. For example:

   http://host2:8443/cas/login?service=http://host1:8080/jasperserver[-pro]/j_spring_security_check.

   

   URLs may appear translated when viewed in browsers or in log files. For example, the previous URL might be written http://host2:8443/cas/login?service=http%3A%2F%2Fhost1:8080%2Fjasperserver%2Fj_spring_security_check.

3. The user's browser requests the CAS login page from the CAS server. If the user has logged in previously and has a Ticket Granting Ticket (TGT) cookie, the TGT is included in the request.
4. The CAS server attempts to authenticate the user's credentials.
   - If the CAS server detects the TGT cookie in the user's login request, CAS skips verification of the user credentials.
   - If there's no TGT cookie, CAS serves its login page to the user. When the user enters a username and password, CAS verifies them against its own data source.

   After the user authenticates correctly, CAS issues the user a service ticket (for example: http://host1:8080/jasperserver-[pro]/j_spring_security_check?ticket=ST-8670-123buTvFFjo980) and a TGT cookie if one is not present. The TGT cookie can be used for single sign-on with other applications that use the same CAS server.

   

   If the user has disabled cookies, the CAS login protocol still works, but single sign-on fails. When no cookie is detected on the user's browser, the user is prompted to log into the CAS server every time he accesses a client application.

5. CAS redirects the user to the JasperReports Server with the service ticket in the redirect URL.
6. The JasperReports Server authentication request filter is activated by the request for the j_spring_security_check resource. As part of the user authentication process, JasperReports Server establishes a secure HTTP connection (HTTPS) to CAS to validate the service ticket. If the ticket is valid, CAS replies with the username; otherwise, CAS responds with an error.
7. If an external data source is configured, JasperReports Server connects to the data source and requests the user organization and roles associated with the username returned by CAS. If the organization and roles are configured statically, this step is skipped.
8. JasperReports Server creates the principal object that establishes the user's session. The username, roles, and organization are also synchronized with the internal database, where the user account is marked as an external user. For more information, see **2.4.2, "Synchronization of External Users," on page 17**.
9. As with the default internal authorization, JasperReports Server now sends the requested content to the user.

Content sent to the user is subject to authorization. For example the home page has different options for administrators than for regular users, as determined by the roles of the user in the principal object.

When comparing these steps with those in **2.3, "Default Internal Authentication," on page 15**, you'll notice several significant differences:

- JasperReports Server must redirect to the CAS login page instead of its own.
- JasperReports Server must receive the service ticket as part of the security check.
- JasperReports Server must process the service ticket and communicate with the CAS server over HTTPS.
- The roles and organization ID in the user's principal object are mapped from the response to the request for user details.
- The internal database must be synchronized with any new information in the user's principal object.

4.2 CAS Server for Testing

This section describes how to set up a simple CAS server for testing purposes. If you have a CAS server you want to use, you can skip this section.

The CAS server is Java servlet built on the Spring Framework. Its primary responsibility is to authenticate users and grant access to CAS-enabled services by issuing and validating tickets. You can download the server from the following page: https://www.apereo.org/projects/cas/download-cas.

As described in the next section, the CAS validation service accepts only requests using a secure transport. This means you must have a valid certificate on your CAS server machine, and your CAS client (the JasperReports Server JVM) must be configured to trust that certificate. There are two important points to keep in mind:

- Test with the CAS server on a separate machine, not the localhost where JasperReports Server is installed. For this purpose, you can use a virtual machine.
- Most issues in configuring CAS are caused by the improper use of certificates. The single most common failure occurs when the hostname in the server's certificate doesn't match the actual hostname.

To create a certificate for the server you must use the Java `keytool` utility. Run the following command on the host of the CAS server:

```
keytool -genkey -alias tomcat -keyalg RSA -validity 365 -keystore <filename>
```

The utility prompts you for several pieces of information, two of which are critical. When prompted for your first and last name enter the hostname of the CAS server. When asked for the keystore password use `changeit` to match what Apache Tomcat uses by default.

After installation of the CAS server, configure the Apache Tomcat application server that's running the CAS server so it uses the certificate in the keystore created above. Modify $CATALINA_HOME/conf/server.xml, locate the commented section about setting up a secure HTTPS connector, and follow the instructions it contains. Restart the Tomcat server and test that it accepts HTTPS connections.

For further information about CAS, including deployment information, documentation, and community links, refer to the CAS website https://www.apereo.org/projects/cas. In particular, the page https://apereo.github.io/cas/4.2.x/installation/Troubleshooting-Guide.html can help you deploy your certificates.

CAS server is based on Spring Security, like JasperReports Server. In a production environment, you must replace the built-in authentication for testing with an external authority that validates your users when they log into CAS. As with JasperReports Server, you can configure CAS with a variety of external authorities to suit your needs, including LDAP. However, the external authority used by CAS may not be accessible to JasperReports Server.

Follow the CAS documentation to ensure you create a secure and robust configuration on your CAS server.

4.3 Configuring Java to Trust the CAS Certificate

The CAS protocol requires that the response to the service validation be established over HTTPS for security. This connection is established from the Java classes of Spring Security. So you need to configure the Java security system. Java security must trust the certificate it receives from the CAS server, otherwise it refuses to connect. This trust is based on two factors:

1. The host name in the certificate has to match the host name in the URL of the connection. Certain JVMs require hostnames as opposed to IP addresses, even if the IP addresses match. If you're using a CAS test server, see **4.2, "CAS Server for Testing ," on page 58** for instructions to create a certificate in a keystore.
2. You must tell Java to trust the signing certificate:
   a. On the CAS server, export your CAS certificate using the command line `keytool` utility. For example:

```
keytool -exportcert -alias cascert -file cascertfile.cer
```

   Enter the password to the keystore when prompted.
   b. Copy the CAS certificate you just exported to the JasperReports Server host and import it to the Java certificate store. For example, to import the certificate to the default truststore location, you might use the following:

```
keytool -importcert -alias cascert -keystore $JAVA_HOME/jre6/lib/security/cacerts -file cascertfile.cer
```

   Enter the password to the keystore when prompted.

A non-default cacerts location can be specified using the -Djavax.net.ssl.trustStore JVM parameter.

4.4 Configuring JasperReports Server for CAS Authentication

We provide sample files for configuring JasperReports Server for external CAS authentication. These files are specific to your edition (commercial or community). CAS sample files are located in the <js-install>/samples/externalAuth-sample-config directory of your JasperReports Server.

The sample files in your deployment that integrate Spring Security's default CAS implementation with JasperReports Server may include the following:

- sample-applicationContext-externalAuth-CAS.xml (community only): Sample file for integrating CAS with JasperReports Server, assigning the same roles to all users.
- sample-applicationContext-externalAuth-CAS-staticRoles.xml (community only): Sample file for integrating CAS with JasperReports Server, assigning additional roles to a list of administrative users.

- sample-applicationContext-externalAuth-CAS-db-mt.xml (commercial editions only): Sample file for integrating CAS with JasperReports Server with support for multiple organizations, retrieving role and organization data from a JDBC database.
- sample-applicationContext-externalAuth-CAS-LDAP-mt.xml (commercial editions only): Sample file for integrating CAS with JasperReports Server with support for multiple organizations, retrieving role and organization data from an LDAP server.

To use LDAP or an external database with CAS, configure the LDAP or database connection parameters in default_master.properties before installing JasperReports Server. You can set up encryption for the password to your LDAP server or database at that time. See the *JasperReports Server Security Guide* for more information.

To configure JasperReports Server to work with your implementation of CAS, select the sample configuration file you want, then modify and deploy it:

1. Make a copy of the CAS sample file and name it in the form applicationContext-<Name>.xml, for example, applicationContext-externalAuth-CAS-staticRoles-mt.xml.
2. Edit the file you created and configure the beans correctly for your deployment, as described in the following sections.
3. Place the modified applicationContext-externalAuth-CAS-staticRoles-mt.xml file in the <js-webapp>/WEB-INF directory.

<js-webapp> is the location of the JasperReports Server web application in your application server, or where you're modifying the configuration files, as explained in **2.1.2, "WEB-INF Directory Location," on page 12**. The rest of this chapter refers to file names alone.

4.5 Beans to Configure

The sample-applicationContext-externalAuth-CAS*.xml files contain the beans needed to enable and perform CAS authentication. This section summarizes the beans you need to modify to configure JasperReports Server to work with your CAS provider.

- `externalAuthProperties` – Configure this bean to specify the external authentication properties needed by the JasperReports Server API, including the HTTPS URL where tickets are validated on the CAS server and the login and logout URLs for external users. See **4.6.2, "Configuring externalAuthProperties," on page 61** for more information.
- `casServiceProperties` – Configure this bean to specify the Spring CAS API properties, such as the service parameter (the JasperReports Server URL to which CAS redirects after successful authentication) and whether to disregard the TGT (`sendRenew`).
- `externalUserSetupProcessor` – Configure this bean to specify how to synchronize external users and roles with the internal `jasperserver` database.
- `externalDataSource` (optional) – Configure this bean to specify an external data source for `casJDBCUserDetailsService` to query for user details. External details can be obtained from other data sources, such as LDAP. To enable an external data source, you must implement the `ExternalUserDetailsService` interface and substitute it in the `externalDataSynchronizer` bean.
- `casJDBCUserDetailsService` (optional) – Configure this bean with the SQL queries to extract external user information from the data source specified in the `externalDataSource` bean.

4.6 Setting CAS Authentication Properties

Configure the following beans to set up the properties related to CAS authentication:

- `casServiceProperties` – Configure this bean to set the properties related to Spring's CAS service.
- `externalAuthProperties` – Configure this bean to set properties specific to Jaspersoft's implementation of CAS.

4.6.1 Configuring casServiceProperties

The `casServiceProperties` bean implements the Spring `JSCASServiceProperties` bean and stores the properties related to Spring's CAS: service and sendRenew properties. Configure the following bean to set up the security check locations:

- `casServiceProperties` – This bean has the following properties:
  - `service` property – Configure this property with the JasperReports Server URL that receives and processes tickets for HTTP connections. Make sure to include the `j_spring_security_check` pattern at the end of the URL; this pattern activates user authentication (service ticket validation) in the Spring Security filter chain.
  - `sendRenew`: When the user is redirected to CAS for authentication, the `sendRenew` property (`true|false`) determines whether or not to disregard TGT (default).

For example, to enable the security check on your localhost, you would set the pattern as in this example:

```
<bean id="casServiceProperties" class="com.jaspersoft.jasperserver.api.security.
      externalAuth.wrappers.spring.cas.JSCASServiceProperties">
  <property name="service" value="http://localhost:8080/jasperserver-pro/
        j_spring_security_check"/>
  <property name="sendRenew" value="false"/>
</bean>
```

4.6.2 Configuring externalAuthProperties

The `externalAuthProperties` bean stores properties specific to Jaspersoft's external authentication API. For CAS authentication, configure the `externalAuthProperties` bean to specify the following URLs for communication with the CAS server:

- `ssoServerLocation` property — Specifies the HTTPS URL where tickets are validated on the CAS server.
- `externalLoginUrl` property — Specifies the login URL for external users. Can be set relative to `ssoServerLocation`.
- `logoutUrl` property — Specifies the logout URL for external users. Can be set relative to `ssoServerLocation`.

The following example shows how you might set these beans:

```
<bean id="externalAuthProperties" class="com.jaspersoft.jasperserver.api.security.
      externalAuth.ExternalAuthProperties">
  <property name="externalLoginUrl" value="#ssoServerLocation#/login"/>
  <property name="logoutUrl" value="#ssoServerLocation#/logout"/>
  <property name="ssoServerLocation" value="https://casserver:8443/cas"/>
</bean>
```

4.7 Mapping the User Roles

You can define user roles in one of the following ways:

- Define user roles statically – Define static roles for administrative and non-administrative users using the `externalUserSetupProcessor` or `mtExternalUserSetupProcessor` bean. This is the most common option. See **4.7.1, "Defining Static Roles," on page 62**.
- Retrieve user roles from an external data source – See **4.7.2, "Retrieving User Roles from an External Data Source," on page 63**.

4.7.1 Defining Static Roles

If you're mapping all your external users to a single organization, you can assign static roles to users. This lets you specify a list of administrative users and roles, and a list of roles for non-administrative users. To define static roles, use the `externalUserSetupProcessor` or `mtExternalUserSetupProcessor` bean. To set up static roles, locate the version of the bean used in your sample file and configure the following properties:

- `adminUserNames` property — A list of usernames granted internal administrator privileges in JasperReports Server. The username values must exactly match the usernames authenticated and returned by the external authority.
- `defaultAdminRoles` property — A list of JasperReports Server internal roles. These are assigned to every user in the list of administrators.
- `defaultInternalRoles` property — A list of JasperReports Server roles assigned to every user not in the list of administrators.

The following example shows how to use the `mtExternalUserSetupProcessor` bean to define static roles. The configuration for `externalUserSetupProcessor` is similar:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
    externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
       ...
  <property name="adminUsernames">
    <list>
      <value>myorgadmin</value>
    </list>
  </property>

  <property name="defaultAdminRoles">
    <list>
      <value>ROLE_USER</value>
      <value>ROLE_ADMINISTRATOR</value>
    </list>
  </property>

  <property name="defaultInternalRoles">
    <list>
      <value>ROLE_USER</value>
    </list>
  </property>

...
```

4.7.2 Retrieving User Roles from an External Data Source

If the static configuration available with CAS is insufficient, you can import external user information, like roles and organization ID, from an external data source. Imported roles are stored in the internal `jasperserver` database (synchronization); they can be mapped to internal JasperReports Server roles or created as new external roles.

To retrieve external roles, the `externalUserDetailsService` property in the `externalDataSynchronizer` bean needs to point to an `ExternalUserDetailsService` implementation. The sample file includes `CasJDBCUserDetailsService`, which connects to an external MySQL database. `externalUserDetailsService` then makes the external data available to `externalUserSetupProcessor` or `mtExternalUserSetupProcessor`, which is responsible for mapping the external information and synchronizing it with the database.

The `casJDBCUserDetailsService` bean is configured with the following properties:

- `dataSource` property – Points to external database to query user details
- `usersByUsernameQuery` property – SQL query returning a list of user properties for the user name to be processed by `externalUserSetupProcessor` or `mtExternalUserSetupProcessor`. The result is returned as a map where keys are the column names in the query.
- `authoritiesByUsernameQuery` property – SQL query returning a list of user roles for the user name.

The `externalUserSetupProcessor` or `mtExternalUserSetupProcessor` bean has the following properties:

- `defaultInternalRoles` property – A list of internal roles assigned to the external user by default.
- `organizationRoleMap` property – A list of key/value pairs that map external role names to internal ones. For a commercial JasperReports Server deployment, you need to choose the level at which the role is assigned:
  - To map to an internal role at the organization level, append |* to the name of the internal role, for example, ROLE_EXTERNAL_USER|*. Roles mapped at the organization level do not have administrative privileges.
  - To map to an internal role at the system (null) level, do not modify the internal role name, for example, ROLE_EXTERNAL_ADMINISTRATOR. Roles at the system level are usually reserved for special users like the system administrator and allow access to the repository folders of all organizations.

The following shows how you might configure the `externalUserSetupProcessor` bean:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
    externalAuth.processors.MTExternalUserSetupProcessor"  parent="abstractExternalProcessor">
<property name="organizationRoleMap">
   <map>
    <!-- Example of mapping customer roles to JRS roles -->
    <entry>
      <key>
        <value>ROLE_ADMIN_EXTERNAL_ORGANIZATION</value>
      </key>
      <!-- JRS role that the <key> external role is mapped to-->
      <value>ROLE_ADMINISTRATOR</value>
    </entry>
  </map>
</property>
```

4.7.3 Setting Default Roles

You can assign roles to all users using the `defaultInternalRoles` property of `externalUserSetupProcessor` or `mtExternalUserSetupProcessor`. The following example shows how to use this property in `externalUserSetupProcessor` to assign `ROLE_USER` to all users, in addition to the roles assigned by mapping:

```
<property name="defaultInternalRoles">
  <list>
    <value>ROLE_USER</value>
  </list>
</property>
```

4.7.4 Avoiding Role Collisions

If an external role has the same name as an internal role at the same organization level, JasperReports Server adds a suffix such as _EXT to the external role name to avoid collisions. For example, a user with the externally defined role ROLE_ADMINISTRATOR is assigned the role ROLE_ADMINISTRATOR_EXT in the JasperReports Server database. This ensures that internal administrator accounts like jasperadmin and superuser can still log in as internal administrators with the associated permissions.

You can set the extension in the `conflictingExternalInternalRoleNameSuffix` property in the `externalUserSetupProcessor` or `mtExternalUserSetupProcessor` bean. If the property doesn't appear in the bean, the extension is still implemented but defaults to _EXT. The following example shows how to configure this property:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
    externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
  <property name="conflictingExternalInternalRoleNameSuffix"
        value="_EXTERNAL"/>
  <property name="organizationRoleMap">
      ...
     <!-- Example of mapping customer roles to JRS roles -->
      ...
  </property>
```

4.7.5 Restricting the Mapping to Whitelisted Roles

You may not want every role in your external authority to appear as a role in JasperReports Server. Use the `permittedRolesRegex` property of the `externalUserSetupProcessor` bean or `mtExternalUserSetupProcessor` bean to specify which external roles become roles in JasperReports Server. You can use regular expressions to specify multiple roles that match the expression.

For example, to restrict the roles you create in JasperReports Server to roles that begin with JRS_ or EXT_ in your external authority, you would configure `permittedRolesRegex` in a way similar to the following:

```
        <property name="permittedRolesRegex">
            <list>
                <value>JRS_.*</value>
                <value>EXT_.*</value>
            </list>
        </property>
```

To allow all roles, use .* or comment out the property. If the property is omitted, all roles in the external authority are synchronized with roles in JasperReports Server.

4.7.6 Supporting Additional Characters in Role Names

The default mapping from attributes in your external authentication server to roles in JasperReports Server supports only alphanumeric characters and underscores. If a role in your external authority contains unsupported characters, each sequence of unsupported characters is replaced with a single underscore. For example, ROLE$-DEMO)EXT maps to ROLE_DEMO_EXT.

You can extend the supported character set by modifying the `permittedExternalRoleNameRegex` property of the `externalUserSetupProcessor` bean or `mtExternalUserSetupProcessor` bean. Check the sample configuration file for your deployment to determine which bean to modify.

The default value of the `permittedExternalRoleNameRegex` property is the regular expression [A-Za-z0-9_]+. Edit this expression to add supported characters. For example, the following syntax allows alphanumeric characters, underscores, and the Cyrillic letter Я (Unicode 042F):

```
<bean id="mtExternalUserSetupProcessor"  class="com.jaspersoft.jasperserver.api.security.
        externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
 <property name="userAuthorityService">
   <ref bean="${bean.internalUserAuthorityService}"/>
 </property>
  .....
 <property name="permittedExternalRoleNameRegex"
     value="[A-Za-z0-9_\u042F]+">
</bean>
```

Do not allow the following in role names: spaces, periods or |, [ ], `, ", ', ~, !, #, $, %, ^, &, [,], *, +, =, ;, :, ?, <, >, }, {, ), (, ], [, /, or \. Adding these characters in the `permittedExternalRoleNameRegex` property may cause unexpected behavior, such as the inability to delete or edit roles containing those characters.

4.8 Setting the User Organization

Organizations are a feature of JasperReports Server commercial editions. Skip this section if you have JasperReports Server community edition.

Spring's default CAS configuration supports only user authentication. However, you can extend this to set organizations in one of two ways:

- Extract organization data with an additional technology, such as LDAP or a JDBC database. See **4.8.1, "Mapping to Multiple Organizations," on page 66**.
- Use the `defaultOrganization` property of the `externalTenantSetupProcessor` bean to set a single organization assigned to all external users. See **4.8.2, "Mapping to a Single Organization," on page 67**.

4.8.1 Mapping to Multiple Organizations

To assign your external CAS users to multiple organizations in JasperReports Server, you need an additional technology like LDAP or JDBC to supply the user's organization data. We provide sample files that show how to extract the organization data from a third-party technology and integrate it with CAS authentication. These files are described briefly; an in-depth discussion is beyond the scope of this guide.

4.8.1.1 Setting Multiple Organizations Using LDAP

You can configure your connection to the LDAP server in one of two ways:

- Configure the connection during installation of JasperReports Server by configuring the `external.ldapUrl`, `external.ldapDn`, and/or `external.ldapPassword` properties in default_master.properties. You have the option to encrypt any of the LDAP connection parameters. This is the preferred method for setting the LDAP connection parameters. See the *JasperReports Server Security Guide* for more information.
- If you have an existing JasperReports Server and can't reinstall it for some reason, you can configure the connection properties directly in your sample-applicationContext-externalAuth-CAS-mt.xml file. In this case, the properties, including the password, can't be encrypted. See **3.4, "Setting the LDAP Connection Parameters," on page 29**for more information.

The following file gives an example of how to assign users to multiple organizations by integrating CAS with LDAP:

<js-install>/samples/externalAuth-sample-config/sample-applicationContext-externalAuth-CAS-LDAP-mt.xml

This sample uses the `ldapExternalTenantProcessor` bean to extract an organization hierarchy from the user's distinguished name. For more information about the `ldapExternalTenantProcessor` bean, see **Mapping to Multiple Organizations** in **Chapter 3, "LDAP Authentication," on page 25**.

4.8.1.2 Setting Multiple Organizations Using JDBC

You can configure your connection to the database in one of two ways:

- Configure the connection during installation of JasperReports Server by configuring the `external.jdbc.driverClassName`, `external.jdbc.url`, `external.jdbc.username`, and/or `external.jdbc.Password` properties in default_master.properties. You have the option to encrypt any of the LDAP connection parameters. This is the preferred method for setting the database connection parameters. See the *JasperReports Server Security Guide* for more information.
- If you have an existing JasperReports Server and can't reinstall it for some reason, you can configure the connection properties directly in your sample-applicationContext-externalAuth-CAS-db-mt.xml file. In this case, the properties, including the password, can't be encrypted. See **5.4, "Setting the Database Connection Parameters," on page 72** for more information.

The following file gives an example of how to assign users to multiple organizations by integrating CAS with a JDBC database:

<js-install>/samples/externalAuth-sample-config/sample-applicationContext-externalAuth-CAS-db-mt.xml

This sample uses the `detailsQuery` property of the `casJDBCUserDetailsService` bean to extract `tenantId` from an external database using an appropriate SQL query. Note that the `tenantId` column name has to be returned by the SQL query in order for `externalTenantSetupProcessor` to catch and process it correctly. In cases where the external database column has a different name, cast the column name as `tenantId`, as in the following example:

```
SELECT organizationId AS tenantId from org_table
```

4.8.2 Mapping to a Single Organization

If you have multiple organizations in your deployment, you can use the `externalTenantSetupProcessor` bean to specify a single organization assigned to all external users. To do this, set `externalTenantSetupProcessor`'s `defaultOrganization` property to the organization ID of the selected organization. If an organization with that ID already exists, all external users are assigned to that organization. If the organization does not exist, it's created when the first external user authenticates correctly.

When specifying the `defaultOrganization` value, the organization ID must not contain the following characters: |, &, *, ?, <, >, /, \, ~, !, #, $, %, ^, [, or].

The following example shows how to configure `externalTenantSetupProcessor` to assign all users to organization_1:

```
<bean id="externalTenantSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.
    security.externalAuth.processors.MTExternalTenantSetupProcessor"
    parent="abstractExternalProcessor">
  <property name="multiTenancyService">
    <ref bean="internalMultiTenancyService"/>
  </property>
  <property name="defaultOrganization" value="organization_1"/>
</bean>
```

Do not specify a null value for the `defaultOrganization` property. The null organization ID is usually reserved for special users like the system administrator and allows access to the repository folder of all other organizations.

Organizations created during external user login have an administrator with the default password. For security reasons, you should change the default password of any organization admin. See **2.4.4, "Initialization of JasperReports Server for External Users," on page 19** for a process to initialize JasperReports Server, including organization administrators, before going into production with external authentication.

4.9 Adding a Custom Processor

To create custom code to run on the server after the user has been authenticated, you can create a custom processor and add it to the processors list for the `externalUserProcessors` property of the `externalDataSynchronizer` bean. For example, you can add a processor that calls code to automatically create a user home folder for the authenticated user. See **7.3, "Creating a Custom Processor," on page 102**.

4.10 Customizing the JasperReports Server Interface for CAS

When a user who's logged into JasperReports Server through CAS clicks the **Log Out** link, they'll see the JasperReports Server login page, instead of the CAS login page.

This is due to the way the Spring framework integrates with CAS. Because all login requests are redirected to CAS, JasperReports Server doesn't know the logged-in or logged-out state of the user. Instead, JasperReports Server treats all requests as if they're already logged in. If the CAS check doesn't go through, JasperReports Server redirects back to the CAS login page. But if the user clicks **Log Out**, this redirection doesn't happen.

To avoid confusion, you can remove or hide the **Log Out** link using the techniques in “Customizing the User Interface” in the *JasperReports Server Ultimate Guide*.

4.11 Restarting JasperReports Server

When you've configured all the beans in the appropriate files, restart JasperReports Server. Because of the modification to the application server configuration in **4.3, “Configuring Java to Trust the CAS Certificate,” on page 59**, you must restart the application server, which also restarts JasperReports Server.

Instead of navigating to the JasperReports Server login page, go directly to another page in JasperReports Server, for example your home page:

http://host1:8080/jasperserver/flow.html?_flowId=homeFlow

If JasperReports Server, CAS, and your certificates are configured correctly, you'll be prompted to log into the CAS server; when you do, you'll be redirected to the JasperReports Server home page for the logged-in user.

CHAPTER 5 EXTERNAL DATABASE AUTHENTICATION

This chapter shows how to configure JasperReports Server to perform external authentication and authorization using tables in an external database. To help you with this, the JasperReports Server deployment includes a sample file, sample-applicationContext-externalAuth-db-mt.xml, that serves as a template for external database authentication. Customizing this template should satisfy the requirements of most external database authentication cases.

This chapter assumes you have some familiarity with Spring Security filter chains. Examples in this chapter assume you are running JasperReports Server on an application server, such as Apache Tomcat, and an external SQL database.

This chapter contains the following sections:

5.1 Overview of External Database Authentication

This section describes how JasperReports Server integrates Spring Security to authenticate users against an external database.

The following figure shows the general steps for external database authentication:

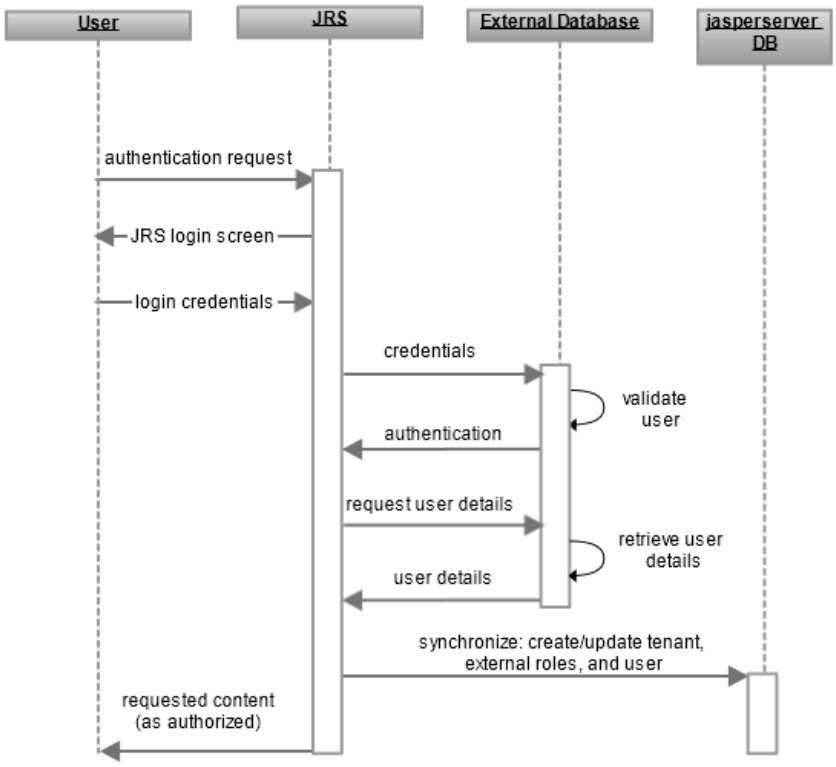

Figure 5-1 General Steps of External Database Authentication

The interaction between the user's browser, JasperReports Server, and the external database includes these steps:

1. An unauthenticated user requests any page in JasperReports Server.
2. JasperReports Server detects that the user is not logged in and replies with the JasperReports Server login page.
3. The user enters their credentials.
4. The JasperReports Server establishes a connection to the database server to verify the user's credentials.
5. If the user submitted a valid username and password, the database server authenticates the user to JasperReports Server.
6. JasperReports Server requests user details from the database server using a database query specified in the configuration file.
7. The database server returns the requested details.

   JasperReports Server maps the username to a predefined set of roles and an organization ID. The username, roles, and organization are also synchronized with the internal database, where the user account is marked as an external user. (Community editions don't need to synchronize the organization.) For more information, see **2.4.2, "Synchronization of External Users," on page 17**.
8. JasperReports Server now sends the requested content to the user. Content that is sent to the user is subject to authorization. For example, the home page has different options for administrators than for regular users.

The only difference between these steps and those in **2.3, "Default Internal Authentication," on page 15**, is that instead of searching for the user in the `jasperserver` internal database, JasperReports Server makes a JDBC call to the external database and then synchronizes the user details.

5.2 Configuring JasperReports Server for External Database Authentication

To use an external database with your JasperReports Server, configure the database connection parameters in default_master.properties before installing JasperReports Server. You can set up encryption for the password for your database at that time. See the *JasperReports Server Security Guide* for more information.

A sample file for configuring JasperReports Server for external database authentication is included in the JasperReports Server distribution. To configure JasperReports Server to work with your external database, modify and deploy the sample configuration file as follows:

1. Make a copy of the external database sample file:
   <js-install>/samples/externalAuth-sample-config/sample-applicationContext-externalAuth-db-mt.xml
   and name it applicationContext-externalAuth-db-mt.xml.
2. Edit the file you created and configure the beans correctly for your deployment, as described in the following sections.
3. Place the modified applicationContext-externalAuth-db-mt.xml file in the <js-webapp>/WEB-INF directory.

<js-webapp> is the location of the JasperReports Server web application in your application server, or where you're modifying the configuration files. The rest of this chapter refers to file names alone.

5.3 Beans to Configure

The sample-applicationContext-externalAuth-db-mt.xml file contains the beans needed to enable and perform authentication to an external database. This section summarizes the most important beans in this file, including the beans you need to configure to make the samples work with your deployment.

- `proxyAuthenticationProcessingFilter` – Filter bean that enables external authentication of web application requests. When `proxyAuthenticationProcessingFilter` is present in the application context, that is, when it appears in an applicationContext-<customName>.xml file in the <js-webapp>/WEB-INF directory, the Spring Security filter chain processes the authentication via the proxy definitions instead of the default internal filter found in applicationContext-security-web.xml. You don't need to configure this bean.
- `dbAuthenticationManager` – Lists the available authentication providers. The providers in the list are invoked in the order they appear in the configuration file until one of them authenticates the user. When one of the providers successfully authenticates the user, the rest of the providers are skipped. The final provider in the list, `${bean.daoAuthenticationProvider}` authenticates to the `jasperserver` internal database. You can customize authentication by adding more providers to this bean.
- `externalDaoAuthenticationProvider` – Custom authentication provider for external database authentication.
- `externalUserTenantDetailsService` – Configure this bean to define the query necessary to retrieve user, role, and organization information from the external database, as described in **"Configuring User Authentication and Authorization via Database Queries" on page 74**.

- `passwordValidator` – Bean that specifies a key for encoding user passwords. First, import the same cryptographic key that is used for passwords in your database. Then configure this bean with the key's alias and password, as described in **"Setting the Password Encryption " on page 75**.
- `externalDataSynchronizer` – Bean whose class creates a mirror image of the external user in the internal jasperserver database. The sample includes the following processors:
  - `externalTenantSetupProcessor` – Bean that sets up external organizations in the `jasperserver` database. For multi-organization JasperReports Server deployments, configure this bean to specify the mapping between fields and field values retrieved from the database and JasperReports Server organizations, as described in **"Setting the User Organization" on page 79**.
  - `mtExternalUserSetupProcessor` – Bean that creates and configures the internal user corresponding to a successfully authenticated external user. Configure this bean to specify the default internal role given to the external users in JasperReports Server and to map external user roles to internal JasperReports Server roles if needed, as described in **5.7, "Mapping User Roles," on page 76**. Optionally, you can assign administrative roles to specific users.
  - `externalUserFolderProcessor` – Bean that creates a user folder as an example of additional post-authentication processing. Post-authentication processing is described in **"Authentication Based on Request" on page 103**.
- `externalDataSource` – Configure this bean with the JDBC connection information to the external database to which users are authenticated, as described in **"Setting the Database Connection Parameters" on page 72**.
- `externalAuthProperties` – This bean stores properties necessary to configure JasperReports Server for external authentication. For multi-organization deployments of JasperReports Server, configure this bean to require an organization ID on the login form, as described in **"Configuring the Login Page for a Single-Organization Deployment" on page 82**

5.4 Setting the Database Connection Parameters

You can configure your connection to the database in one of two ways:

- Configure the connection during installation of JasperReports Server by configuring the `external.jdbcDriverClass`, `external.jdbcUrl`, `external.dbUsername`, and/or `external.dbPassword` properties before installation or upgrade. You can choose to encrypt any of the database connection parameters. This is the preferred method for setting the database connection parameters. See the *JasperReports Server Security Guide* for more information.
- If you have a JasperReports Server, you can configure the connection properties directly in your sample-applicationContext-externalAuth-db-mt.xml file. In this case, the properties, including the password, can't be encrypted.

5.4.1 Setting Database Connection Parameters in default_master.properties

The preferred method for setting the database connection parameters is to configure the `external.jdbcDriverClass`, `external.jdbcUrl`, `external.dbUsername` and `dbPassword` properties in the default_master.properties file before installation or upgrade. In JasperReports Server 5.6 and later, the default configuration of the `externalDataSource` bean in sample-applicationContext-externalAuth-db-mt.xml uses context properties for the database connection properties:

```
<bean id="externalDataSource" class="com.jaspersoft.jasperserver.api.security.
    externalAuth.wrappers.spring.jdbc.JSDriverManagerDataSource">
  <property name="driverClassName" value="${external.jdbc.driverClassName}"/>
  <property name="url" value="${external.jdbc.url}"/>
  <property name="username" value="${external.jdbc.username}"/>
  <property name="password" value="${external.jdbc.password}"/>
</bean>
```

To configure these properties using default_master.properties, follow these steps:

1. Open default_master.properties in a text editor.
2. Locate and set the following properties for your LDAP server:
   - `external.jdbcDriverClass` property – The name of the JDBC driver class for your database. Make sure the driver jar library is available on the classpath; for example, you can place the jar in the lib directory of your application server or in the <js-webapp>/lib directory.
   - `external.jdbcUrl` property – The JDBC URL for your database server, including the hostname, port, and database you want to access.
   - `external.dbUsername` property – The username of your database administrator.
   - `external.dbPassword` property — The password of your database administrator.
3. You can choose to encrypt any of the LDAP connection parameters.

The following example shows the syntax of the properties in the default_master.properties file:

```
external.jdbcDriverClass=com.mysql.jdbc.Driver
external.jdbcUrl=jdbc:mysql://127.0.0.1:3306/external_sso_test
external.dbUsername=username
external.dbPassword=password
```

To encrypt the password property, also set the following:

```
encrypt=true
propsToEncrypt=dbPassword,external.dbPassword
```

See the *JasperReports Server Security Guide* for more information on encrypting passwords using buildomatic.

5.4.2 Setting Database Connection Parameters Manually

If you configured your database connection during JasperReports Server installation, don't set the parameters using `externalDataSource`. You can verify whether the parameters are set by looking at the default_master.properties file.

To set the connection parameters for the external database server directly in the application context file, configure the `externalDataSource` helper bean as follows:

1. In sample-applicationContext-externalAuth-db-mt.xml, locate the `externalDataSource` bean.

2. Specify the following information:
   - `driverClassName` property – The name of the JDBC driver class for your database. Make sure the driver jar library is available on the classpath. For example, you can place the jar in the lib directory of your application server or in the <js-webapp>/lib directory.
   - `url` property – The JDBC URL for your database server, including the hostname, port, and database you want to access.
   - `username` property – The username of your database administrator.
   - `password` property – The password of your database administrator.

The following is an example of the connection information for a MySQL database:

```
<bean id="externalDataSource" class="com.jaspersoft.jasperserver.api.security.
      externalAuth.wrappers.spring.jdbc.JSDriverManagerDataSource">
  <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
  <property name="url" value="jdbc:mysql://127.0.0.1:3306/external_sso_test"/>
  <property name="username" value="username"/>
  <property name="password" value="password"/>
</bean>
```

5.5 Configuring User Authentication and Authorization via Database Queries

To authenticate a user, JasperReports Server first queries the external database to retrieve the user by username and organization. After authentication, a second query is executed to retrieve user roles. These queries are configured in the `externalUserTenantDetailsService` bean. The results are used to map users, roles, and organizations.

- `externalUserTenantDetailsService` – Configure this bean to define the queries needed to retrieve user, organization, and roles from the external database. This bean has the following properties:
  - `dataSource` property – References the `externalDataSource` bean, which configures the JDBC connection to the database. The `externalDataSource` bean is defined later in the file.
  - `usersByUsernameAndTenantNameQuery` – Property that takes as input a single `username` parameter and returns a username, encrypted password, and an organization ID. Configure this property with a database query that retrieves the information that authenticates the user, that is, username, encrypted password, and organization.
  - `authoritiesByUsernameQuery` – Property that takes as input a single `username` parameter and returns one or more records of `username`, `rolename` tuples. Configure this property with a database query that retrieves the user and roles from the external database.

The following example shows how to set up the `externalUserTenantDetailsService` bean queries:

```
<bean id="externalUserTenantDetailsService" class="com.jaspersoft.jasperserver.
        multipleTenancy.security.externalAuth.db.MTExternalJDBCUserDetailsService">
    <property name="dataSource" ref="externalDataSource"/>
    <property name="usersByUsernameAndTenantNameQuery" value="SELECT u.username,
      u.password, t.tenantId FROM jiuser u, jitenant t WHERE u.tenantId = t.id and
      username = ?"/>
    <property name="authoritiesByUsernameQuery" value="SELECT u.username, r.rolename
      FROM jiuser u, jiuserrole ur, jirole r WHERE u.id = ur.userId and ur.roleId=r.id and
      u.username = ?"/>
    <property name="multiTenancyConfiguration"><ref bean="multiTenancyConfiguration"/></property>
</bean>
```

Note that the semantics, order, and number of the columns in each tuple is fixed. Changing the order or number of the columns requires customization, which is beyond the scope of this manual.

5.6 Setting the Password Encryption

JasperReports Server receives the credentials from the user in the login request as plaintext. If your database stores encrypted user passwords, you must share the encryption key with JasperReports Server. The server can then encrypt the password from the user request and compare it to the password from the external database.

As of JasperReports Server 7.5, the server uses a central keystore (.jrsks file) to securely hold all the encryption keys it needs. The Java Cryptography Architecture (JCA) defines the ciphers and the protocols for the keys and the keystore. The encryption used in your external database must be compatible with the JCA and stored in a compatible key. Then you can import that key so it is shared by JasperReports Server. For example, if you have your database's password encryption key in a keystore file, run the following commands as the system user who installed the server:

```
cd <js-install>/buildomatic
js-import.sh --input-key --keystore <path>/mykeystore --storepass password
             --keyalias mydbkey --keypass mydbkeypw
```

The key will be copied to the server's keystore and keep the same properties, including alias and password. You can also import keys as hexadecimal values if necessary. The following command creates a new key with the given algorithm, alias, and password:

```
js-import.sh --input-key "0x59 0xe3 0xd9 0xce 0x7f 0x34 0xab 0x27 0xb8 0xdf 0xc3 0x7e
0x01 0xab 0x4d 0x6c" --keyalg AES --keyalias mydbkey --keypass mydbkeypw
```

In the case where your external database is new and not yet provisioned with users, it will need a key to encrypt passwords. In that case, JasperReports Server can generate the key, store it in the keystore, and you can export it to use in your database. The following commands create a new random key in the keystore and export the same key for use externally:

```
js-import.sh --input-key --genkey --keyalg AES --keysize 128 --keyalias mydbkey
             --keypass mydbkeypw
js-export.sh --destkeystore mystore --deststorepass storepw --keyalias mydbkey
             --keypass mydbkeypw
```

For more information about the keystore and exporting keys, see the *JasperReports Server Security Guide*

Once the key is in the server's keystore, configure the `passwordValidator` bean in the applicationContext-externalAuth-db-mt.xml file. Set the bean's property values to match those of the key you have imported. The following example shows how to configure the bean with the keys imported above:

```
<bean id="passwordValidator" class="com.jaspersoft.jasperserver.api.common.crypto.CipherFactory" lazy-
init="false">
    <property name="cipherClass" value-
e="com.jaspersoft.jasperserver.api.metadata.common.service.impl.PasswordValidator"/>
    <property name="transformation" value="AES/CBC/PKCS5Padding"/>
    <property name="blockSize" value="16"/>
    <property name="keyAlgorithm" value="AES"/>
    <property name="keySize" value="128"/>
```

```
    <property name="keyAliasProp" value="mydbkey.keyalias"/>
    <property name="keyPassProp" value="mydbkey.keypass"/>
</bean>
```

By default, the JasperReports Server keystore supports AES and DES keys. If your database uses a different encryption algorithm, you can configure your own password encoder using the Spring implementations of the `PasswordEncoder` interface. This is an advanced configuration that is beyond the scope of this guide.

5.7 Mapping User Roles

The roles an external user has in JasperReports Server are imported from the external data source and stored in the internal JasperReports Server database. External roles can be reflected as new external roles in JasperReports Server or they can be mapped to internal roles.

5.7.1 Retrieving Roles from the External Database

To configure the retrieval and mapping for user roles in sample-applicationContext-externalAuth-db-mt.xml file, first make sure you've set up the `externalUserTenantDetailsService` bean as described in **5.5, "Configuring User Authentication and Authorization via Database Queries," on page 74**. Then configure `externalUserSetupProcessor` to map the external information to roles in the JasperReports Server as follows:

- `defaultInternalRoles` property – A list of internal roles assigned to the external user by default.
- `organizationRoleMap` property – A list of key/value pairs that maps external role names to internal ones. For commercial JasperReports Server deployments, you need to choose the level at which the role is assigned:
  - To map to an internal role at the organization level, append |* to the name of the internal role, for example, ROLE_EXTERNAL_USER|*. Roles mapped at the organization level don't have administrative privileges.
  - To map to an internal role at the system (null) level, don't modify the internal role name, for example, ROLE_EXTERNAL_ADMINISTRATOR. Roles at the system level are usually reserved for special users like the system administrator and allow access to the repository folder of all other organizations.

The following example shows how to configure the `organizationRoleMap` property:

```
<property name="organizationRoleMap">
  <map>
  <!-- Example of mapping customer roles to JRS roles -->
    <entry>
      <key>
        <value>ROLE_ADMIN_EXTERNAL_ORGANIZATION</value>
      </key>
      <!-- JRS role that the <key> external role is mapped to-->
      <value>ROLE_ADMINISTRATOR|*</value>
    </entry>
  </map>
</property>
```

5.7.2 Defining Static Roles

If you're mapping all your external users to a single organization, you can assign static roles to users. This lets you specify a list of administrative users and roles, and a list of roles for non-administrative users. To define static roles, use the `externalUserSetupProcessor` or `mtExternalUserSetupProcessor` bean. To set up static roles, locate the version of the bean used in your sample file and configure the following properties:

- `adminUserNames` property — A list of usernames granted internal administrator privileges in JasperReports Server. The username values must exactly match the usernames authenticated and returned by the external authority.
- `defaultAdminRoles` property — A list of JasperReports Server internal roles. These are assigned to every user in the list of administrators.
- `defaultInternalRoles` property — A list of JasperReports Server roles assigned to every user not in the list of administrators.

The following example shows how to use the `mtExternalUserSetupProcessor` bean to define static roles. The configuration for `externalUserSetupProcessor` is similar:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
    externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
       ...
  <property name="adminUsernames">
    <list>
      <value>myorgadmin</value>
    </list>
  </property>

  <property name="defaultAdminRoles">
    <list>
      <value>ROLE_USER</value>
      <value>ROLE_ADMINISTRATOR</value>
    </list>
  </property>

  <property name="defaultInternalRoles">
    <list>
      <value>ROLE_USER</value>
    </list>
  </property>

...
```

5.7.3 Setting Default Roles

You can assign roles to all users using the `defaultInternalRoles` property of `externalUserSetupProcessor` or `mtExternalUserSetupProcessor`. The following example shows how to use this property in `externalUserSetupProcessor` to assign `ROLE_USER` to all users, in addition to the roles assigned by mapping:

```
  <property name="defaultInternalRoles">
    <list>
      <value>ROLE_USER</value>
    </list>
  </property>
```

5.7.4 Avoiding Role Collisions

If an external role has the same name as an internal role at the same organization level, JasperReports Server adds a suffix such as _EXT to the external role name to avoid collisions. For example, a user with the externally defined role ROLE_ADMINISTRATOR is assigned the role ROLE_ADMINISTRATOR_EXT in the JasperReports Server database. This ensures that internal administrator accounts like jasperadmin and superuser can still log in as internal administrators with the associated permissions.

You can set the extension in the `conflictingExternalInternalRoleNameSuffix` property in the `externalUserSetupProcessor` or `mtExternalUserSetupProcessor` bean. If the property doesn't appear in the bean, the extension is still implemented but defaults to _EXT. The following example shows how to configure this property:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
    externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
  <property name="conflictingExternalInternalRoleNameSuffix"
        value="_EXTERNAL"/>
  <property name="organizationRoleMap">
      ...
     <!-- Example of mapping customer roles to JRS roles -->
      ...
  </property>
```

5.7.5 Restricting the Mapping to Whitelisted Roles

You may not want every role in your external authority to appear as a role in JasperReports Server. Use the `permittedRolesRegex` property of the `externalUserSetupProcessor` bean or `mtExternalUserSetupProcessor` bean to specify which external roles become roles in JasperReports Server. You can use regular expressions to specify multiple roles that match the expression.

For example, to restrict the roles you create in JasperReports Server to roles that begin with JRS_ or EXT_ in your external authority, you would configure `permittedRolesRegex` in a way similar to the following:

```
        <property name="permittedRolesRegex">
            <list>
                <value>JRS_.*</value>
                <value>EXT_.*</value>
            </list>
        </property>
```

To allow all roles, use .* or comment out the property. If the property is omitted, all roles in the external authority are synchronized with roles in JasperReports Server.

5.7.6 Supporting Additional Characters in Role Names

The default mapping from attributes in your external authentication server to roles in JasperReports Server supports only alphanumeric characters and underscores. If a role in your external authority contains unsupported characters, each sequence of unsupported characters is replaced with a single underscore. For example, ROLE$-DEMO)EXT maps to ROLE_DEMO_EXT.

You can extend the supported character set by modifying the `permittedExternalRoleNameRegex` property of the `externalUserSetupProcessor` bean or `mtExternalUserSetupProcessor` bean. Check the sample configuration file for your deployment to determine which bean to modify.

The default value of the `permittedExternalRoleNameRegex` property is the regular expression [A-Za-z0-9_]+. Edit this expression to add supported characters. For example, the following syntax allows alphanumeric characters, underscores, and the Cyrillic letter Я (Unicode 042F):

```
<bean id="mtExternalUserSetupProcessor"  class="com.jaspersoft.jasperserver.api.security.
        externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
 <property name="userAuthorityService">
   <ref bean="${bean.internalUserAuthorityService}"/>
 </property>
  .....
 <property name="permittedExternalRoleNameRegex"
     value="[A-Za-z0-9_\u042F]+">
</bean>
```

Do not allow the following in role names: spaces, periods or |, [ ], `, ", ', ~, !, #, $, %, ^, &, [,], *, +, =, ;, :, ?, <, >, }, {, ), (, ], [, /, or \. Adding these characters in the `permittedExternalRoleNameRegex` property may cause unexpected behavior, such as the inability to delete or edit roles containing those characters.

5.8 Setting the User Organization

Organizations are a feature of JasperReports Server commercial editions. Skip this section if you have JasperReports Server community edition.

In the sample-applicationContext-externalAuth-db-mt.xml file, the `tenantId` is returned within the field `usersByUsernameAndTenantNameQuery` in `externalUserTenantDetailsService`. This query provides the required information to the tenant processor, so no additional configuration is needed.

Organizations created during external user login have an administrator with a default password. The admin username and password are configurable. See **5.8.1, "Setting Up Default Admins for Organizations," on page 79**. For security reasons, you should change the default password of any organization admin. See **2.4.4, "Initialization of JasperReports Server for External Users," on page 19** for a process to initialize the server, including organization admins, before going into production with external authentication.

5.8.1 Setting Up Default Admins for Organizations

In a multi-organization deployment, JasperReports Server creates a `jasperadmin` user whenever you create a new organization. The `jasperadmin` user is also given a standard default password. When creating multiple organizations using external authentication, you can set a different default password for jasperadmin, remove the jasperadmin user, and/or create additional default users in each new organization created by external authentication. Optionally, you can encrypt the password in the configuration files. See the *JasperReports Server Security Guide* for more information on default users in every organization.

For security reasons, you should change the default password of any organization admin. See **2.4.4, "Initialization of JasperReports Server for External Users," on page 19** for a process to initialize the server, including organization admins, before going into production with external authentication.

To set up admin users:

1. Open your sample-applicationContext-xxx-externalAuth.xml file in a text editor.
2. Locate the `externalTenantSetupUsers` property in the `externalTenantSetupProcessor` bean.
3. The sample contains a bean of class `ExternalTenantSetupUser` already configured for `jasperadmin`.

```
<property name="externalTenantSetupUsers">
  <list>
    <bean class="com.jaspersoft.jasperserver.multipleTenancy.security.
                externalAuth.processors.MTAbstractExternalProcessor.ExternalTenantSetupUser">
      <property name="username" value="${new.tenant.user.name.1}"/>
      <property name="fullName" value="${new.tenant.user.fullname.1}"/>
      <property name="password" value="${new.tenant.user.password.1}"/>
      <property name="emailAddress" value="${new.tenant.user.email.1}"/>
      <property name="roleSet">
        <set>
          <value>ROLE_ADMINISTRATOR</value>
          <value>ROLE_USER</value>
        </set>
      </property>
    </bean>
  </list>
</property>
```

4. To create additional admin users for each external organization, create a bean of class `ExternalTenantSetupUser` for each admin user you want.

```
    <bean class="com.jaspersoft.jasperserver.multipleTenancy.security.
                externalAuth.processors.MTAbstractExternalProcessor.ExternalTenantSetupUser">
      <property name="username" value="${new.tenant.user.name.2}"/>
      <property name="fullName" value="${new.tenant.user.fullname.2}"/>
      <property name="password" value="${new.tenant.user.password.2}"/>
      <property name="emailAddress" value="${new.tenant.user.email.2}"/>

      <property name="roleSet">
        <set>
          <value>ROLE_ADMINISTRATOR</value>
          <value>ROLE_USER</value>
        </set>
      </property>
    </bean>
```

5. The ${...} syntax above references values configured in the following file:
   <js-install>\buildomatic\conf_source\iePro\js.config.properties file.
   To set these values, open <js-install>\buildomatic\conf_source\iePro\js.config.properties and edit the entries there.

```
new.tenant.user.name.1=jasperadmin
new.tenant.user.fullname.1=jasperadmin
new.tenant.user.password.1=mynewpassword
new.tenant.user.email.1=

new.tenant.user.name.2=anotheradmin
new.tenant.user.fullname.2=Another Admin
new.tenant.user.password.2=anotherpassword
new.tenant.user.email.2=
```

Note: The property names, for example, `new.tenant.user.name.1`, are arbitrary. You can use any name for each property as long as the name in the applicationContext-externalAuth-xxx.xml file matches the name in the js.config.properties file.

6. If you want to obfuscate the default passwords in the js.config.properties files, encrypt them as described in the *JasperReports Server Security Guide*. Obfuscation must be implemented before you install the server.
7. If you don't want to obfuscate default passwords, you can eliminate the reference to js.config.properties and instead configure the values directly in the `externalTenantSetupUsers` property in the applicationContext-externalAuth-xxx.xml file. For example:

```
    <property name="username" value="anotheradmin"/>
    <property name="fullName" value="Another Admin"/>
    <property name="password" value="anotherpassword"/>
    <property name="emailAddress" value=""/>
```

5.8.2 Mapping Organization Names

You have the option to use the `organizationMap` property in the `externalTenantSetupProcessor` bean to map organization names extracted from your external authority to organization names in JasperReports Server. To do this, create a key/value pair for each organization you want to map, specifying the external organization name as the key and the organization name in JasperReports Server as the value. When mapping organizations, the server determines the mapped name and uses it as the name, ID, and description of the organization.

For example, the following would map users in External_Org_1 in the external authority to JRS_Org_1 in JasperReports Server and users in External_Org_2 in the external authority to JRS_Org_2 in JasperReports Server:

```
<property name="organizationMap">
    <map>
        <entry key="External_Org_1" value="JRS_Org_1" />
        <entry key="External_Org_2" value="JRS_Org_2" />
    </map>
</property>
```

The `organizationMap` property is optional. Any organization in your external authority that is not listed in `organizationMap` is mapped to an organization of the same name in JasperReports Server. However, if an organization in your external authority contains unsupported characters, each sequence of unsupported characters is replaced with a single underscore. For example, Human Resources maps to Human_Resources.

The `tenantIdNotSupportedSymbols` property of the `configurationBean` bean in the applicationContext.xml file lists the unsupported characters, including spaces and the following characters: `|`, `&`, `*`, `?`, `<`, `>`, `/`, `\`, `~`, `!`, `#`, `$`, `%`, `^`, `[`, `]`, or a space. If you want to list additional characters that should be replaced with an underscore, you can add them in this bean. However, we do not recommend removing any of the pre-defined characters, as JasperReports Server may not handle them correctly.

5.8.3 Specifying a Single Organization

If you have multiple organizations in your deployment, you can use the `externalTenantSetupProcessor` bean to specify a single organization assigned to all external users. To do this, set `externalTenantSetupProcessor`'s `defaultOrganization` property to the organization ID of the selected organization. If an organization with that ID already exists, all external users are assigned to that organization. If the organization does not exist, it's created when the first external user authenticates correctly.

When specifying the `defaultOrganization` value, the organization ID must not contain the following characters: |, &, *, ?, <, >, /, \, ~, !, #, $, %, ^, [, or].

The following example shows how to configure `externalTenantSetupProcessor` to assign all users to organization_1:

```
<bean id="externalTenantSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.
    security.externalAuth.processors.MTExternalTenantSetupProcessor"
    parent="abstractExternalProcessor">
  <property name="multiTenancyService">
    <ref bean="internalMultiTenancyService"/>
  </property>
  <property name="defaultOrganization" value="organization_1"/>
</bean>
```

Do not specify a null value for the `defaultOrganization` property. The null organization ID is usually reserved for special users like the system administrator and allows access to the repository folder of all other organizations.

Organizations created during external user login have an administrator with the default password. For security reasons, you should change the default password of any organization admin. See **2.4.4, "Initialization of JasperReports Server for External Users," on page 19** for a process to initialize JasperReports Server, including organization administrators, before going into production with external authentication.

5.9 Adding a Custom Processor

To create custom code to run on the server after the user has been authenticated, you can create a custom processor and add it to the processors list for the `externalUserProcessors` property of the `externalDataSynchronizer` bean. For example, you can add a processor that calls code to automatically create a user home folder for the authenticated user. See **7.3, "Creating a Custom Processor," on page 102**.

5.10 Configuring the Login Page for a Single-Organization Deployment

Even for a single-organization deployment, the user needs to enter an external organization name on the login page. To make the organization field available on the login form, you need to set the `alwaysRequestOrgIdOnLoginForm` property in the `externalAuthProperties` bean to `true`.

Deployments with multiple organizations in the database always display a line for the organization ID on the login page.

5.11 Restarting JasperReports Server

When you've configured all the beans in the appropriate files, restart JasperReports Server to make the changes take effect.

To test your configuration, navigate to the JasperReports Server login page. If the server and your external database are configured correctly, you can log into JasperReports Server with credentials stored in your external database. Try several users with different roles or organizations to verify your configuration.

CHAPTER 6 TOKEN-BASED AUTHENTICATION

If you have an application or portal you want to use with JasperReports Server, but no single sign-on environment, you can use the Jaspersoft token-based authentication and user management framework. To work with token-based authentication, your application or portal must do the following:

- Authenticate the end user according to the standards of your environment or application.
- Construct and, optionally, encrypt a token based on the authenticated user values within your application or process. The token values can include username, organization (if multi-tenancy is enabled), roles, and profile attributes. You can configure the token based on your needs for reporting and analysis within the JasperReports Server.
- Send the token to the JasperReports Server as part of an HTTP request.

When JasperReports Server receives the token, it will:

- Attempt to decrypt the token (if encrypted) and validate the token format.
- If the token is successfully parsed, use the information in the token to create and update the external user within JasperReports Server automatically.

The JasperReports Server deployment includes a sample file for token-based authentication in the <js-install>/samples/externalAuth-sample-config folder: the sample-applicationContext-externalAuth-preauth-mt.xml file (commercial editions) or sample-applicationContext-externalAuth-preauth.xml (community editions).

When using token-based authentication, it's extremely important that your external system is configured properly to prevent an attacker from forging the token. Consider taking security measures like connecting over HTTPS, encrypting the token, and using a time stamp, so the token is not exposed in the browser cache and cannot be easily reused.

6.1 Overview of Token-based Authentication

This section explains how JasperReports Server performs external authentication using a token, highlighting the differences with **“Default Internal Authentication” on page 15**.

The following diagram shows the general steps involved in logging into JasperReports Server using a token:

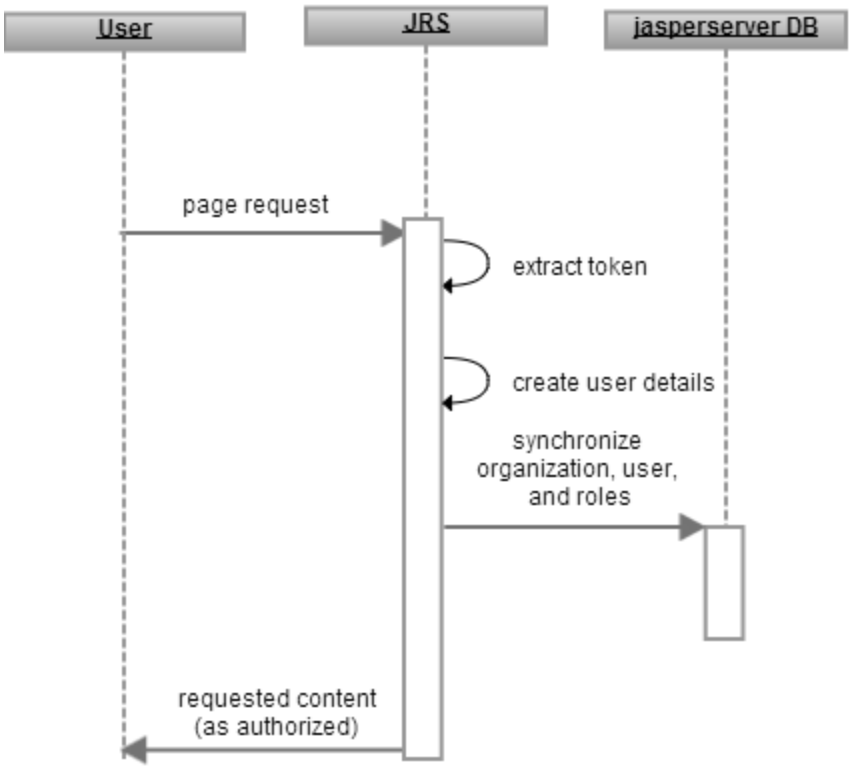

Figure 6-1 General Steps for Token-based Authentication

The following steps explain the interaction between the user's browser, JasperReports Server, and a pre-authenticated user:

1. A user requests any page in JasperReports Server.
2. If the user has not previously accessed JasperReports Server, the server looks for the `principalParameter` in the URL or request header. If the token is present and correctly formatted, the user is automatically authenticated.

   In token-based authentication, the JasperReports Server login screen is not displayed to the user and the user does not log in directly.

   After the user has authenticated and a JasperReports Server session has been created, future requests do not require the `principalParameter`.
3. JasperReports Server decrypts the token in the URL or request header and creates a principal object to represent the user's session in memory. The username, roles, and organization information are extracted from the token and synchronized with the internal database, where the user account is marked as an external user. The JasperReports Server environment reflects the user's roles and organization as defined in the token. For more information about synchronization, see **"Synchronization of External Users" on page 17**.
4. As with the default internal authorization, JasperReports Server now sends the requested content to the user, or if none was specified, the home page appropriate for the user. An application-server user session is

established and the connection between the requesting browser or process is maintained by repeatedly sending session identification information, usually in the form of an HTTP cookie. The token doesn't need to be resent until the user logs out or the session is inactive for a period of time.

When comparing these steps with those in **2.3, "Default Internal Authentication," on page 15**, you'll notice three significant differences:

- Token-based authentication doesn't use a login screen. Instead, it depends on the `principalParameter` in the URL or request header. As long as the `principalParameter` is present, the request is automatically authenticated.
- The portal or other external authentication mechanism is responsible for passing the correct username, and any roles, organization, or profile attributes as part of a token in the URL or request header.
- JasperReports Server decrypts the token, extracts the user details, and uploads the data to the internal database. The internal database is synchronized with any new information in the user's principal object.

6.2 Configuring JasperReports Server for Token-based Authentication

A sample file for configuring JasperReports Server to work with token-based authentication is included in the JasperReports Server distribution. Sample files are located in the <js-install>/samples/externalAuth-sample-config directory of your JasperReports Server. The file included depends on your version of JasperReports Server:

- sample-applicationContext-externalAuth-preauth.xml — Sample file for implementing token-based authentication for a single organization. This file is included in the community edition of JasperReports Server.
- sample-applicationContext-externalAuth-preauth-mt.xml — Sample file for implementing token-based authentication for multiple organizations. This file is included in commercial editions of JasperReports Server. To use external authentication with a commercial version and a single organization, you need to specify your organization as described in **6.6.3, "Specifying a Single Organization," on page 98**.

To configure JasperReports Server to work with your authentication method, modify and deploy the sample configuration file:

1. Make a copy of the preauth sample file in the <js-install>/samples/externalAuth-sample-config/ directory and rename it to remove the sample- prefix. For example, rename sample-applicationContext-externalAuth-preauth.xml to applicationContext-externalAuth-preauth.xml.
2. Edit the file you created and configure the beans correctly for your deployment, as described in the following sections.
3. Place the modified file in the <js-webapp>/WEB-INF directory.

<js-webapp> is the location of the JasperReports Server web application in your application server, or where you're modifying the configuration files. The rest of this chapter refers to file names alone.

6.3 Overview of Token-based Authentication Beans

The sample-applicationContext-externalAuth-preauth[-mt].xml file contains the beans needed to enable token-based authentication. This section summarizes the most important beans in this file, including the beans you need to modify to configure JasperReports Server to work with token-based authentication.

- `proxyPreAuthenticatedProcessingFilter` — Bean that enables token-based authentication. This bean extracts and optionally decrypts the token from the request. When this proxy bean definition is present in

the application context, the Spring Security filter chain processes the authentication via the proxy definitions instead of the default internal filter. See **6.4, "Configuring the Token," on page 88** for more information.

- `preAuthenticatedManager` — Lists the available authentication providers. In token-based authentication, there's a single provider, the `JSPreAuthenticatedAuthenticationProvider`. This bean doesn't have to be configured
- `JSPreAuthenticatedAuthenticationProvider` — Custom authentication provider for token-based authentication. This bean has a constructor argument, `preAuthenticatedUserDetailsService`, which it uses to create the user details from the values passed in the token. See **6.4, "Configuring the Token," on page 88**.
- `externalDataSynchronizer` — Bean whose class creates a mirror image of the external user in the internal `jasperserver` database.
- `mtExternalUserSetupProcessor` or `externalUserSetupProcessor`– Bean that creates and configures the internal user corresponding to a successfully authenticated external user. Configure this bean to specify the roles given to external users. See **6.5, "User Roles," on page 92**.
- `externalTenantSetupProcessor` — For multi-tenant deployments, this bean creates and configures the internal organization for a successfully authenticated external user. See **6.6, "Mapping the User Organization," on page 95**.
- `externalProfileAttributeProcessor` — Optional processor that sets up user profile attributes. You don't need to configure this processor; the mapping is set up in the token configuration. Comment this bean out if not used.

6.4 Configuring the Token

To use your authentication provider with JasperReports Server's token-based authentication, you must pass a correctly formatted token in the HTTP header or the URL of the request. A token is a string of key/value pairs separated by a character specified in the configuration file.

6.4.1 Security of the Token

JasperReports Server will accept any properly formatted token; therefore, you need to protect the integrity of the token using measures such as:

- Use SSL to Connect to JasperReports Server to protect against token interception.
- Encrypt the token to protect against tampering. See **6.4.2.1, "Setting Token Decryption," on page 89** for more information.
- Configure the token to use a timestamp to protect against replay attacks. Without a timestamp, when you include the token in a web page or REST web service URL, the URL can be copied and used by unauthorized people or systems. Setting the expire time for the token will prevent tokens/URLs from being used to authenticate beyond the indicated time. You can set the expiry time depending on your use case. For a user who is logged into the application/portal and requesting access to JasperReports Server, expiry time of a minute or less from the request time is appropriate. See the descriptions for `expireTime` and `tokenExpireTimestampFormat` in **6.4.3, "preAuthenticatedUserDetailsService," on page 90**.

6.4.2 proxyPreAuthenticatedProcessingFilter bean

Some token configuration is specified in the `proxyPreAuthenticatedProcessingFilter` bean using the following properties:

- `principalParameter` — A fixed string at the start of the token that triggers token-based authentication. The first time the user accesses JasperReports Server, the `principalParameter` must be present in the request. `principalParameter` can be any URL-safe string that's different from all other JasperReports Server request parameter names.
- `tokenInRequestParam` — Boolean that specifies the location of `principalParameter` as follows:
  - `true` — JasperReports Server looks for `principalParameter` in the request URL only.
  - `false` — JasperReports Server looks for `principalParameter` in the request header only.
  - absent — JasperReports Server checks the request header first and then the URL.
- `tokenDecryptor` — Specifies the class to use to decrypt the token. For security reasons, you should encrypt the token to keep its payload from being intercepted when it's exposed in the browser's cache or when SSL is not enabled. If the token is encrypted, you must provide an implementation of Jaspersoft's token decryptor interface `CipherI`. The default assumes the token is unencrypted and passes the token through as plaintext.

6.4.2.1 Setting Token Decryption

If you want to encrypt the token, you need to select the encryption you'll use to encrypt the token in the calling application and decrypt the token in the JasperReports Server. To implement the token decryption algorithm in JasperReports Server, you must write Java code that implements Jaspersoft's `CipherI`interface and include it in the Spring configuration for security. `CipherI` is located in the com.jaspersoft.jasperserver.api.common.crypto package in the jasperserver-api.common.*[version]*.jar jar file.

`CipherI` is a stub interface that allows you to define two methods:

- `encrypt` — Takes a text string and returns a text string encrypted according to the specified algorithm
- `decrypt` — Takes a text string and returns a text string decrypted according to the specified algorithm.

You can implement any encryption method you choose.

```
package com.mycompany;

public class MyCipher implements CipherI {
	@Override
	public String encrypt(String plainText) {

		...
		return cipherText;
	}

	@Override
	public String decrypt(String cipherText) {

		...
		return plainText;
	}
}
```

Once you've created your implementation of `CipherI`, you need to incorporate it in a jar file and reference it in the `tokenDecryptor` property of `proxyPreAuthenticatedProcessingFilter`:

```
<bean id="proxyPreAuthenticatedProcessingFilter"  class="com.jaspersoft.jasperserver.api.security.
    externalAuth.preauth.BasePreAuthenticatedProcessingFilter">
        ...
  <property name="tokenDecryptor">
    <bean class="com.mycompany.MyCipher"/>
  </property>
        ...
</bean>
```

6.4.3 preAuthenticatedUserDetailsService

Specify the token format using the `preAuthenticatedUserDetailsService` constructor argument in the `preAuthenticatedManager` bean. An inline bean in `preAuthenticatedUserDetailsService` lets you specify the token properties. The following example shows one way you might configure this bean:

Table 6-1 Sample Code Configuration

```
<bean class="com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.preauth.
              JSPreAuthenticatedAuthenticationProvider">
  <property name="preAuthenticatedUserDetailsService">
    <bean class="com.jaspersoft.jasperserver.multipleTenancy.security.externalAuth.preauth.
          MTJSPreAuthenticatedUserDetailsService">
      <property name="tokenPairSeparator" value="|"/>
        <property name="tokenFormatMapping">
          <map>
            <entry key="username" value="u" />
            <entry key="roles" value="r" />
            <entry key="orgId" value="o" />
            <entry key="expireTime" value="exp" />
            <entry key="profile.attribs" >
              <map>
                <entry key="profileAttrib1" value="pa1" />
                <entry key="profileAttrib2" value="pa2" />
              </map>
            </entry>
          </map>
        </property>
      <property name="tokenExpireTimestampFormat" value="yyyyMMddHHmmssZ"/>
    </bean>
  </property>
</bean>
```

The inline bean has the following properties:

- `tokenPairSeparator` — A single character used as the separator for key-value pairs in the token, for example, pipe (|, encoded in URLs as %7C) (default). Cannot include comma (,), question mark (?), or equals sign (=), as these are used elsewhere in the token.

If you're passing the token in the URL, you need to encode all equal signs (=, encoded as %3D) and pipe symbols (|, encoded as %7C) in the token to make the token URL-safe.

- `tokenFormatMapping` — The mapping between parameters in the token and attributes of the principal object in JasperReports Server, like roles and profile attributes. Entries in the mapping have the format <entry key ="JRS" value="keyintoken">, for example, <entry key="username" value="u" />. The key, such as username, is a fixed value used in the JasperReports Server code. The value is a parameter in the token mapped to the key in JasperReports Server.
- tokenFormatMapping supports the following JasperReports Server fixed values:
  - `username` — The external user's username.
  - `roles` — The external user's roles.
  - `orgId` — (optional, multi-tenant implementations only) In a multi-organization deployment, the external user's organization. In a single-tenant deployment, or in a deployment that maps all users to the same organization using `defaultOrganization`, you can comment out this property.
  - `expireTime` — The key for the token expiration time. A time formatted as configured in the `tokenExpireTimestampFormat` property.
  - `profile.attribs` — (optional) This lets you specify a map of key-value pairs for profile attributes. Comment this out if you don't want to map profile attributes. You must also comment out the reference to `externalProfileAttributeProcessor` in the `externalUserProcessors` property of the `externalDataSynchronizer`.

    

    The user password does not appear as a standard parameter in the token format. Passwords don't need to be passed in the token in most cases, because JasperReports Server doesn't store passwords for externally-defined users. If you want to store passwords, you can pass them in using profile attributes.

  `tokenFormatMapping` also supports the following property:
  - `tokenExpireTimestampFormat` — The format of the token expiration time, `expireTime`, based on the `java.text.SimpleDateFormat`. Valid patterns for timestamps can contain only letters, numbers, dashes, underscores, and periods. The default pattern is `yyyyMMddHHmm`, for example `2001404150601`. For more information about the valid patterns for this field, refer to: http://download.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html.

The token has to match the configuration you specify in `preAuthenticatedUserDetailsService`. With the configuration shown in **Table 6-1, "Sample Code Configuration," on page 90**, you would use the following token syntax in the page header:

```
pp=u=user|r=role1,role2,...|o=org1[,org2,...]|pa1=PA11,PA12,...|pa2=PA21,PA22,...|exp=time
```

In this example, `pp` (`principalParameter`) is the name of the header attribute or the URL token parameter name.

If you're passing the token in the URL, you need to encode all equal signs (=, encoded as %3D) and pipe symbols (|, encoded as %7C) in the token to make the token URL-safe.

The key-value pairs can appear in the token in any order, but must be delineated by the separator specified in the `tokenPairSeparator` property:

- `u` — Takes a single value that maps to the username in JasperReports Server.
- `r` — Takes as values a comma separated list of roles, each mapped to a separate role in JasperReports Server. If you have defined default internal roles, this parameter is optional. See **6.5, "User Roles," on page 92**.
- `o` — Takes as values a comma separated list that maps to an organization in the JasperReports Server. If more than one organization is listed, it's interpreted as an organization hierarchy. For example, `o=A,B` maps the user to the B suborganization of A.

- exp — Takes a single time value formatted as configured in the tokenExpireTimestampFormat property. If exp is earlier than current time, authentication is denied. if exp is absent, the token never expires.
- pa1 — Profile attribute that maps to profileAttrib1 in the JasperReports Server repository.
- pa2 — Profile attribute that maps to profileAttrib2 in the JasperReports Server repository.

With this configuration, the following would be a valid token. The user would be placed in the Sales suborganization of the EMEA organization:

```
pp=u=Sven|r=Manager|o=EMEA,Sales|pa1=Sweden
```

If you're passing the token in the URL, you need to encode all equal signs (=) (as %3D) and pipe symbols (|) (as %7C) in the token to make the token URL-safe:

```
http://localhost:8080/jasperserver?pp=u%3DSven%7Cr%3DManager%7Co%3DEMEA,Sales%7Cpa1%3DSweden
```

6.5 User Roles

User roles are mapped using the externalUserSetupProcessor or mtExternalUserSetupProcessor bean. See the configuration file for your deployment to see which bean applies in your case. For token-based authentication, you can define user roles in the following ways:

- Map the user roles received from the token – Define a mapping between roles in the external authority and roles in JasperReports Server. See 6.5.1, “Mapping User Roles,” on page 92.
- Define user roles statically – If you don't have user roles in the token, you can define static roles for administrative and non-administrative users. See 6.5.2, “Defining Static Roles,” on page 94.

6.5.1 Mapping User Roles

If you have user roles in the token, you can map them to roles in JasperReports Server using the organizationRoleMap property of the externalUserSetupProcessor or mtExternalUserSetupProcessor bean.

The organizationRoleMap property contains key/value pairs that map external role names to internal ones. For commercial JasperReports Server deployments, you need to choose the level at which the role is assigned:

- To map to an internal role at the organization level, append |* to the name of the internal role, for example, ROLE_USER|*. Roles mapped at the organization level don't have administrative privileges.
- To map to an internal role at the system (null) level, don't modify the internal role name, for example, ROLE_ADMINISTRATOR. Roles at the system level are usually reserved for special users like the system administrator and allow access to the repository folder of all other organizations.

The following example shows how you might configure the externalUserSetupProcessor bean to map roles from the external authority to roles in JasperReports Server:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
    externalAuth.processors.MTExternalUserSetupProcessor"  parent="abstractExternalProcessor">
<property name="organizationRoleMap">
   <map>
    <!-- Example of mapping customer roles to JRS roles -->
    <entry key= ROLE_ADMIN_EXTERNAL_ORGANIZATION value=ROLE_ADMINISTRATOR />
  </map>
</property>
```

6.5.1.1 Setting Default Roles

You can assign roles to all users using the `defaultInternalRoles` property of `externalUserSetupProcessor` or `mtExternalUserSetupProcessor`. The following example shows how to use this property in `externalUserSetupProcessor` to assign `ROLE_USER` to all users, in addition to the roles assigned by mapping:

```
<property name="defaultInternalRoles">
  <list>
    <value>ROLE_USER</value>
  </list>
</property>
```

6.5.1.2 Avoiding Role Collisions

If an external role has the same name as an internal role at the same organization level, JasperReports Server adds a suffix such as _EXT to the external role name to avoid collisions. For example, a user with the externally defined role ROLE_ADMINISTRATOR is assigned the role ROLE_ADMINISTRATOR_EXT in the JasperReports Server database. This ensures that internal administrator accounts like jasperadmin and superuser can still log in as internal administrators with the associated permissions.

You can set the extension in the `conflictingExternalInternalRoleNameSuffix` property in the `externalUserSetupProcessor` or `mtExternalUserSetupProcessor` bean. If the property doesn't appear in the bean, the extension is still implemented but defaults to _EXT. The following example shows how to configure this property:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
    externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
  <property name="conflictingExternalInternalRoleNameSuffix"
        value="_EXTERNAL"/>
  <property name="organizationRoleMap">
      ...
     <!-- Example of mapping customer roles to JRS roles -->
      ...
  </property>
```

6.5.1.3 Restricting the Mapping to Whitelisted Roles

You may not want every role in your external authority to appear as a role in JasperReports Server. Use the `permittedRolesRegex` property of the `externalUserSetupProcessor` bean or `mtExternalUserSetupProcessor` bean to specify which external roles become roles in JasperReports Server. You can use regular expressions to specify multiple roles that match the expression.

For example, to restrict the roles you create in JasperReports Server to roles that begin with JRS_ or EXT_ in your external authority, you would configure `permittedRolesRegex` in a way similar to the following:

```
        <property name="permittedRolesRegex">
            <list>
                <value>JRS_.*</value>
                <value>EXT_.*</value>
            </list>
        </property>
```

To allow all roles, use .* or comment out the property. If the property is omitted, all roles in the external authority are synchronized with roles in JasperReports Server.

6.5.1.4 Supporting Additional Characters in Role Names

The default mapping from attributes in your external authentication server to roles in JasperReports Server supports only alphanumeric characters and underscores. If a role in your external authority contains unsupported characters, each sequence of unsupported characters is replaced with a single underscore. For example, ROLE$-DEMO)EXT maps to ROLE_DEMO_EXT.

You can extend the supported character set by modifying the `permittedExternalRoleNameRegex` property of the `externalUserSetupProcessor` bean or `mtExternalUserSetupProcessor` bean. Check the sample configuration file for your deployment to determine which bean to modify.

The default value of the `permittedExternalRoleNameRegex` property is the regular expression [A-Za-z0-9_]+. Edit this expression to add supported characters. For example, the following syntax allows alphanumeric characters, underscores, and the Cyrillic letter Я (Unicode 042F):

```
<bean id="mtExternalUserSetupProcessor"  class="com.jaspersoft.jasperserver.api.security.
        externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
 <property name="userAuthorityService">
   <ref bean="${bean.internalUserAuthorityService}"/>
 </property>
  .....
 <property name="permittedExternalRoleNameRegex"
     value="[A-Za-z0-9_\u042F]+">
</bean>
```

Do not allow the following in role names: spaces, periods or |, [ ], `, ", ', ~, !, #, $, %, ^, &, [,], *, +, =, ;, :, ?, <, >, }, {, ), (, ], [, /, or \. Adding these characters in the `permittedExternalRoleNameRegex` property may cause unexpected behavior, such as the inability to delete or edit roles containing those characters.

6.5.2 Defining Static Roles

If you're mapping all your external users to a single organization, you can assign static roles to users. This lets you specify a list of administrative users and roles, and a list of roles for non-administrative users. To define static roles, use the `externalUserSetupProcessor` or `mtExternalUserSetupProcessor` bean. To set up static roles, locate the version of the bean used in your sample file and configure the following properties:

- `adminUserNames` property — A list of usernames granted internal administrator privileges in JasperReports Server. The username values must exactly match the usernames authenticated and returned by the external authority.
- `defaultAdminRoles` property — A list of JasperReports Server internal roles. These are assigned to every user in the list of administrators.
- `defaultInternalRoles` property — A list of JasperReports Server roles assigned to every user not in the list of administrators.

The following example shows how to use the `mtExternalUserSetupProcessor` bean to define static roles. The configuration for `externalUserSetupProcessor` is similar:

```
<bean id="mtExternalUserSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.security.
    externalAuth.processors.MTExternalUserSetupProcessor"
    parent="abstractExternalProcessor">
       ...
  <property name="adminUsernames">
    <list>
      <value>myorgadmin</value>
    </list>
  </property>

  <property name="defaultAdminRoles">
    <list>
      <value>ROLE_USER</value>
      <value>ROLE_ADMINISTRATOR</value>
    </list>
  </property>

  <property name="defaultInternalRoles">
    <list>
      <value>ROLE_USER</value>
    </list>
  </property>

...
```

6.6 Mapping the User Organization

There are two ways to set the user organization:

- If you have multiple organizations in your deployment, set up the token to contain organization information as above. If you want organization names in JasperReports Server to be different from the ones in the external authority, use the organizationMap property, as described in **6.6.2, "Mapping Organization Names," on page 97**.
- If your JasperReports Server has multiple organizations, but you want all users to be in the same organization, use the externalTenantSetupProcessor bean to specify the organization, as described in **6.6.3, "Specifying a Single Organization," on page 98**; in this case you don't need to include organization information in the token.

If your JasperReports Server deployment supports only a single organization (all community deployments and some professional editions), you don't need to set organization information.

6.6.1 Setting Up Default Admins for Organizations

In a multi-organization deployment, JasperReports Server creates a `jasperadmin` user whenever you create a new organization. The `jasperadmin` user is also given a standard default password. When creating multiple organizations using external authentication, you can set a different default password for jasperadmin, remove the jasperadmin user, and/or create additional default users in each new organization created by external authentication. Optionally, you can encrypt the password in the configuration files. See the *JasperReports Server Security Guide* for more information on default users in every organization.

For security reasons, you should change the default password of any organization admin. See **2.4.4, "Initialization of JasperReports Server for External Users," on page 19** for a process to initialize the server, including organization admins, before going into production with external authentication.

To set up admin users:

1. Open your sample-applicationContext-xxx-externalAuth.xml file in a text editor.
2. Locate the `externalTenantSetupUsers` property in the `externalTenantSetupProcessor` bean.
3. The sample contains a bean of class `ExternalTenantSetupUser` already configured for `jasperadmin`.

```
<property name="externalTenantSetupUsers">
  <list>
    <bean class="com.jaspersoft.jasperserver.multipleTenancy.security.
                externalAuth.processors.MTAbstractExternalProcessor.ExternalTenantSetupUser">
      <property name="username" value="${new.tenant.user.name.1}"/>
      <property name="fullName" value="${new.tenant.user.fullname.1}"/>
      <property name="password" value="${new.tenant.user.password.1}"/>
      <property name="emailAddress" value="${new.tenant.user.email.1}"/>
      <property name="roleSet">
        <set>
          <value>ROLE_ADMINISTRATOR</value>
          <value>ROLE_USER</value>
        </set>
      </property>
    </bean>
  </list>
</property>
```

4. To create additional admin users for each external organization, create a bean of class `ExternalTenantSetupUser` for each admin user you want.

```
    <bean class="com.jaspersoft.jasperserver.multipleTenancy.security.
                externalAuth.processors.MTAbstractExternalProcessor.ExternalTenantSetupUser">
      <property name="username" value="${new.tenant.user.name.2}"/>
      <property name="fullName" value="${new.tenant.user.fullname.2}"/>
      <property name="password" value="${new.tenant.user.password.2}"/>
      <property name="emailAddress" value="${new.tenant.user.email.2}"/>

      <property name="roleSet">
        <set>
          <value>ROLE_ADMINISTRATOR</value>
          <value>ROLE_USER</value>
        </set>
      </property>
    </bean>
```

5. The ${...} syntax above references values configured in the following file:
   <js-install>\buildomatic\conf_source\iePro\js.config.properties file.
   To set these values, open <js-install>\buildomatic\conf_source\iePro\js.config.properties and edit the entries there.

```
new.tenant.user.name.1=jasperadmin
new.tenant.user.fullname.1=jasperadmin
new.tenant.user.password.1=mynewpassword
new.tenant.user.email.1=

new.tenant.user.name.2=anotheradmin
new.tenant.user.fullname.2=Another Admin
new.tenant.user.password.2=anotherpassword
new.tenant.user.email.2=
```

Note: The property names, for example, `new.tenant.user.name.1`, are arbitrary. You can use any name for each property as long as the name in the applicationContext-externalAuth-xxx.xml file matches the name in the js.config.properties file.

6. If you want to obfuscate the default passwords in the js.config.properties files, encrypt them as described in the *JasperReports Server Security Guide*. Obfuscation must be implemented before you install the server.
7. If you don't want to obfuscate default passwords, you can eliminate the reference to js.config.properties and instead configure the values directly in the `externalTenantSetupUsers` property in the applicationContext-externalAuth-xxx.xml file. For example:

```
    <property name="username" value="anotheradmin"/>
    <property name="fullName" value="Another Admin"/>
    <property name="password" value="anotherpassword"/>
    <property name="emailAddress" value=""/>
```

6.6.2 Mapping Organization Names

You have the option to use the `organizationMap` property in the `externalTenantSetupProcessor` bean to map organization names extracted from your external authority to organization names in JasperReports Server. To do this, create a key/value pair for each organization you want to map, specifying the external organization name as the key and the organization name in JasperReports Server as the value. When mapping organizations, the server determines the mapped name and uses it as the name, ID, and description of the organization.

For example, the following would map users in External_Org_1 in the external authority to JRS_Org_1 in JasperReports Server and users in External_Org_2 in the external authority to JRS_Org_2 in JasperReports Server:

```
<property name="organizationMap">
    <map>
        <entry key="External_Org_1" value="JRS_Org_1" />
        <entry key="External_Org_2" value="JRS_Org_2" />
    </map>
</property>
```

The `organizationMap` property is optional. Any organization in your external authority that is not listed in `organizationMap` is mapped to an organization of the same name in JasperReports Server. However, if an organization in your external authority contains unsupported characters, each sequence of unsupported characters is replaced with a single underscore. For example, Human Resources maps to Human_Resources.

The `tenantIdNotSupportedSymbols` property of the `configurationBean` bean in the applicationContext.xml file lists the unsupported characters, including spaces and the following characters: `|`, `&`, `*`, `?`, `<`, `>`, `/`, `\`, `~`, `!`, `#`, `$`, `%`, `^`, `[`, `]`, or a space. If you want to list additional characters that should be replaced with an underscore, you can add them in this bean. However, we do not recommend removing any of the pre-defined characters, as JasperReports Server may not handle them correctly.

6.6.3 Specifying a Single Organization

In a multi-tenant deployment, if you don't have organization information in your token, your users are mapped to the NULL organization, which means they will be given root privilege. If your deployment has only one organization, this organization is used by default.

If you have multiple organizations in your deployment, you can use the `externalTenantSetupProcessor` bean to specify a single organization assigned to all external users. To do this, set `externalTenantSetupProcessor`'s `defaultOrganization` property to the organization ID of the selected organization. If an organization with that ID already exists, all external users are assigned to that organization. If the organization does not exist, it's created when the first external user authenticates correctly.

When specifying the `defaultOrganization` value, the organization ID must not contain the following characters: |, &, *, ?, <, >, /, \, ~, !, #, $, %, ^, [, or].

The following example shows how to configure `externalTenantSetupProcessor` to assign all users to organization_1:

```
<bean id="externalTenantSetupProcessor" class="com.jaspersoft.jasperserver.multipleTenancy.
    security.externalAuth.processors.MTExternalTenantSetupProcessor"
    parent="abstractExternalProcessor">
  <property name="multiTenancyService">
    <ref bean="internalMultiTenancyService"/>
  </property>
  <property name="defaultOrganization" value="organization_1"/>
</bean>
```

Do not specify a null value for the `defaultOrganization` property. The null organization ID is usually reserved for special users like the system administrator and allows access to the repository folder of all other organizations.

Organizations created during external user login have an administrator with the default password. For security reasons, you should change the default password of any organization admin. See **2.4.4, "Initialization of JasperReports Server for External Users," on page 19** for a process to initialize JasperReports Server, including organization administrators, before going into production with external authentication.

6.7 Adding a Custom Processor

To create custom code to run on the server after the user has been authenticated, you can create a custom processor and add it to the processors list for the `externalUserProcessors` property of the `externalDataSynchronizer` bean. For example, you can add a processor that calls code to automatically create a user home folder for the authenticated user. See **7.3, "Creating a Custom Processor," on page 102**.

6.8 Restarting JasperReports Server

When you've configured all the beans in the appropriate files, restart JasperReports Server to make the changes take effect.

To test your configuration, log into your external authority and request a page from JasperReports Server. If the token and server are configured correctly, you can access JasperReports Server. Try several users with different roles or organizations to verify your configuration.

CHAPTER 7 ADVANCED TOPICS

External authentication as presented in this guide is a configuration of the default JasperReports Server product. To implement other behavior during external authentication, you must write custom classes and deploy them with the server. This chapter describes the bean APIs you can use for customization. It also explains specific customizations such as custom processors.

This chapter contains the following sections:

7.1 Internal Authentication Beans

JasperReports Server uses Spring Security for authentication. When JasperReports Server receives a request, from either a person using the web interface or an application using the web services, the Spring Security filter chain processes that request. The filter chains in applicationContext-security-web.xml are based on the standard Spring Security filter chains.

The following figure shows the most important authentication-related beans in the filter chain:

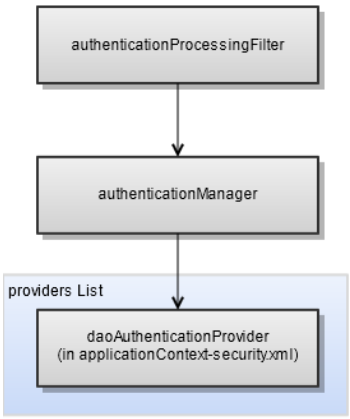

Figure 7-1 Beans for Internal Authentication in JasperReports Server

1. `authenticationProcessingFilter` — Responsible for authenticating the user and creating the principal object in memory. With default authentication, this filter redirects the user to the login page, then processes the organization ID, username, and password.
2. `authenticationManager` — Bean of Spring class `ProviderManager` that manages default authentication by invoking a list of providers. JasperReports Server relies on `${bean.daoAuthenticationProvider}` for internal authentication.
3. `${bean.daoAuthenticationProvider}` — Bean for performing authentication to the `jasperserver` internal database configured in <js-webapp>/WEB-INF/applicationContext-security.xml.

The filter chains for authentication are configured for the following patterns:

- The `/xmla` pattern represents the XML for Analysis (XML/A) servlet. They are configured with a set of filters designed to receive client SOAP requests. SOAP is implemented in the sample files for LDAP.
- The `/services/**` pattern represents the XML for SOAP web services. In this filter chain, the `${bean.basicProcessingFilter}` bean initiates internal authentication. SOAP is implemented in the sample files for LDAP.
- The `/rest/login`, `/rest/**`, and `rest_v2/**` patterns represent the XML for REST web services. In this filter chain, the `restAuthenticationProcessingFilter` and `${bean.basicProcessingFilter}` beans initiate internal authentication. REST is implemented in the sample files for LDAP.
- The `/**` pattern matches anything that wasn't caught by another pattern. This pattern is designed for people using web browsers. In this filter chain, the `authenticationProcessingFilter` bean initiates.

7.2 External Authentication Framework

JasperReports Server uses an external authentication bean framework to make it easier to configure the security system for external authentication. Although each external authentication situation is different, we've extracted the common principals and extended the set of Spring Security beans to help you configure your own external authentication.

Jaspersoft has implemented a proxy bean for each filter chain, for example, `delegatingAuthenticationRestProcessingFilter`, that serves as a proxy for the standard Spring

authenticationProcessingFilter and initiates external authentication for the pattern. These proxy beans inspect the application context for external authentication beans, and use them if they are present. Otherwise, the default internal authentication beans are used.

The following sample files use JasperReports Server's external authentication APIs to integrate with custom SSO servers following a CAS-like protocol:

- sample-applicationContext-externalAuth-sso.xml — Sample file for integrating CAS with a single-organization JasperReports Server. Included with the community edition of JasperReports Server only.
- sample-applicationContext-externalAuth-sso-mt.xml — Sample file for integrating CAS with a multiple-organization JasperReports Server. In this example, user details like external roles and organization are retrieved from an external database. Included with the commercial version of JasperReports Server only.

7.2.1 External Authentication Beans

The external authentication framework consists of the following beans:

1. proxyAuthenticationProcessingFilter – Bean to which the filter chain configured in applicationContext-security-web.xml delegates authentication via delegatingExceptionTranslationFilter. When a proxyAuthenticationProcessingFilter bean appears in an applicationContext-<customName>.xml file in the <js-webapp>/WEB-INF directory, delegatingExceptionTranslationFilter processes the authentication via the proxy definition instead of the default authenticationProcessingFilter.
2. customAuthenticationProvider – Bean for providing custom authentication. This bean takes information from the JasperReports Server login request and authenticates the user; it returns user name, roles, and organizations in a Spring userDetails object. customAuthenticationProvider is invoked by Spring's ProviderManager class, as specified in the providers property.
3. externalDataSynchronizer – Bean whose class creates a mirror image of the external user in the internal jasperserver database. The user synchronization work is performed by a list of processors specified in the externalUserProcessors property, called in the order of their appearance. This bean is invoked by the proxyAuthenticationProcessingFilter bean's class on successful user authentication. The externalDataSynchronizer bean has the following property:
   - externalUserProcessors property – Lists processors implementing the Processor interface. These processors run on JasperReports Server after the user has been authenticated. Mandatory processors to implement are:
     - TenantSetupProcessor – If you have multiple organizations in your deployment, implement a bean of this class to specify the mapping from organizations in your external data source to organizations in JasperReports Server.
     - ExternalUserSetupProcessor or mtExternalUserSetupProcessor – Implement a bean of this class to specify how to synchronize external users and roles with the internal JasperReports Server database.

   You can also write custom processors as described in **"Creating a Custom Processor" on page 102**.

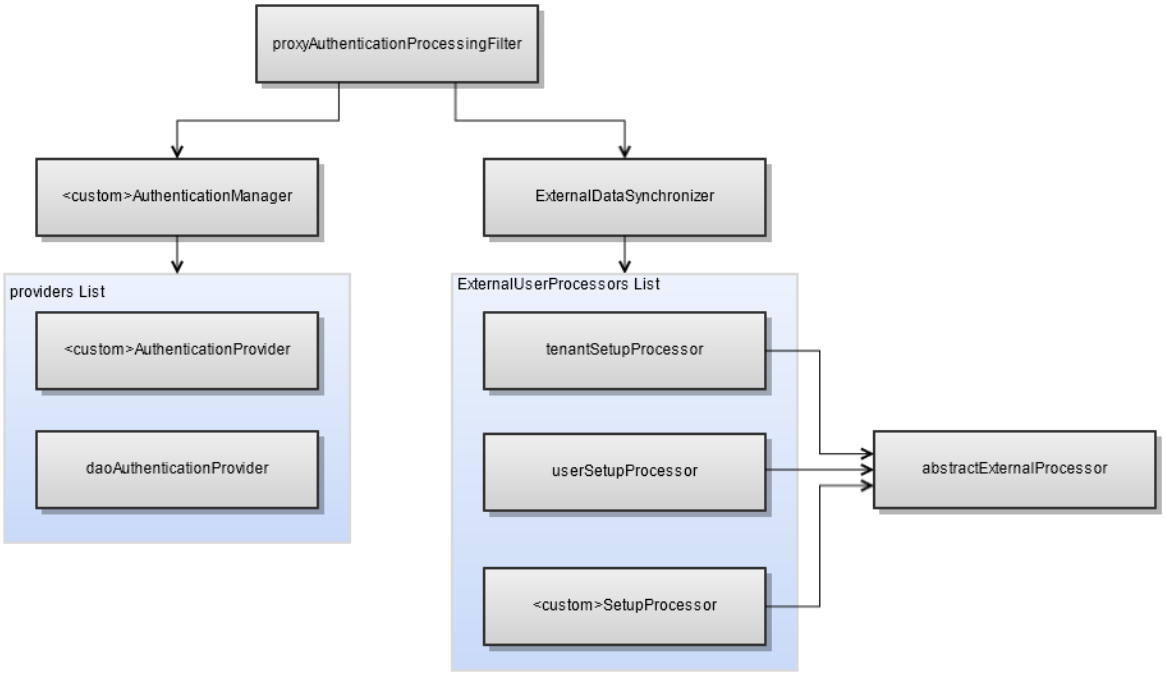

Figure 7-2 Beans for JasperReports Server External Authentication API

7.3 Creating a Custom Processor

To create custom code to run on the server after the user has been authenticated, you can create a custom processor and add it to the processors list for the `externalUserProcessors` property of the `externalDataSynchronizer` bean. To create a custom processor, extend the `AbstractExternalSetupProcessor` class or `MTAbstractExternalProcessor` class, which provide access to all the JasperReports Server service classes like the repository and user management. For example, to create a folder, you must use `HibernateRepositoryService` and `ObjectPermissionService`. See the description of Java APIs in the chapter "JasperReports Server APIs" in the *JasperReports Server Ultimate Guide* for more information on the available JasperReports Server services.

The processors for user setup and tenant mapping in the sample configuration files, like `externalUserProcessors` or `ldapExternalTenantProcessor`, are themselves examples of processor implementation. These processors are described in detail in the sections for each authentication mechanism, for example, **"Mapping to Multiple Organizations" on page 41**.

In addition, many of the sample application context files include an `externalUserFolderProcessor` bean that calls code to automatically create a user home folder for the authenticated user. The following example shows how to add `externalUserFolderProcessor` to the `externalUserProcessors` property of the `ExternalDataSynchronizer` bean:

```
<bean id="externalDataSynchronizer"
      class="com.jaspersoft.jasperserver.api.security.externalAuth.ExternalDataSynchronizerImpl">
  <property name="externalUserProcessors">
```

```
    <list>
      <ref local="externalUserSetupProcessor"/>
      <ref local="externalUserFolderProcessor"/>
    </list>
  </property>
</bean>
```

The following code block shows how you might configure an `externalUserFolderProcessor` bean in your applicationContext-externalAuth-xxx.xml configuration file:

```
<bean id="externalUserFolderProcessor"
      class="com.jaspersoft.jasperserver.api.security.externalAuth.processors.
      ExternalUserFolderProcessor"
      parent="abstractExternalProcessor">
  <property name="repositoryService" ref="${bean.unsecureRepositoryService}"/>
</bean>
```

To write a processor, extend `AbstractExternalUserProcessor` and overwrite the process method with your java code. This gives you access to the following services:

- `RepositoryService`
- `UserAuthorityService`
- `TenantService`
- `ProfileAttributeService`
- `ObjectPermissionService`

If you extend `MTAbstractExternalProcessor`, you can access the following `multiTenancyService` service:

7.4 Authentication Based on Request

When your user request has sufficient information for your custom authentication method to authenticate directly from the request, you can create a custom authentication provider to automatically authenticate the user and create organizations and roles. This corresponds to Spring Security's pre-authenticated scenario. The exact implementation depends on the external authentication you are using. In some cases, you may need to obtain user roles and organizations from a separate source.

If you're passing information in the HTTP request, as with Siteminder, it's extremely important that your external system is configured properly to prevent an attacker from forging the HTTP headers.

The JasperReports Server deployment includes a sample file for custom authentication in the <js-install>/samples/externalAuth-sample-config folder: the sample-applicationContext-externalAuth-template-mt.xml file (commercial editions) or sample-applicationContext-externalAuth-template.xml (community editions). This sample takes the IP address from the user's authentication request, creates a user with the same name in JasperReports Server, and uses the JasperReports Server API to create a user folder in the JasperReports Server Repository and set permissions.

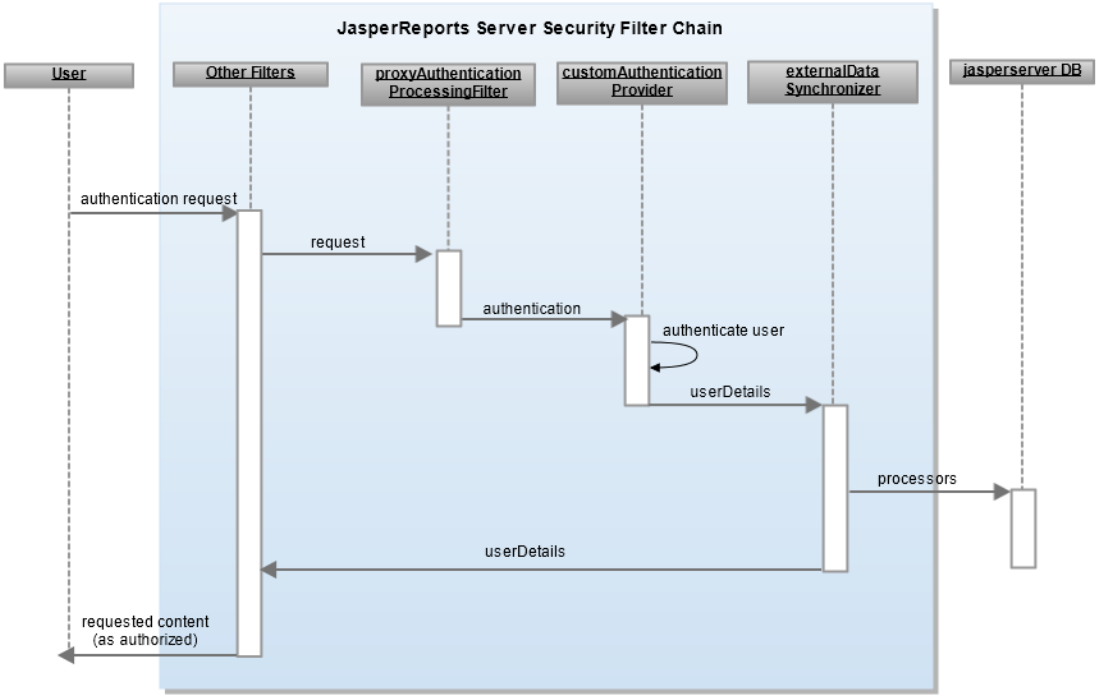

Figure 7-3 Sequence Diagram for Authentication Based on the Request

To set up authentication based on the request:

1. Modify `CustomAuthenticationProcessingFilter.java` to work with your authentication method. This class takes a single `HttpServletRequest` parameter and returns a Spring `Authentication` object. You can use one of Spring's implementations of `Authentication` or our `CustomAuthenticationToken`. Your user request needs to have sufficient information for your custom authentication method to authenticate using the request.
2. Create a `myCustomProvider` class implementing `AuthenticationProvider`. In this class you must use the authentication object created in the previous step. For more information, refer to the documentation for Spring Security, as described in **1.2, "Spring Security," on page 8**.
3. In sample-applicationContext-template.xml, add `myCustomProvider` to the providers list in `customAuthenticationManager`. Your provider should authenticate using the object returned by `CustomAuthenticationProcessingFilter`.

```
<bean id="customAuthenticationManager" class="com.jaspersoft.jasperserver.api.security.
      externalAuth.wrappers.spring.JSProviderManager">
  <property name="providers">
    <list>
     <ref bean="${bean.myCustomProvider}"/>
     <ref bean="${bean.daoAuthenticationProvider}"/>
   </list>
 </property>
</bean>
```

4. Comment out or remove the sample provider.

```
/* <bean id="customAuthenticationProvider  class="com.jaspersoft.jasperserver.api.security.
      externalAuth.custom.CustomAuthenticationProvider"/> */
```

5. Set up your processors to work with your users and organizations. You can use the processors for LDAP or CAS as examples.
6. Copy the modified file to the WEB-INF folder and remove the sample- prefix.

7.5 Other Customizations

Spring Security and JasperReports Server are complex systems that allow many different opportunities for customization, for example:

- Modifying the authentication provider to:
  - Locate external users in different ways.
  - Extract additional user information from the external authority.
  - Implement custom mappings of organizations and roles.
- Writing a new authentication provider to implement external authentication for a new protocol.
- Modifying or replacing filters to perform synchronization differently.
- Writing additional filters to take further action during the login sequence.

As with any design, you must determine the benefits and drawbacks in the areas of complexity, cost, and maintenance of your implementation to help you decide your preferred approach.

The best way to customize is to copy one of the sample files and make modifications. Details on customization are beyond the scope of this guide.

GLOSSARY

Ad Hoc Editor

The interactive data explorer in JasperReports Server Professional and Enterprise editions. Starting from a predefined collection of fields, the Ad Hoc Editor lets you drag and drop fields, dimensions, and measures to explore data and create tables, charts, and crosstabs. These Ad Hoc views can be saved as reports.

Ad Hoc Report

In previous versions of JasperReports Server, a report created through the Ad Hoc Editor. Such reports could be added to dashboards and be scheduled, but when edited in Jaspersoft Studio, lost their grouping and sorting. In the current version, the Ad Hoc Editor is used to explore views which in turn can be saved as reports. Such reports can be edited in Jaspersoft Studio without loss, and can be scheduled and added to dashboards.

Ad Hoc View

A view of data that is based on a Domain, Topic, or OLAP client connection. An Ad Hoc view can be a table, chart, or crosstab and is the entry point to analysis operations such as slice and dice, drill down, and drill through. **Compare OLAP View.** You can save an Ad Hoc view as a report in order to edit it in the interactive viewer, schedule it, or add it to a dashboard.

Aggregate Function

An aggregate function is one that is computed using a group of values; for example, Sum or Average. Aggregate functions can be used to create calculated fields in Ad Hoc views. Calculated fields containing aggregate functions cannot be used as fields or added to groups in an Ad Hoc view and should not be used as filters. Aggregate functions allow you to set a level, which specifies the scope of the calculation; level values include Current (not available for PercentOf), ColumnGroup, ColumnTotal, RowGroup, RowTotal, Total.

Amazon Web Services (AWS)

Cloud platform, used to provide and host a family of services, such as RDS, S3, and EC2.

Analysis View

See OLAP View.

Audit Archiving

To prevent audit logs from growing too large to be easily accessed, the installer configures JasperReports Server to move current audit logs to an archive after a certain number of days, and to delete logs in the archive after a certain age. The archive is another table in the JasperReports Server's repository database.

Audit Domains

A Domain that accesses audit data in the repository and lets administrators create Ad Hoc reports of server activity. There is one Domain for current audit logs and one for archived logs.

Audit Logging

When auditing is enabled, audit logging is the active recording of who used JasperReports Server to do what when. The system installer can configure what activities to log, the amount of detail gathered, and when to archive the data. Audit logs are stored in the same private database that JasperReports Server uses to store the repository, but the data is only accessible through the audit Domains.

Auditing

A feature of JasperReports Server Enterprise edition that records all server activity and allows administrators to view the data.

Calculated Field

In an Ad Hoc view or a Domain, a field whose value is calculated from a user-defined formula that may include any number of fields, operators, and constants. For Domains, a calculated field becomes one of the items to which the Domain's security file and locale bundles can apply. There are more functions available for Ad Hoc view calculations than for Domains.

CloudFormation (CF)

Amazon Web Services CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning, and updating them in an orderly and predictable fashion.

CRM

Customer Relationship Management. The practice of managing every facet of a company's interactions with its clientele. CRM applications help businesses track and support their customers.

CrossJoin

An MDX function that combines two or more dimensions into a single axis (column or row).

Cube

The basis of most OLAP applications, a cube is a data structure that contains three or more dimensions that categorize the cube's quantitative data. When you navigate the data displayed in an OLAP view, you are exploring a cube.

Custom Field

In the Ad Hoc Editor, a field that is created through menu items as a simple function of one or two available fields, including other custom fields. When a custom field becomes too complex or needs to be used in many reports, it is best to define it as a calculated field in a Domain.

Dashboard

A collection of reports, input controls, graphics, labels, and web content displayed in a single, integrated view. Dashboards often present a high level view of your data, but input controls can parametrize the data to display. For example, you can narrow down the data to a specific date range. Embedded web content, such as other web-based applications or maps, make dashboards more interactive and functional.

Dashlet

An element in a dashboard. Dashlets are defined by editable properties that vary depending on the dashlet type. Types of dashlet include reports, text elements, filters, and external web content.

Data Island

A single join tree or a table without joins in a Domain. A Domain may contain several data islands, but when creating an Ad Hoc view from a Domain, you can only select one of them to be available in the view.

Data Policy

In JasperReports Server, a setting that determines how the server processes and caches data used by Ad Hoc reports. Select your data policies by clicking **Manage > Server > Settings Ad Hoc Settings**. By default, this setting is only available to the superuser account.

Data Source

Defines the connection properties that JasperReports Server needs to access data. The server transmits queries to data sources and obtains datasets in return for use in filling reports and previewing Ad Hoc reports. JasperReports Server supports JDBC, JNDI, and Bean data sources; custom data sources can be defined as well.

Dataset

A collection of data arranged in columns and rows. Datasets are equivalent to relational results sets and the `JRDataSource` type in the JasperReports Library.

Datatype

In JasperReports Server, a datatype is used to characterize a value entered through an input control. A datatype must be of type text, number, date, or date-time. It can include constraints on the value of the input, for example maximum and minimum values. As such, a datatype in JasperReports Server is more structured than a datatype in most programming languages.

Denormalize

A process for creating table joins that speeds up data retrieval at the cost of having duplicate row values between some columns.

Derived Table

In a Domain, a derived table is defined by an additional query whose result becomes another set of items available in the Domain. For example, with a JDBC data source, you can write an SQL query that includes complex functions for selecting data. You can use the items in a derived table for other operations on the Domain, such as joining tables, defining a calculated field, or filtering. The items in a derived table can also be referenced in the Domain's security file and locale bundles.

Dice

An OLAP operation to select columns.

Dimension

A categorization of the data in a cube. For example, a cube that stores data about sales figures might include dimensions such as time, product, region, and customer's industry.

Domain

A virtual view of a data source that presents the data in business terms, allows for localization, and provides data-level security. A Domain is not a view of the database in relational terms, but it implements the same functionality within JasperReports Server. The design of a Domain specifies tables in the database, join clauses, calculated fields, display names, and default properties, all of which define items and sets of items for creating Ad Hoc reports.

Domain Topic

A Topic that is created from a Domain by the Data Chooser. A Domain Topic is based on the data source and items in a Domain, but it allows further filtering, user input, and selection of items. Unlike a JRXML-based Topic, a Domain Topic can be edited in JasperReports Server by users with the appropriate permissions.

Drill

To click on an element of an OLAP view to change the data that is displayed:

- Drill down. An OLAP operation that exposes more detailed information down the hierarchy levels by delving deeper into the hierarchy and updating the contents of the navigation table.
- Drill through. An OLAP operation that displays detailed transactional data for a given aggregate measure. Click a fact to open a new table beneath the main navigation table; the new table displays the low-level data that constitutes the data that was clicked.
- Drill up. An OLAP operation for returning the parent hierarchy level to view to summary information.

Eclipse

An open source Integrated Development Environment (IDE) for Java and other programming languages, such as C/C++.

ETL

Extract, Transform, Load. A process that retrieves data from transactional systems, and filters and aggregates the data to create a multidimensional database. Generally, ETL prepares the database that your reports will access. The Jaspersoft ETL product lets you define and schedule ETL processes.

Fact

The specific value or aggregate value of a measure for a particular member of a dimension. Facts are typically numeric.

Field

A field is equivalent to a column in the relational database model. Fields originate in the structure of the data source, but you may define calculated fields in a Domain or custom fields in the Ad Hoc Editor. Any type of field, along with its display name and default formatting properties, is called an item and may be used in the Ad Hoc Editor.

Frame

In Jaspersoft Studio, a frame is a rectangular element that can contain other elements and optionally draw a border around them. Elements inside a frame are positioned relative to the frame, not to the band, and when you move a frame, all the elements contained in the frame move together. A frame automatically stretches to fit its contents.

Group

In a report, a group is a set of data rows that have an identical value in a designated field.

- In a table, the value appears in a header and footer around the rows of the group, while the other fields appear as columns.
- In a chart, the field chosen to define the group becomes the independent variable on the X axis, while the other fields of each group are used to compute the dependent value on the Y axis.

Hierarchy Level

In an OLAP cube, a member of a dimension containing a group of members.

Input Control

A button, check box, drop-down list, text field, or calendar icon that allows users to enter a value when running a report or viewing a dashboard that accepts input parameters. For JRXML reports, input controls and their associated datatypes must be defined as repository objects and explicitly associated with the report. For Domain-based reports that prompt for filter values, the input controls are defined internally. When either type of report is used in a dashboard, its input controls are available to be added as special content.

Item

When designing a Domain or creating a Topic based on a Domain, an item is the representation of a database field or a calculated field along with its display name and formatting properties defined in the Domain. Items can be grouped in sets and are available for use in the creation of Ad Hoc reports.

JasperReport

A combination of a report template and data that produces a complex document for viewing, printing, or archiving information. In the server, a JasperReport references other resources in the repository:

- The report template (in the form of a JRXML file)
- Information about the data source that supplies data for the report
- Any additional resources, such as images, fonts, and resource bundles referenced by the report template.

The collection of all the resources that are referenced in a JasperReport is sometimes called a report unit. End users usually see and interact with a JasperReport as a single resource in the repository, but report creators must define all of the components in the report unit.

JasperReports IO

An HTTP-based reporting service for JasperReports Library that provides a REST API for running, exporting, and interacting with reports and a JavaScript API for embedding reports and their input controls into your web pages and web applications.

JasperReports Library

An embeddable, open source, Java API for generating a report, filling it with current data, drawing charts and tables, and exporting to any standard format (HTML, PDF, Excel, CSV, and others). JasperReports processes reports defined in JRXML, an open XML format that allows the report to contain expressions and logic to control report output based on run-time data.

JasperReports Server

A commercial open source, server-based application that calls the JasperReports Library to generate and share reports securely. JasperReports Server authenticates users and lets them upload, run, view, schedule, and send reports from a web browser. Commercial versions provide metadata layers, interactive report and dashboard creation, and enterprise features such as organizations and auditing.

Jaspersoft Studio

A commercial open source tool for graphically designing reports that leverage all features of the JasperReports Library. Jaspersoft Studio lets you drag and drop fields, charts, and sub-reports onto a canvas, and also define parameters or expressions for each object to create pixel-perfect reports. You can generate the JRXML of the report directly in Jaspersoft Studio, or upload it to JasperReports Server. Jaspersoft Studio is implemented in Eclipse.

Jaspersoft ETL

A graphical tool for designing and implementing your data extraction, transforming, and loading (ETL) tasks. It provides hundreds of data source connectors to extract data from many relational and non-relational systems.

Then, it schedules and performs data aggregation and integration into data marts or data warehouses that you use for reporting.

Jaspersoft OLAP

A relational OLAP server integrated into JasperReports Server that performs data analysis with MDX queries. The product includes query builders and visualization clients that help users explore and make sense of multidimensional data. Jaspersoft OLAP also supports XML/A connections to remote servers.

Jaspersoft Studio

An open source tool for graphically designing reports that leverage all features of the JasperReports Library. Jaspersoft Studio lets you drag and drop fields, charts, and sub-reports onto a canvas, and also define parameters or expressions for each object to create pixel-perfect reports. You can generate the JRXML of the report directly in Jaspersoft Studio, or upload it to JasperReports Server. Jaspersoft Studio is implemented in Eclipse.

JavaBean

A reusable Java component that can be dropped into an application container to provide standard functionality.

JDBC

Java Database Connectivity. A standard interface that Java applications use to access databases.

JNDI

Java Naming and Directory Interface. A standard interface that Java applications use to access naming and directory services.

Join Tree

In Domains, a collection of joined tables from the actual data source. A join is the relational operation that associates the rows of one table with the rows of another table based on a common value in given field of each table. Only the fields in a same join tree or calculated from the fields in a same join tree may appear together in a report.

JPivot

An open source graphical user interface for OLAP operations. For more information, visit http://jpivot.sourceforge.net/.

JRXML

An XML file format for saving and sharing reports created for the JasperReports Library and the applications that use it, such as Jaspersoft Studio and JasperReports Server. JRXML is an open format that uses the XML standard to define precisely all the structure and configuration of a report.

Level

Specifies the scope of an aggregate function in an Ad Hoc view. Level values include Current (not available for PercentOf), ColumnGroup, ColumnTotal, RowGroup, RowTotal, Total.

MDX

Multidimensional Expression Language. A language for querying multidimensional objects, such as OLAP (On Line Analytical Processing) cubes, and returning cube data for analytical processing. An MDX query is the query that determines the data displayed in an OLAP view.

Measure

Depending on the context:

- In a report, a formula that calculates the values displayed in a table's columns, a crosstab's data values, or a chart's dependent variable (such as the slices in a pie).

- In an OLAP view, a formula that calculates the facts that constitute the quantitative data in a cube.

Mondrian

A Java-based, open source multidimensional database application.

Mondrian Connection

An OLAP client connection that consists of an OLAP schema and a data source. OLAP client connections populate OLAP views.

Mondrian Schema Editor

An open source Eclipse plug-in for creating Mondrian OLAP schemas.

Mondrian XML/A Source

A server-side XML/A source definition of a remote client-side XML/A connection used to populate an OLAP view using the XML/A standard.

MySQL

An open source relational database management system. For information, visit http://www.mysql.com/.

Navigation Table

The main table in an OLAP view that displays measures and dimensions as columns and rows.

ODBO Connect

Jaspersoft ODBO Connect enables Microsoft Excel 2003 and 2007 Pivot Tables to work with Jaspersoft OLAP and other OLAP servers that support the XML/A protocol. After setting up the Jaspersoft ODBO data source, business analysts can use Excel Pivot Tables as a front-end for OLAP analysis.

OLAP

On Line Analytical Processing. Provides multidimensional views of data that help users analyze current and past performance and model future scenarios.

OLAP Client Connection

A definition for retrieving data to populate an OLAP view. An OLAP client connection is either a direct Java connection (Mondrian connection) or an XML-based API connection (XML/A connection).

OLAP Schema

A metadata definition of a multidimensional database. In Jaspersoft OLAP, schemas are stored in the repository as XML file resources.

OLAP View

Also called an analysis view. A view of multidimensional data that is based on an OLAP client connection and an MDX query. Unlike Ad Hoc views, you can directly edit an OLAP view's MDX query to change the data and the way they are displayed. An OLAP view is the entry point for advanced analysis users who want to write their own queries. **Compare Ad Hoc View.**

Organization

A set of users that share folders and resources in the repository. An organization has its own user accounts, roles, and root folder in the repository to securely isolate it from other organizations that may be hosted on the same instance of JasperReports Server.

Organization Admin

Also called the organization administrator. A user in an organization with the privileges to manage the organization's user accounts and roles, repository permissions, and repository content. An organization admin can also create suborganizations and mange all of their accounts, roles, and repository objects. The default organization admin in each organization is the `jasperadmin` account.

Outlier

A fact that seems incongruous when compared to other member's facts. For example, a very low sales figure or a very high number of help desk tickets. Such outliers may indicate a problem (or an important achievement) in your business. The analysis features of Jaspersoft OLAP excel at revealing outliers.

Parameter

Named values that are passed to the engine at report-filling time to control the data returned or the appearance and formatting of the report. A report parameter is defined by its name and type. In JasperReports Server, parameters can be mapped to input controls that users can interact with.

Pivot

To rotate a crosstab such that its row groups become column groups and its column groups become rows. In the Ad Hoc Editor, pivot a crosstab by clicking  .

Pivot Table

A table with two physical dimensions (for example, X and Y axis) for organizing information containing more than two logical dimensions (for example, PRODUCT, CUSTOMER, TIME, and LOCATION), such that each physical dimension is capable of representing one or more logical dimensions, where the values described by the dimensions are aggregated using a function such as SUM. Pivot tables are used in Jaspersoft OLAP.

Properties

Settings associated with an object. The settings determine certain features of the object, such as its color and label. Properties are normally editable. In Java, properties can be set in files listing objects and their settings.

Report

In casual usage, *report* may refer to:

- A JasperReport. **See JasperReport.**
- The main JRXML in a JasperReport.
- The file generated when a JasperReport is scheduled. Such files are also called content resources or output files.
- The file generated when a JasperReport is run and then exported.
- In previous JasperReports Server versions, a report created in the Ad Hoc Editor. **See Ad Hoc Report.**

Report Run

An execution of a report, Ad Hoc view, or dashboard, or a view or dashboard designer session, it measures and limits usage of Freemium instances of JasperReports Server. The executions apply to resources no matter how they are run (either in the web interface or through the various APIs, such as REST web services). Users of our Community Project and our full-use commercial licenses are not affected by the limit. For more information, please contact sales@jaspersoft.com.

Repository

Depending on the context:

- In JasperReports Server, the repository is the tree structure of folders that contain all saved reports, dashboards, OLAP views, and resources. Users access the repository through the JasperReports Server web interface or through Jaspersoft Studio. Applications can access the repository through the web service API. Administrators use the import and export utilities to back up the repository contents.
- In JasperReports IO, the repository is where all the resources needed to create and run reports are stored. The repository can be stored in a directory on the host computer or in an S3 bucket hosted by Amazon Web Services. Users access the repository through a file browser on the host machine or through the AWS console.

Resource

In JasperReports Server, anything residing in the repository, such as an image, file, font, data source, Topic, Domain, report element, saved report, report output, dashboard, or OLAP view. Resources also include the folders in the repository. Administrators set user and role-based access permissions on repository resources to establish a security policy.

Role

A security feature of JasperReports Server. Administrators create named roles, assign them to user accounts, and then set access permissions to repository objects based on those roles. Certain roles also determine what functionality and menu options are displayed to users in the JasperReports Server interface.

S3 Bucket

Cloud storage system for Amazon Web Services. JasperReports IO can use an S3 bucket to store files for its repository.

Schema

A logical model that determines how data is stored. For example, the schema in a relational database is a description of the relationships between tables, views, and indexes. In Jaspersoft OLAP, an OLAP schema is the logical model of the data that appears in an OLAP view; they are uploaded to the repository as resources. For Domains, schemas are represented in XML design files.

Schema Workbench

A graphical tool for easily designing OLAP schemas, data security schemas, and MDX queries. The resulting cube and query definitions can then be used in Jaspersoft OLAP to perform simple but powerful analysis of large quantities of multi-dimensional data stored in standard RDBMS systems.

Set

In Domains and Domain Topics, a named collection of items grouped together for ease of use in the Ad Hoc Editor. A set can be based on the fields in a table or entirely defined by the Domain creator, but all items in a set must originate in the same join tree. The order of items in a set is preserved.

Slice

An OLAP operation for filtering data rows.

SQL

Structured Query Language. A standard language used to access and manipulate data and schemas in a relational database.

Stack

A collection of Amazon Web Services resources you create and delete as a single unit.

System Admin

Also called the system administrator. A user who has unlimited access to manage all organizations, users, roles, repository permissions, and repository objects across the entire JasperReports Server instance. The system admin can create root-level organizations and manage all server settings. The default system admin is the `superuser` account.

Topic

A JRXML file created externally and uploaded to JasperReports Server as a basis for Ad Hoc reports. Topics are created by business analysts to specify a data source and a list of fields with which business users can create reports in the Ad Hoc Editor. Topics are stored in the Ad Hoc Components folder of the repository and displayed when a user launches the Ad Hoc Editor.

Transactional Data

Data that describe measurable aspects of an event, such as a retail transaction, relevant to your business. Transactional data are often stored in relational databases, with one row for each event and a table column or field for each measure.

User

Depending on the context:

- A person who interacts with JasperReports Server through the web interface. There are generally three categories of users: administrators who install and configure JasperReports Server, database experts or business analysts who create data sources and Domains, and business users who create and view reports and dashboards.
- A user account that has an ID and password to enforce authentication. Both people and API calls accessing the server must provide the ID and password of a valid user account. Roles are assigned to user accounts to determine access to objects in the repository.

View

Several meanings pertain to JasperReports Server:

- An Ad Hoc view. **See Ad Hoc View.**
- An OLAP view. **See OLAP View.**
- A database view. See http://en.wikipedia.org/wiki/View_%28database%29.

Virtual Data Source

A virtual data source allows you to combine data residing in multiple JDBC and/or JNDI data sources into a single data source that can query the combined data. Once you have created a virtual data source, you create Domains that join tables across the data sources to define the relationships between the data sources.

WCF

Web Component Framework. A low-level GUI component of JPivot. For more information, see http://jpivot.sourceforge.net/wcf/index.html.

Web Services

A SOAP (Simple Object Access Protocol) API that enables applications to access certain features of JasperReports Server. The features include repository, scheduling and user administration tasks.

XML

eXtensible Markup language. A standard for defining, transferring, and interpreting data for use across any number of XML-enabled applications.

XML/A

XML for Analysis. An XML standard that uses Simple Object Access protocol (SOAP) to access remote data sources. For more information, see http://www.xmla.org/.

XML/A Connection

A type of OLAP client connection that consists of Simple Object Access Protocol (SOAP) definitions used to access data on a remote server. OLAP client connections populate OLAP views.

INDEX

R

S

T

U