TIBCO JASPERREPORTS® SERVER COMMUNITY PROJECT ADMINISTRATOR GUIDE

RELEASE 6.4

TABLE OF CONTENTS

CHAPTER 1 OVERVIEW OF JASPERREPORTS SERVER ADMINISTRATION

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.

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.

This administrator guide describes features that are only available to users who have administrator roles. Many of the configuration procedures also assume you have access to the installed files on the host computer.

• 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.

This chapter contains the following sections:

• **Overview of the Repository**
• **Overview of Users and Roles**
• **Overview of Security**
• **Administrator Login**
• **Administrator Pages**

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.

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 charts, dashboards, Domains, auditing, and a multi-organization architecture for hosting large BI deployments. This guide documents the features of the community project.

1.1   Overview of the Repository

The repository is a hierarchical structure of folders where administrators and users create and store resources for running reports. In its appearance and function, the repository resembles a file system with a structure of folders containing files. However, the repository is actually implemented as a database that is private to the server instance. As a result, it lacks a few of the functions of a file system.

1.1.1   Folder Structure

The repository is a tree structure composed of folders and resources. The root contains folders for the sample data installed with the server.
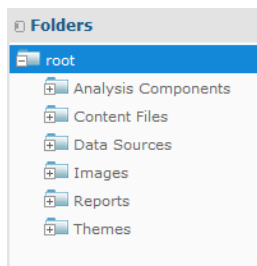
Figure 1-1 Root of the Repository Showing Default Folders

In general, we recommend that you avoid placing resources directly in the root. Instead, use folders for various resource types, as in the sample data. The Themes folder contains files that control the look and feel of the user interface, as described in **Chapter 6, "Themes," on page 97**.

1.1.2 Resources

Resources are stored in the repository and used as input for creating reports and performing analysis. Some resources, such as images, fonts, or JRXML files created in Jaspersoft Studio, are uploaded from files. Others, such as data sources and Domains, are created in JasperReports Server itself. Of course, dashboards, data views, and reports can also be saved in the repository to be run as often as needed. Output such as PDF or HTML can be saved in the repository as well.

All resources, including folders, have unique IDs, names, and optional descriptions. The ID of a resource, along with the ID of its enclosing folders create the path used to reference that resource. The path of a resource is also called its *repository URI* (Universal Resource Indicator). The name and description appear in the user interface when you browse or search the repository.

Resources are stored in an internal format not accessible to users or administrators. Any resource can be exported with the js-export utility, but the resulting files are for backup or transfer to another JasperReports Server instance and cannot be modified.

JasperReports Server restricts access to folders and resources based on organizations, permissions, user names, and roles. For more information, see **Chapter 3, "Repository Administration," on page 31**.

The sample data includes dashboards, reports, Domains, data sources, and many of their components, such as input types and image files. Each type of content is stored in a separate folder, making it easy to locate. The Supermart Demo folder contains a complete example of dashboards, reports, and resources for various business scenarios in a fictional grocery store company.

1.1.3 Browsing and Searching

Users and administrators can browse or search the repository, depending on what action they want to perform and how resources are organized. Searching the repository finds specific resources faster. For more information on browsing and searching the repository, see the *JasperReports Server User Guide*.

- Browsing - On the Home page, click **View > Repository**.
  The Folders panel on the left lists the folders in the repository and the Repository panel lists the contents of the selected folder. The tool bar in the Repository panel allows administrators to perform actions such as **Copy**, **Cut**, **Paste**, and **Delete**; select several resources with Control-click or Shift-click to perform actions in bulk.

Figure 1-2 Browsing the Repository

- Searching - Enter a search term in the search field at the top of any page, or select **View > Search Results**.

Figure 1-3 Search Field in the Menu Bar

The search results page displays a search field with the current search term at the top of the Repository panel. The search applies to resource IDs, names, and descriptions. Use the filters in the left-hand panel to refine your search.

Figure 1-4 Searching the Repository

1.2 Overview of Users and Roles

User accounts and role membership provide authentication and authorization for access control in JasperReports Server. When logging, in users enter their username and password to access the server. Administrators assign named roles to users and then create role-based permissions to further control access to the repository.

Administrators define permissions directly on the resources and folders in the repository. You can define a level of access, such as read-write, read-only, or no access, and assign each permission based on either a username or a role.

1.2.1   Administering Users and Roles

Administrators perform the following actions to manage users:

- Create, modify, and delete users.
- Set user account properties such as name, email, and attributes.
- Reset a user password. However, no administrator can ever view a user's existing password.
- Login as any user to test permissions.
- Create, modify, and delete roles.
- Assign roles to users.
- Set access permissions on repository folders and resources.

1.2.2   Delegated Administration

JasperReports Server enables two levels of delegated administration:

- The Administer permission allows a user to view and set permissions on a folder or resource. This can allow a power-user to manage a section of the repository, but not to create or manage users.
- Granting ROLE_ADMINISTRATOR allows a user to see the management interface and create users and roles. This is true delegated administration, whereby a user other than jasperadmin has administration abilities.

1.3   Overview of Security

JasperReports Server ensures that people can access only the data they are allowed to see. The mechanisms that define users, roles, and repository resources work together to provide complete access control. Security has many facets covered in this guide and other guides:

Authentication	Authentication is the process of restricting access to identified users. Users must log in with their user ID and password so that they have an identity in JasperReports Server. The server stores user definitions, including encrypted passwords, in a private database. Administrators create, modify, and delete user accounts through the administrator pages, as described in **2.1, "Managing Users," on page 17**.
Password policies	Every company must establish a password policy that weighs its security risks against the convenience of its users. JasperReports Server supports many different password policies such as password expiration, reuse, and strong patterns. This configuration is described in the *JasperReports Server Security Guide*.
External authentication	External authentication uses centralized identity services such as LDAP (used by Microsoft Active Directory and Novell eDirectory), Central Authentication Service (CAS), or Java Authentication and Authorization Service (JAAS). For more information, see the *JasperReports Server External Authentication Cookbook*.

Application Security	System admins who install and maintain enterprise software know they must protect their servers against hackers. JasperReports Server protects your data against intruders with many protocols and tools, such as HTTPS, encryption, CSRF prevention, and input validation against cross-site scripting and SQL injection. For these topics and others, see the *JasperReports Server Security Guide*.
Roles	JasperReports Server also implements roles that can be assigned to any number of users. Roles let administrators create groups or classes of users that are granted similar permissions. A user may belong to any number of roles and receive the privileges from each of them. Administrators create, modify, and delete roles through the administrator pages, as described in **2.2, "Managing Roles," on page 21**.
Resource permissions	Administrators can define access permissions on every folder and resource in the repository. Permissions are enforced when accessing any resource either directly through the repository interface, indirectly when called from a report, or programmatically through the web services.
	Permissions can be defined for every role and every user, or they can be left undefined so they are inherited from the parent folder. To restrict access or hide resources such as database connections, the administrator can set no-access or execute-only permission. For more information, see **3.4, "Repository Permissions," on page 41**.
Administrator privileges	JasperReports Server distinguishes between administrators and users. Administrators are granted access to the UI for permissions, user management, and sensitive resources such as database connections. Administrators also set the UI appearance with themes. Regular users are restricted to the folders, reports, and dashboards that admins allow them to access. Most of the features in this guide are not accessible to regular users. See **1.2.2, "Delegated Administration," on page 13**.
Menus and pages	The menus that appear in JasperReports Server depend on the user's roles. For example, only users with the administrator role can see the **Manage** menu and access the administrator pages. By modifying the server's configuration, you can modify access to menus, menu items, and individual pages. Refer to the *JasperReports Server Community Project Source Build Guide* and *JasperReports Server Ultimate Guide* for more information.
Attributes	Attributes are name-value pairs associated with a user, organization, or server. Attributes can be used to restrict or enable a user's access to data in several ways. See **2.3, "Managing Attributes," on page 24**.

Administrators must keep security in mind at all times when managing user, roles, and resources, because effective security relies on all of them working together.

1.4   Administrator Login

Administrators log in on the standard login page, using the following default password:

Administrator: user ID `jasperadmin` and password `jasperadmin`

> For security reasons, always change the default administrator password immediately after installing JasperReports Server. For instructions, see **2.1.3, "Editing a User," on page 19**.

For more information about options on the Login page and logging in with multiple organizations, see the *JasperReports Server User Guide*.

The first time you log in as an administrator, you may be prompted to opt-into the Heartbeat program. You should also set the administrator passwords and email.

1.4.1   JasperReports Server Heartbeat

When you log into JasperReports Server for the first time after installation, administrators may be prompted to opt into the server's Heartbeat program. It reports specific information to Jaspersoft about your implementation: the operating system, JVM, application server, database (type and version), and JasperReports Server edition and version number. By tracking this information, we can build better products that function optimally in your environment. No personal information is collected.

To opt into the program, click **OK**. To opt out, clear the check box then click **OK**.

1.4.2   Administrator Email

After logging in for the first time, you should set the email on the `jasperadmin` account to your email address. In rare cases, the server may notify you by email about issues with your license.

> This is also a good time to change the default passwords on the `jasperadmin`account.

To set the email and passwords on the administrator account, edit the user account information as described in **2.1.3, "Editing a User," on page 19**.

1.5 Administrator Pages

Administrators have access to special pages to manage the server. After logging in, select an item from the **Manage** menu on any page.
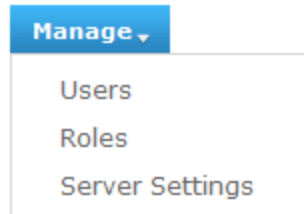
Figure 1-5 Manage Menu for Administrators

The pages for administering users and roles are documented in **Chapter 2, "User and Role Management," on page 17**. The pages for administering Server Settings are documented in **8.1, "Configuration Settings in the User Interface," on page 126**.

The **About TIBCO JasperReports Server** link in the footer of all pages displays the product version along with the software build.

Figure 1-6 About TIBCO JasperReports Server Dialog

CHAPTER 2   USER AND ROLE MANAGEMENT

Administrators create users, assign them to roles, and optionally define attributes for them. The interface in JasperReports Server for managing users and roles makes it easy to search among hundreds of users and roles and edit their properties.

Attributes are name-value pairs that can be defined on users and at the server level. These values can provide flexibility, for example allowing each user to access a different database when using the same data source. Attributes can provide data-level security when combined with the features of Domains and OLAP.

This chapter contains the following sections:

*   **Managing Users**
*   **Managing Roles**
*   **Managing Attributes**

2.1   Managing Users

Administrators create and manage all users in the server, including creating other administrators, as described in **1.2.2, "Delegated Administration," on page 13**.

The default installation of JasperReports Server includes the following users:

Table 2-1 Default Users after Installation

User Name	Default Password (case sensitive)	Description
jasperadmin	jasperadmin	Default administrator.
joeuser	joeuser	Default end user.
anonymousUser	anonymoususer	Allows anonymous login; disabled by default. If you do not allow anonymous access, this user can be deleted.

You should advise your users to change their passwords regularly. To configure periodic expiration of passwords, refer to the *JasperReports Server Security Guide*.

2.1.1 Viewing User Properties

1. Log in as an administrator (`jasperadmin`).
2. Select **Manage > Users** . The Manage Users page displays the users and properties of the selected user.
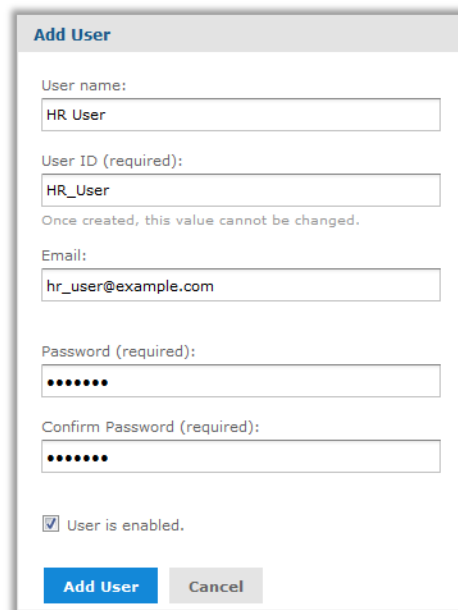
Figure 2-1 **Manage Users Page**

The columns in the Users panel list the user ID and the user name of each user in the server

3. To locate a user:
   - Browse for users – Scroll through the list of users if it is too long to fit on your screen.
   - Search for a user – Enter a search string in the **Search** field of the Users panel. The search results show all user ID or names that match the search string.
4. Select a user account to view its Properties in the right-hand panel.

   User status can be **Enabled** or **Disabled**; disabled users are displayed in gray text in the Users list. For convenience, the role names link to the role management page for each role. For information about attributes on the user, see **2.3.6, "Managing User Attributes," on page 27**.

2.1.2 Creating a User

1. Log in as an administrator (`jasperadmin`).
2. Select **Manage > Users**.
3. Click **Add User**. The Add User dialog appears.

Figure 2-2 **Adding a User**

4. Enter the following information:
    - User name – The new user's full name. The name is optional but recommended; It will appear in the menu bar of the UI when the user is logged in.
    - User ID – Generated automatically from the user name; you can accept the suggested value or type your own. The user ID is used to log into JasperReports Server, and for administrators to manage users and resources. The User ID must be unique.
    - Email – This is optional but must be in a valid email format.
    - Password and confirmation – Enter and confirm a password for the user.
    - User is enabled – To enable the user to log in, select this check box. Users who are not enabled can't log in. If you implement role-based permissions, you might want to delay enabling the user until you assign more roles. For more information on roles, see **2.2, "Managing Roles," on page 21**.
5. Click **Add User**.

    The new user is available in the Users panel. To assign roles to the user, click **Edit** in the user's Properties panel.

2.1.3   Editing a User

One way to assign roles to a user is to edit the user's properties. Alternatively, when you edit a role, you can assign it to any number of users. To edit a user's properties:

1. Log in as an administrator (jasperadmin).
2. Click **Manage > Users**.
3. In the Users panel, select the user.
4. In the user's Properties panel, click **Edit**.

Figure 2-3 **Editing the Properties of a User**

5. Edit the user's properties as needed. You can't edit the user ID; it always has the value defined when the user was created originally.

6. To assign or remove roles from the user, select the roles, and use the arrow buttons between the Roles Available and Roles Assigned lists.

7. For information about attributes on the user, see **2.3.6, "Managing User Attributes," on page 27**.

8. Click **Save** to keep your changes.

9. In the Properties panel, click **Login as User** to test the user's permissions, as explained in **3.4.7, "Testing User Permissions," on page 46**.

    Logging in as another user is also necessary when you are maintaining resources that use absolute references in the repository. For more information, see **"Referencing Resources in the Repository" on page 1**.

2.1.4   Enabling or Disabling Multiple Users

You may sometimes need to disable user accounts. For example, when making configuration changes, you may want to lock out all users until the changes are finished. The administrator (jasperadmin) can select any number or all of the users, except himself.

1. Log in as an administrator (jasperadmin).

2. Click **Manage > Users**.

3. In the Users list, use Control-click and Shift-click to make multiple selections. If the User list is too long, enter a search term to find users and enable or disable them individually.

4. Click **Enable** or **Disable** in the menu bar.

2.1.5 Deleting One or More Users

1. Log in as an administrator (`jasperadmin`).

2. Click **Manage > Users**.

3. In the Users list, use Control-click and Shift-click to make multiple selections. If the list of users is too long, enter a search term to find and select the user.

4. In the tool bar of the Users panel, click **Delete** and confirm the action.

2.2 Managing Roles

Roles define sets of users who are granted similar permissions. Administrators create roles, assign them to users, and set permissions in the repository (see **3.4, "Repository Permissions," on page 41**). By default, JasperReports Server includes the following roles; some are needed for system operation, some are included as part of the sample data:

Table 2-2 Default Roles in JasperReports Server Installations

Role	Description
ROLE_ADMINISTRATOR	This role determines administrator privileges. This role is automatically assigned to the default `jasperadmin` user. It's a special system-level role, and administrators can assign it to other users, as explained in **1.2.2, "Delegated Administration," on page 13**. Never delete this role, it's required for proper administration of the server.
ROLE_USER	Required to log in. This role is automatically assigned to every user in the server. It's a special system-level role. Never delete this role, it's required to create users and allow them to log in.
ROLE_ANONYMOUS	When anonymous access is enabled, this role is automatically assigned to any agent accessing the server without logging in. It's a special system-level role. This role is also assigned to the default anonymous user. By default, anonymous access is disabled and this role isn't used. It's a system role that even administrators can't delete.

When you need to define permissions for sets of users, administrators can create new roles and assign them to users. Users can belong to any number of roles and each can be used for access to different resources.

2.2.1 Viewing Role Properties

1. Log in as an administrator (`jasperadmin`).

2. Select **Manage > Roles**. The Manage Roles page displays the roles defined in the server and properties for each role.
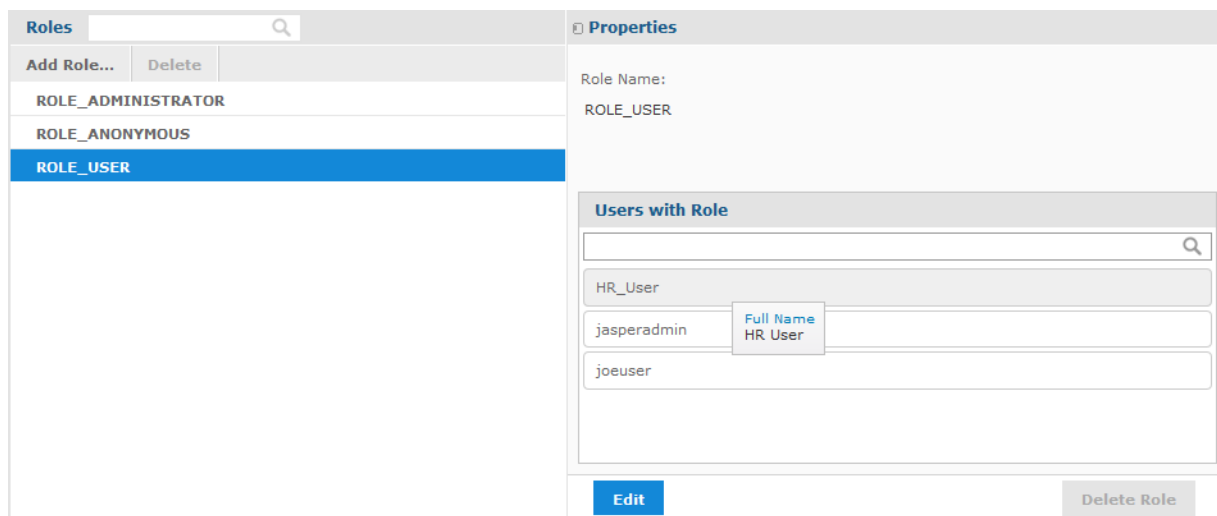
Figure 2-4 **Manage Roles Page**

3. To filter the list of roles, enter a search string in the search field of the Roles panel. The search results show all of the roles whose names contain the search string. If necessary, scroll through the new list or refine your search.

4. Select the role in the Roles panel. The role's properties appear in the Properties panel.

   The Properties panel shows the role name and the users assigned to the role. You can enter a search term to find users in the list. Hover over a user ID to see a user's full name, as shown in the figure.

2.2.2   Creating a Role

1. Log in as an administrator (`jasperadmin`).
2. Select **Manage > Roles**.
3. Click **Add Role**. The Add Role dialog appears.
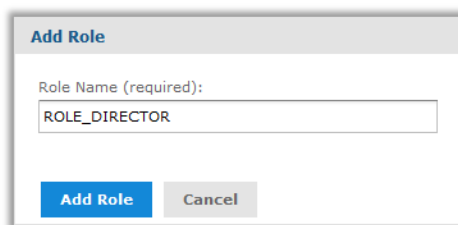
Figure 2-5 **Adding a Role**

4. Enter the name of the role. The role name is also the role ID and does not accept spaces or special characters.

5. Click **Add Role** to create the role.

   The new role is included in the Roles panel. If you want to assign users to the role, click **Edit** in the Properties panel of the new role.

2.2.3 Assigning Users to a Role

You can assign multiple users to one role. To assign multiple roles to one user, edit the user's properties as described in **2.1.3, "Editing a User," on page 19**.

1. Log in as an administrator (`jasperadmin`).
2. Select **Manage > Roles**.
3. Select the role in the Roles panel.
4. In the Properties panel, click **Edit**. The role's properties become editable.

Figure 2-6 Editing the Members of a Role

5. Enter a different name to change the role name throughout the server.

> Permissions in the repository that use the role name are automatically updated. However, role names in security files for Domains and OLAP are *not* updated with the new role name and may cause a security risk. If you use security files for Domains or OLAP, do not change role names without verifying the files as well. For more information, see the *JasperReports Server User Guide*.

6. To assign or remove role users, select the users, and click the arrow buttons between the Users Available and Users Assigned lists. You can enter a search term to find users in the lists. Hover over a user ID to see a user's full name, as shown in the figure.
7. Click **Save** to keep your changes, or **Cancel** to quit without saving.

2.2.4 Deleting One or More Roles

1. Log in as an administrator (`jasperadmin`).
2. Select **Manage > Roles**.
3. Select the role in the Roles panel. Use Control-click and Shift-click to make multiple selections.
4. In the tool bar of the Roles panel, click **Delete** and confirm the action.

2.3   Managing Attributes

Attributes are name-value pairs that are associated with users and the server itself. Attribute values can be used to parameterize access to data sources OLAP data, and report output. Attributes were previously known as profile attributes.

As of JasperReports Server release 6.1, attributes also have permissions, so that administrators can restrict the visibility and use of specific attributes.

> The permission feature of attributes is designed to work with multiple organizations in commercial editions of JasperReports Server. The feature is enabled in all editions of the server, but has limited use in the Community Project edition.

2.3.1   Referencing Attributes

There are several features of JasperReports Server that can read attributes and use attribute values. The following features can limit or restrict access to data or take different behavior based on the user or server-level attributes. Each feature reads or references attributes in its own syntax that is described with that feature:

- Data sources – The host name, port number, database name, user name and user password can be defined through attributes. See **4.1, "Attributes in Data Source Definitions," on page 48**.
- Reports – Attribute values can be referenced in calculations as described in **Table 5-2, "Attribute-based Parameters for Queries and Reports," on page 85**.
- OLAP schemas – Similar to Domains, access rules can be defined with attributes so that users may only see data they are allowed to access. See the *Jaspersoft OLAP User Guide*.

Attributes are always referenced in relation to the currently-logged in user who is performing an operation or running a report:

- If the value of a user-level attribute is requested, it refers to the attributes of the logged-in user.
- If the value of a server-level attribute is requested, the value depends on which server the user is logged into. For example, test and production servers may have a copy of the same repository, but different server-level attributes. For a given server, server-level attributes are shared and thus have the same value for all users.

2.3.2   Attribute Hierarchy

An attribute is a named value that is defined on a user or root of the server. Any number of attributes with any name can be defined at either or both of these levels. The definition of each attribute at each level is independent of other attributes at other level. Attributes at different levels may have the same name, unless explicitly forbidden as described in **2.3.4, "Attribute Permissions," on page 25**.

In the places that you reference attributes (see above), there are two ways to determine the value of an attribute:

- Categorical value – References the value of a named attribute at a specific level, either user level or server level. If the attribute does not exist at the requested level, no value is returned.
- Hierarchical value – References the value of a named attribute across all levels, starting at the user level. The server searches for an attribute with the given name in the following order, stopping and returning the first value that it finds:
  - At the user level, in the user attributes of the logged-in user.
  - At the server level.
  - If the attributes does not exist at any level, no value is returned.

With hierarchical values, attributes with same name but different values may be defined at several levels and on many users. Each attribute is a distinct attribute definition, but the hierarchical value referencing takes into account the level and the presence of the definition for every given user.

To help administrators define attributes on users, the UI displays all attributes that are visible in the attribute hierarchy. If an attribute is defined at the server level, its hierarchical value is shown at the user level, and the attribute is called inherited. This allows the administrator to see all hierarchical values that are in effect in lower levels, either to override them in the hierarchy or leave them as inherited.

Categorical and hierarchical approaches usually involve different strategies for defining and naming attributes. When using categorical values, you must ensure that each and every user has the attribute defined, and you must handle the case when it is accidentally undefined, such as a new user. When using hierarchical values, you define attributes with the same name at different levels, such that each user may have a custom value, but then the server level provides a default value for all users. Each strategy is useful for different applications, depending on your needs to access and protect data. However, both strategies can co-exist and be used by at the same time by different resources, as long as the names of attributes used in each strategy or each resource are kept distinct.

2.3.3 Attribute Encryption

By default, an attribute and its value can be known to administrators throughout the server.

JasperReports Server provides two mechanisms to protect sensitive attributes:

*   Attribute encryption – Prevents the attribute value from ever being seen. Does *not* prevent the attribute name from being seen or the value from being redefined at a lower level. When an attribute value is encrypted, the value stored internally is encrypted and never decrypted in the user interface. The server decrypts the attribute only when it is referenced by one of the features of the server. The decrypted value is then used for internal operations and never visible to the user.

    For example, an encrypted attribute could store the password for a database. Once typed in and saved, the UI displays ***** as the value of the attribute. When a report uses that database, the server decrypts the password and sends it as part of the database protocol, so the user and the admin never see the password.

    Encrypted attributes are similar to user passwords in the server. Administrators can change the encrypted value to a new value, but they can never view the current value.

*   Attribute permissions – Prevents the attribute from being visible or redefined at a lower level. Attribute permissions are described in the next section.

To properly secure an attribute, you should make it encrypted so it cannot be seen and then set permissions so it cannot be overridden.

2.3.4 Attribute Permissions

> Attribute permissions were designed to work with multiple organizations in commercial editions of JasperReports Server. The feature has limited use in the Community Project edition.

The attribute permissions and their effects are as follows:

*   Administer – This is the default permission.
    Inthe Community Project edition, use this permission unless you want to disable an attribute with the no-access permission.

*   Read Only – Inthe Community Project edition, this permission has no effect.

- Execute Only – Inthe Community Project edition, this permission has no effect.
- No Access – The attribute is disabled and its value cannot be referenced. When referenced, an attribute with no-access permission returns an error.

2.3.5   Managing Server Attributes

If a given attribute name is not defined on a user, it can have a value at the server level to act as a default value. Server-level attributes may also be used as parameters in data source definitions. For example, the same data source may be imported to several servers, such as test and production servers, but a server-level attribute with the same name on each server makes the data source connect to a different database.

To view, create, modify, or delete server-level attributes:

1. Log in as an administrator (`jasperadmin`).
2. Select **Manage > Server Settings** and choose **Server Attributes** from the left-hand panel.

    Each server level attribute is listed with its value, encrypted status, and permission. Holding the pointer over an attribute name shows the description associated with the attribute. When an attribute value is encrypted, its value is shown as *** symbols.

Figure 2-7 **Managing Attributes at the Server Level**

3. To create a new server-level attribute, click **Add new attribute**. Enter the attribute name and value, as well as an optional description. If desired, set the permission from the drop-down list and select the Encrypt check box. Click **OK** to close the dialog and then click **Save** to submit the new attribute.

Figure 2-8 Adding a Server Attribute

4. To modify an attribute, click the edit icon ✏. In the edit dialog, modify the desired fields. Click **OK** to close the dialog and then click **Save** to modify the attribute.

   You can also modify the attribute's permission and encryption by using the drop down and check box in its table row. After confirming any changes, click **Save** to make them take effect.

   When modifying an attribute, be aware of the following:

   - Changing the name of an attribute is equivalent to deleting the original attribute and adding a new attribute with the same value. Because this may impact features that reference the attribute, you are asked to confirm the name change.
   - Be aware that changing the encryption or permission of an attribute can impact the visibility of an attribute and the features that might rely on referencing its value. Again, you are asked to confirm the change.
   - Removing encryption does not decrypt an encrypted attribute. To safeguard encrypted values, removing the encryption on an attribute also erases its value. You should give the attribute a new value by clicking the edit icon.

5. To delete an attribute at the server level, click the delete icon 🗑 in the attribute row and confirm the operation. Because this may impact features that reference attributes, you are asked to confirm the deletion. Click **Save** to make the deletion take effect.

2.3.6 Managing User Attributes

Attributes on users can provided additional information about a user and can restrict access to data OLAP schemas. Users may also have hierarchical attribute values inherited from server-level attributes. Attributes on users do not have permissions.

Users never see the attributes defined on their user profile; only administrators can view and set user attributes.

To view, create, modify, or delete user-level attributes:

1. Log in as an administrator (`jasperadmin`).

2. Click **Manage > Users**.

3. Select the user in the Users panel. The properties panel includes a table of all attributes for the user, both locally defined and inherited from the server level.

Figure 2-9 **Viewing Attributes on the Manage Users Page**

4. To create, modify, or delete the attributes on a user, click **Edit** in the right-hand column, then select the **Attributes** tab.

Figure 2-10 **Editing User Attributes**

5. You can filter attributes in the list to include only the inherited attributes or only the locally defined (not inherited) ones.

6. To create a new user attribute, click **Add new attribute**. Enter the attribute name and value, as well as an optional description. If the attribute value should not be visible to other administrators, select the Encrypt check box. Click **OK** to close the dialog and then click **Save** to submit the new attribute.

Figure 2-11 **Adding a User Attribute**

7. To modify an attribute, click the edit icon . In the edit dialog, modify the desired fields. Click **OK** to close the dialog and then click **Save** to modify the attribute.

You can also modify the attribute's encryption by using the and check box in its table row. After confirming any changes, click **Save** to make them take effect.

When modifying a user attribute, be aware of the following:

*   Inherited attributes belong to the server level and are shown at the user level to display the hierarchical attribute values. Any modification to an inherited attribute actually creates a local attribute definition with the modified parameters. In the hierarchy of attributes, the new attribute takes precedence over the previously inherited attribute.
*   Changing the name of a locally defined attribute is equivalent to deleting the original attribute and adding a new attribute with the same value. Because this may impact features that reference attributes, you are asked to confirm the name change.
*   Removing encryption does not decrypt an encrypted attribute. To safeguard encrypted values, removing the encryption on an attribute also erases its value. Click the edit icon to give the attribute a new value.

8. To delete a user attribute, click the delete icon in the attribute row and confirm the operation. Because this may impact features that reference attributes, you are asked to confirm the deletion. Click **Save** to make the deletion take effect.

When deleting a user attribute, be aware of the following:

*   Upon deletion, some attributes remain in the list as inherited attributes. This indicates that the local definition of the attribute was deleted, but another attribute with the same name exists at the server level.

- You cannot delete an inherited attribute because it belongs to the server level. To remove an inherited attribute, you must delete it or set its permission to No Access at the server level.

CHAPTER 3 REPOSITORY ADMINISTRATION

JasperReports Server provides a powerful and flexible environment for deploying and running JasperReports. The repository stores all the resources used to run and create reports, including data source definitions, JRXML files, datatypes, and helper files such as images. Administrators create the folders and resources so users can create, run, and save the reports they need. For administrators who want to customize the user interface, the repository also holds the CSS and image files that define themes.

The repository is structured as a hierarchy of folders. The JasperReports Server web interface enables you to browse the repository's resources, manage its folder structure, and secure its contents. This chapter covers the basic tasks of administering the repository, including:

- Creating folders and organizing repository objects.
- Managing references to data sources, images, fonts, and other resources upon which reports rely.
- Controlling access to resources in the repository through roles and object-level permissions.

Further information about the repository is available in the following sections:

- **1.1, "Overview of the Repository," on page 10**
- **Chapter 5, "Other Resources in the Repository," on page 73**
- **Chapter 6, "Themes," on page 97**

You can also access the repository using the web services (see the *JasperReports Server REST API Reference*) and APIs (see the *JasperReports Server Ultimate Guide*).

This chapter contains the following sections:

- **Resource Types**
- **JasperReport Structure**
- **Managing Folders and Resources**
- **Repository Permissions**

3.1 Resource Types

Some resources are created interactively, others are uploaded from files. Many of these resource types are used to upload or define the components of reports. The following tables list the resource types that users and administrators can create in the repository:

Table 3-1 Resource Types in the Repository

Resource Type	Description
JasperReport (JRXML report)	A JasperReport combines a JRXML file, a data source, and optional components such as input controls. Depending on the use case, both users and administrators create JasperReports in the server. For more information, see **3.2, "JasperReport Structure," on page 33**. Procedures for adding reports to the server are described in the *JasperReports Server User Guide*.
Content resource	Report output of any format, either from running a report in the background or from scheduling a report. A content resource is a simple file that the repository allows users to view or download.
Data source	A connection that points to a database or other data store. Data sources define where data is stored for reporting. There are several types of data sources, based on the type of connection or location of the data: JDBC, JNDI, and several others. Only administrators may create data sources. For more information, see **Chapter 4, "Data Sources," on page 47**.
Query	A database query string, for example in SQL. The JRXML doesn't necessarily include the query, in which case, you must define a query resource for use in the JasperReport.
Input Control	A complex type that specifies the values users can input for a report and how the input field appears when running the report, for example radio buttons or check boxes. Input controls depend on datatypes or lists of values to specify the format of the input.
Datatype	A basic resource that defines the format for input values, for example text, number, or date. A datatype may also specify a valid range for the input value.
List of Values	A basic resource that defines a list of arbitrary labels for input. Each label is associated with a value that can correspond to your data. For example, the Month Names List in the sample data associates the name of each month with a value 1 to 12.
File	A resource that stores a file in the repository, usually as part of a report, such as a JRXML file or image logo. The supported file formats are listed in **Table 5-3, "File Resource Types," on page 91**.

Administrators and users can also manage OLAP resources in the repository, if their license supports Jaspersoft OLAP. For more information about OLAP and Mondrian resources, see the *Jaspersoft OLAP User Guide*.

Table 3-2 Resource Types for OLAP

Resource Type	Description
Mondrian XML/A Source	A server-side XMLA source definition of a remote client-side XML/A connection.
OLAP Client Connection	Specifies how to retrieve data for 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 View	If you implement Jaspersoft OLAP, a view of multidimensional data based on an OLAP client connection and an MDX query. Like JasperReports, OLAP Views are collections of individual resources that define how to access and present the data.

3.2   JasperReport Structure

The repository resource that aggregates all information needed to run a report is called a JasperReport. A JasperReport is based on a JRXML file that conforms to the JasperReports Library that the server uses to render reports.

A JasperReport is a complex resource composed of other resources:

- The main JRXML file that defines the report
- A data source that supplies data for the report.
- A query if none is specified in the main JRXML.
    - The query may specify its own data source, which overrides the data source defined in the report.
- Input controls for parameters that users may enter before running the report. Input controls are composed of either a datatype definition or a list of values.
- Any additional file resources, such as images, fonts, and resource bundles referenced by the report template.
- If the report includes subreports, the JRXML files for the subreports.

The collection of all the resources 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 component resources.

3.2.1   Referencing Resources in the Repository

There are several ways to define and reference all the resources in a JasperReport.

In environments without JasperReports Server, reports are stored in the file system, and shared resources are usually stored on a network drive accessible to all developers and users. This solution is sometimes impractical, as you can't always add such resources to the classpath, and the use of absolute paths has its own limitations. Also, storing the resources in the file system discourages their reuse. Developers may invest time recreating resources because they don't know they exist.

By storing resources in the repository, JasperReports Server makes it easy and reliable to share resources such as images, style templates, and subreports among reports. The repository mimics a folder and file structure, so references to external files can be handled as references to external resources in the repository.

In versions of JasperReports Server before 5.5, the JasperReports used the `repo:` syntax. As of 5.5, this is no longer necessary, and regular file paths are recognized and managed within the repository.

When you upload your JRXML to the repository, your file references become valid repository references, and you can store all your resources in well-known locations in the repository. This simplifies the process of uploading your reports, because you don't have to upload the resources each time. Also, you can manage these resources either through Jaspersoft Studio, the JasperReports Server user interface, or the server's REST API. For example, when you update a logo image, all reports that reference that resource display the new logo.

3.2.2 Absolute References

Absolute references are the URIs of resources in the JRXML of a report. A path may refer to the file system where the JRXML was created, but when uploaded to the server, it refers to folders in the repository.

The path must start with one of the following:

> `/` to represent the root of the repository. For example, `/images/logo` is the path to a resource in the /images folder.

> `../` to represent the folder where the JasperReport is uploaded. For example, `../myLogo` is the path to a resource in the same folder as the JasperReport.

As with a file system path, the repository path is composed of the resource ID of every parent folder, ending with the ID of the resource. Jaspersoft Studio supports absolute references by allowing you to drag resources from the repository tree view into the design area.

When uploading the JRXML with absolute resource references as part of a JasperReport in the server, you need to ensure only that the resource with the given path exists in the repository before running the report. When the report runs, the server locates the resource in the repository and uses it to render the report.

Because file resources such as images, fonts, and JARs are the only resources for which you can create references directly in JRXML, they are the only resources for which you can create absolute references.

One disadvantage of absolute references is that JasperReports Server does not maintain the dependency between the JRXML and the absolute reference. When uploading the JRXML, there is no warning if the resource does not exist, and the server allows you to delete the resource from the repository even if it is still being referenced. If the resource is not available, running the report fails with an error.

3.2.3 Local Resources and External References

JasperReports Server provides more flexibility and power when you use indirect references instead of absolute references. Indirect references are placeholder names that must be manually linked to the resource when uploading the JasperReport. The syntax for an indirect reference contains only a resource placeholder name, for example:

> `logoImage`

When you upload a JRXML with an indirect reference, the server prompts you to provide the resource. You have two options:

- Create a new resource, in this case by uploading an image that becomes part of the JasperReport. This is called a local resource. You can't access this resource from elsewhere in the repository, it exists only within the JasperReport.
- Select a resource from the repository. This is called an external reference because it is external to the JasperReport. Any number of reports can link to the same external resource, and the resource can be managed independently of them.

While indirect references require slightly more work than absolute references in the JRXML, the server manages the dependency. Local resources exist as part of a JasperReport, and external references cannot be deleted until they are no longer referenced.

In cases where you don't want to reference existing resources, local resources allow reports to be highly customized and self-contained. A local resource defined inside the JasperReport has all the same properties as a repository resource, but it's not accessible in the repository. Users must edit the JasperReport to access any resources it defines locally.

Indirect references are used implicitly in several other cases when you define a JasperReport:

*   The main JRXML itself is either a local resource created by uploading a file or an external reference to an existing JRXML file resource in the repository.
*   Every report must have a data source, and JasperReports Server gives you the option to either create a new local resource or use an external reference to an existing data source.
*   Every report must also have a query that matches its data source. You may choose to create a local query resource or use an external reference to an existing query.
*   Parameters in a report are implicitly handled as an indirect reference to an input control. For every parameter named in your main JRXML, you must define an input control either as a local resource or external reference.

Every level of indirect referencing is independent of the other. For example, when creating a JasperReport, you may choose to create an input control as a local resource, but that input control may have an external reference to its datatype. The server still manages the dependency between the local input control and the datatype resource in the repository.

Local resources and external references are used by several resources, for example when creating input controls, query resources, Domains, and OLAP resources.

3.2.4   References in Subreports

A subreport is a subordinate JRXML file within a JasperReport. As with all other resources referenced by the main JRXML, the subreport JRXML file may be specified by an absolute reference, a local resource or an external reference.

As a JRXML file, a subreport can reference other resources of its own. However, the subreport is run as part of the main JRXML, and any references in the subreport are interpreted relative to the JasperReport resource (represented by the main JRXML) and the context in which the JasperReport is being run.

3.2.5   Data Snapshots

Since JasperReports Server 4.7, report resources can also store a snapshot of the report data. A snapshot is a copy of the data that the query returns when the data is refreshed. This data snapshot is an internal structure not visible or accessible from the repository. However, when data snapshots are enabled, a data snapshot is stored in the repository with each report. When users open a report, the report viewer retrieves and displays the data from the snapshot. Users then have the option of refreshing the data in the report viewer, and if they have permissions, saving the data snapshot back into the report resource.

For more information about interacting with data snapshots, see the *JasperReports Server User Guide*. To enable snapshots, see **8.4, "Enabling Data Snapshots," on page 131**.

3.3　Managing Folders and Resources

Administrators and users with the proper permissions can create, modify, move, and delete folders and resources in the repository. The specific roles and permissions of the user determine the actions available. For the definition of the permissions on folders and resources, see **3.4, "Repository Permissions," on page 41**.

One responsibility of an administrator is to set up an environment for users to create and save dashboards, Ad Hoc views, and reports. That usually means creating a folder structure where users have write permission. Users with write permission can also create their own sub-folders to store their reports and dashboards.

Only administrators can create data sources in the repository. So, although users with write permission can upload JRXML files and define resources, they can't run their reports without data sources. For this reason, we recommend that administrators create data sources and other shared resources in the repository. For example, uploading one shared logo file and keeping it up to date is better than many users uploading and maintaining their own separate logo files.
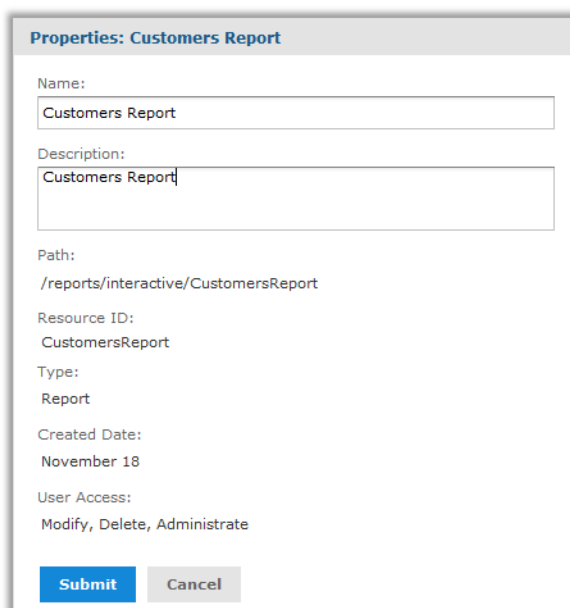
3.3.1　Resource IDs

Each resource, including each folder, has an ID, a name, and an optional description:
- The ID is used internally to reference a resource. The ID must be unique within its folder, but may exist in multiple folders.
- The resource name is displayed in the repository.
- The description, if defined, appears in the repository and in tooltips.

As in a file system, the IDs of nested folders define the path to a resource. For example, the path to a report might be: /reports/samples/Freight.

To view the name and resource ID of a resource, Right click the folder's name or the resource in the repository or search results and select **Properties...** from the context menu.

Figure 3-1　Resource Properties Dialog for a Writable Resource
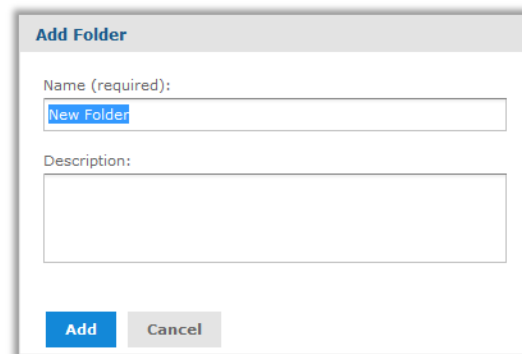
If you have write or administer permission as shown in the figure, you can also edit the name and description of the resource. For some operations such as export, you need the path, also called repository URI, which you can copy from this dialog.

3.3.2   Creating Folders

Any user with write permission on a folder can create new sub-folders.

To create a folder:

1. Log on as a user who has write permission to the parent folder.
2. Select **View > Repository** and locate the parent folder in the Folders panel.
3. Right-click the parent folder and select **Add Folder** from the context menu. The Add Folder dialog appears.

Figure 3-2 **Add Folder Dialog**

4. Enter the folder name and, optionally, a description, then click **Add**.

    The folder is created in the repository. The name appears in the hierarchy of folders. The description is visible only when viewing the properties of the folder, as shown in **Figure 3-1**.

    New folders and their future contents inherit the permissions of their parent folders. Users with Administrator permissions can change permissions folders, as described in **3.4.6, "Setting Permissions," on page 44**.

3.3.3   Adding Resources

Each resource has different requirements, for example some are created from uploaded files, others are created by defining values in a wizard. The procedures for adding each type are available in the documentation below:

*   JasperReports are covered in the *JasperReports Server User Guide*.
*   Mondrian and OLAP resources are covered in the *Jaspersoft OLAP User Guide*.
*   Data sources are explained in the chapter **Chapter 4, "Data Sources," on page 47**.
*   Queries, data types, lists of values, input controls, and file resources are explained in the chapter **Chapter 5, "Other Resources in the Repository," on page 73**

Most resources are created through the Add Resource menu item on the context menu for folders in the repository. In the following figure, you can see the full menu and submenu with all the resources administrators can create:
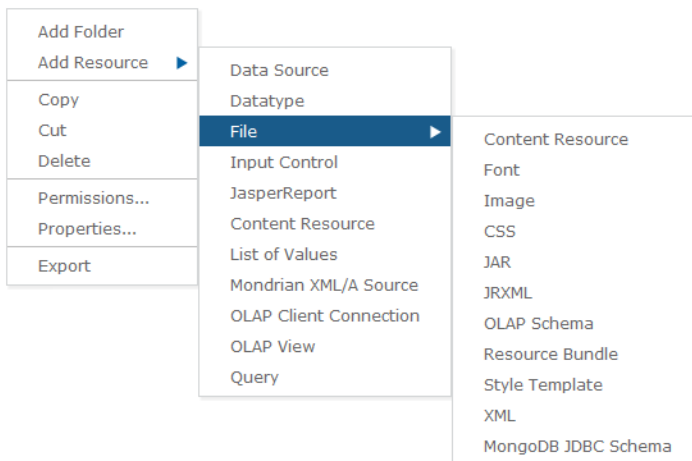
Figure 3-3 **Add Resource Context Menu Expanded**

For every resource you create, you must specify a name and resource ID for referencing the resource in the repository. Each wizard also has one or more pages for specifying the values and controls specific to the resource.
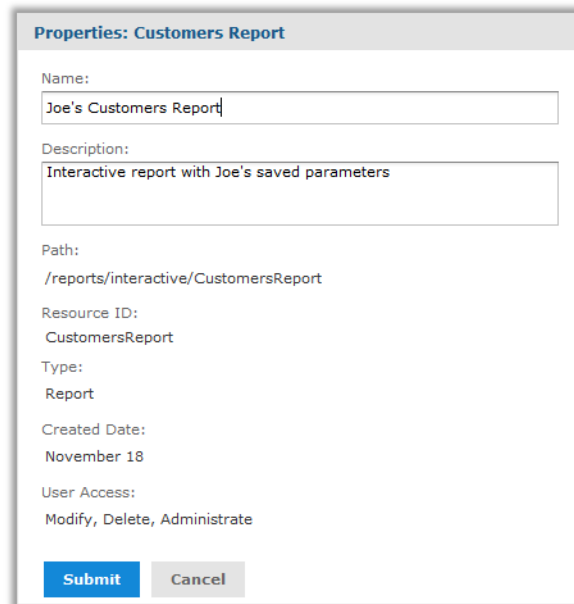
> New resources inherit the permissions of the folder in which they are created. Administrators can change the permissions on the new resource, as described in section **3.4.6, "Setting Permissions," on page 44**.

3.3.4   Renaming Folders and Resources

Any user with write permission on a folder or resource can change its name and description.

To rename a folder or resource:

1. Log on as a user who has write permission for the folder or resource.
2. In the repository, browse or search for the resource. For renaming folders, select **View > Repository** and locate the folder.
3. Right-click the object and select **Properties...** from the context menu. The Properties dialog appears.

Figure 3-4 **Properties Dialog for a Report Resource**

You can change the folder or resource's name and description, but not the ID; the ID is permanent once the resource is created.

4. Click **Submit** to save your changes.

3.3.5 Copying and Moving

The repository interface lets any user with the proper permission copy or move both resources and folders. Copying requires read permission on the source, moving requires delete permission on the source, and both require write permission on the destination folder.

You can drag-and-drop the objects, or you can copy-paste or cut-and-paste them from their context menus. Folders must be moved one at a time, but multiple resources from the same folder can be copied or moved together.

Copying and moving actions are not possible on the search interface, only on the repository interface showing the list of folders. Currently, it's not possible to create a copy of a resource in the same folder.

> The moved objects inherit their permissions from the destination folder; they do *not* keep the permissions they had before the move. If you want the objects to have other permissions, you can set new permissions after the move (see **3.4, "Repository Permissions," on page 41**).

To copy or moving folders and resources:

1. Log on as a user who has the required permissions for the folder or resource.
2. Click **View > Repository**, and expand the folders to display the object to be copied or moved.
3. Right-click the resource and select **Copy** or **Cut** (delete permission is required to cut a resource).
    You can select multiple resources with Control-click or Shift-click, but only a single folder.

4. Right-click the destination folder and select **Paste** in the context menu (write permission is required on the destination folder).

Alternatively, you can drag to move the selected resource or folder to the destination folder. To copy, press and hold the Ctrl key then click and drag. When dragging resources, the destination folder is highlighted in blue if you have write permission to it.

3.3.6 Editing Resources

The procedure for editing a resource depends on the resource type. All of the dialogs are available by right-clicking on the resource in the repository and selecting the proper action from the context menu. In nearly all cases, the dialog for editing is the same one that was used to create the resource.

Table 3-3 Editing Resources

Resource Type	How to Edit
JasperReport	To change the layout of an interactive report, users may **Run** the report, then change the column filters or sorting. Users may save the report, either by overwriting the original or as a new copy, depending on their permissions. To change the definition of a report, right-click the report resource and select **Edit**. Then you can change the data source, input controls, or file resources referenced in the JasperReport. For more information, see **3.2, "JasperReport Structure," on page 33**.
Content Resource	A content resource is report output that is stored as a file stored in the repository. These files cannot be edited, only downloaded or deleted.
Data Source	Administrators only: right-click the data source, then select **Edit** from the context menu. For more information, see **Chapter 4, "Data Sources," on page 47**.
Query	Right-click the resource, then select **Edit** from the context menu on these resources. You edit these resources in the same dialog you used to create them. For more information, see **Chapter 5, "Other Resources in the Repository," on page 73**.
Input Control	
Datatype	
List of Values	
File	

When editing a resource, you have several limitations:
- You can't change a resource's ID. If you need to change an ID, you have to create a new resource and delete the old one.
- You can't change the location of a resource. To change the location of the resource, see **3.3.5, "Copying and Moving," on page 39**.
- For file resources, you can't see the name of the file that was uploaded, nor in most cases download and view the contents of the file. Your only option is to upload a new file to replace the old one.

3.3.7   Deleting Folders and Resources

Users with delete permission on folders or resources can delete those objects from the repository. To delete a folder, you also need delete permissions on all the resources and folders in that folder. Folders must be deleted one at a time, but multiple resources can be deleted together.

> You can't undo a delete.

The repository won't allow you to delete resources currently referenced by other resources. For example, an input type used by a report cannot be deleted as long as the report still references them.

To find the resources that reference the one you want to delete, you need to look at each report that you suspect of referencing it. When you edit the definition of a JasperReport, you can see the resources it references. Then you can either remove the reference from the resource or delete the entire resource containing the reference.

To delete a folder or resource:

1. Log on as a user who has delete permission for the folder or resource.
2. In the repository, browse or search for the object to be deleted.
3. Right-click the object and click **Delete** in the context menu.

   In the repository view, you can select multiple resources and click **Delete** in the tool bar or in the context menu. In the list of folders, you can delete only one folder at a time, but all contents of that folder will be deleted. In the search results, you can select multiple resources and right-click to select **Delete** in the context menu.

3.4   Repository Permissions

Permissions on folders and resources determine what users see in the repository and what actions they're allowed to perform. In the following table, the actions granted by each permission include all of the actions granted by permissions above it, except for the No Access permission. The actions granted by each permission strictly exclude all of the actions granted by permissions below it.

Permission	Actions Granted on Repository Folders and Resources
No Access	Users can never see or access the folder or resource, either directly in the repository or indirectly when running a report or OLAP view. If one of these reference a resource that is set to no-access, the user will see an error when trying to run it.
Execute Only	Users can never see the folder or resource in the repository, but they can run reports or OLAP views that access them. Typically, data sources are execute-only so that users may run reports but not see database connection details.

Permission	Actions Granted on Repository Folders and Resources
Read Only	• See the folder or resource in any server • See the properties of a folder or a resource • Copy a folder and all of its readable contents • Copy resources individually or in bulk • View (run) a report or OLAP view • Run a report in the background • Schedule a report to run later
Read + Delete	• Cut (move) a folder and all of its contents • Delete a folder and all of its contents • Cut (move) resources individually or in bulk • Delete resources individually or in bulk
Read + Write	• Copy resources to a folder with this permission • Edit and modify resources
Read + Write + Delete	• Add a subfolder • Add (create) a resource in a folder • Paste into a folder (copy or cut) • Save the output of a scheduled report in a folder • Rename a folder or resource and change its description string • Add a JasperReport resource to the repository (upload a JRXML) • Edit the definition of a JasperReport resource in the repository (replace the JRXML)
Administer	• Set the permissions (by role and by user) on a folder or resource. This effectively delegates certain repository administration tasks.
Administer and ROLE_ ADMINISTRATOR	• Add (create) a Domain or a data source

Permissions apply when browsing or searching the repository and when using any dialog that accesses the repository, like browsing folders to save a report. Note that:

• Copying does *not* preserve the permissions on an object. Users may copy a read-only object, paste it into a read-write folder, then edit the object. For more details, see **3.3.5, "Copying and Moving," on page 39**.
• Copying and cutting (moving) actions can be completed only by a user with Read + Write + Delete access to the folder in which the object is pasted. For more details, see **3.3.5, "Copying and Moving," on page 39**.
• Cutting, deleting, and setting permissions on folders is allowed only if the user has the same permission on all folder contents.
• Cutting and deleting resources in bulk is allowed only if the user has at least Read + Delete permission on all selected resources.
• Deleting a resource is allowed only if no other resources rely on it. For more details, see **3.3.7, "Deleting Folders and Resources," on page 41**

3.4.1   Inheriting Permissions

According to the permission architecture, there may be a permission setting for every user and role on every folder and resource in the repository. To simplify the definition of permissions, JasperReports Server supports the inheritance of permissions from the parent folder to a folder or resource. If no permission is explicitly defined for a user or role on a given folder or resource, the user or role has the same access permission that is defined on the parent folder. When a permission is defined explicitly, that permission is enforced, regardless of those on the parent folder.

Using permissions, you can manage large hierarchies of content and keep them secure. When you set a permission explicitly, that permission for a given user or role is inherited recursively by all of the folder's contents and subfolders, unless they have an explicit definition of their own.

3.4.2   Cumulative Permissions

Because permissions can be assigned to both users and roles, a user belonging to one or more roles may have multiple permissions defined or inherited on any given folder or resource. In fact, every permission must be defined on the root, even if it has the default value of No Access, and therefore every role- and user-based permission on every folder and resource has a setting through inheritance. So, for every folder or resource, every user has their own user-based permission and the permission assigned to the ROLE_USER.

How does JasperReports Server determine the effective permission from the many that apply? Permissions in the server are strictly cumulative, meaning that the least restrictive among the set of all permission applies. Even if a more restrictive permission, such as No Access, is set explicitly, the less restrictive permission such as Read-Only applies, regardless of whether it is inherited or set explicitly. This is why all default permissions at the root are set to No Access initially.

3.4.3   Administrator Permissions

The JasperReports Server authorization architecture distinguishes between administrators and all other users. Administrators are defined as users with ROLE_ADMINISTRATOR. By design, administrators with the ROLE_ ADMINISTRATOR always have irrevocable Administer access to the entire repository. The administrator cannot modify the permissions for ROLE_ADMINISTRATOR, to prevent being locked out or unable to administer some resource. Therefore, the administrator can set permissions for all other users, on any folder or resource.

3.4.4   Execute-Only Permission

As in file systems, execute-only permission in JasperReports Server allows running reports and OLAP views to access a resource, but keeps the resource from appearing in the repository.

Execute-only permission applies to folders as well, keeping them from appearing in the folder tree when users browse the repository, yet still allowing the resources they contain to inherit the execute-only permission. This is useful for hiding folders and resources such as data sources that only administrators and data analyst roles need to access in the repository. However, if your execute-only folder contains read-only resources, those resource are hidden when browsing folders but can be found with a repository search.

As with all other permissions, execute-only permission is either role-based or user-based, so only certain users can access a resource from a running report.

⚠ If you have data or sensitive content in a resource, always set No Access permission for users or roles that must not be able to access it.

Hiding a resource with execute-only permission does not protect against access, because malicious users who find the resource ID may be able to create a report, dashboard, or OLAP view that extracts the sensitive content.

3.4.5 Default User Permissions

For all non-administrator users, the default permission at the root is No Access and any permissions must be explicitly defined. In practice, the default installation of the repository contains sample data with a mix of no access, execute only, read only, and read-write permissions that allow the sample users to access folders and resources. The sample permissions demonstrate a common approach to permissions, allowing users to see the resources they can access and hiding the ones they can't, while administrators have full access.

We recommend you familiarize yourself with permissions by viewing and setting permissions in the sample data, as described in the following section.

3.4.6 Setting Permissions

Administrators can assign permissions to access any folder or resource throughout the repository. Users with the Administer permission on a folder can assign permissions to that folder and any contents that inherit the permission. Users granted Administer permission to a resource can set the permissions only on that specific resource.

To set permissions on a folder or resource in the repository:

1. Log in as a user with administrator privileges.
2. Browse or search the repository for the folder or resource.
3. Right-click the object and select **Permissions...** from the context menu:

Add Folder	
Add Resource	▶
Copy	
Cut	
Delete	
Permissions...	
Properties...	
Export	
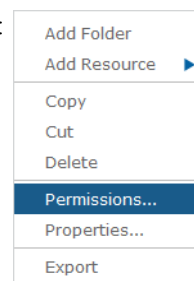
The Permissions dialog opens showing the permissions in effect for the selected object. By default, it first shows the permissions given to roles. Permissions that are inherited from the object's parent are indicated by asterisks (*).
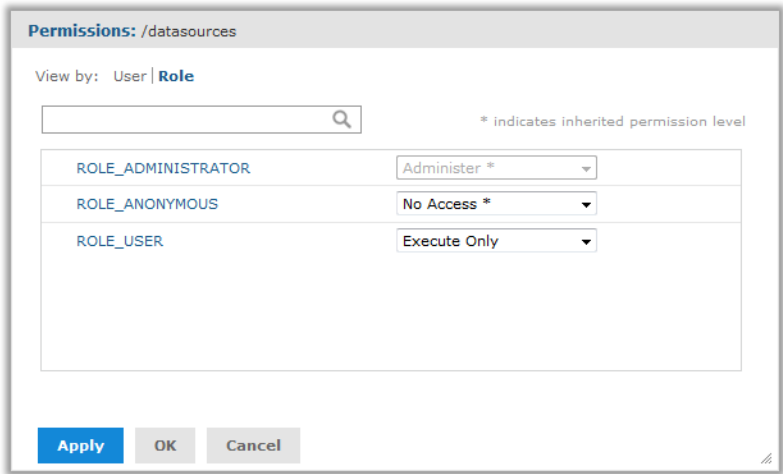
Figure 3-5 Permissions Dialog Showing Permissions by Role

In the previous figure, you can see the default role-based permissions on the sample Data Source folder. Regular users have execute only permission so they do not see this folder, but the reports they run can access its contents. Administrators are prevented from changing the permission for their administrator role or user name, to prevent them from removing their ability to set permissions.

4. In the dialog, click **User** to view the permissions assigned to specific users. Click **Role** when viewing user permissions to toggle back.

5. For each user or role, you can select a new permission from the drop-down.

In the next figure, you can see the default user permissions on this folder. In the default installation, all permissions are defined by role; so, all user permissions are No Access inherited from the root. The figure shows a read-only permission being granted to the sample end user. This enables `joeuser` to see but not modify the Data Sources folder and its contents. For all other end users, the folder is still execute-only due to the settings in **Figure 3-5**.
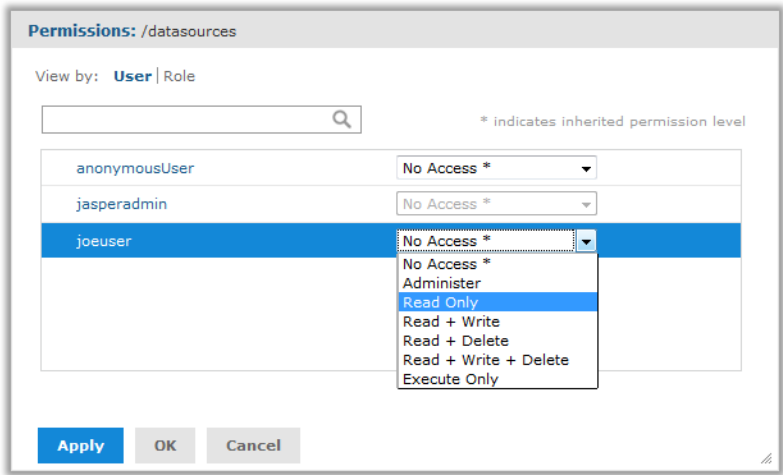
Figure 3-6 Permissions Dialog Showing Permissions by User

6. Click **Apply** to apply your changes. If you toggle between user and role permissions, first apply any changes you made.

7. Click OK to save your changes and close the permissions dialog when you're finished.

   You can open several permissions dialogs for different resources or folders at the same time while navigating the repository. This helps when trying to set permissions uniformly across several folders.

There are two special cases when setting permissions:

- If a resource inherits a permission, for example Read-Only, you cannot set the permission to the same value, at least not directly. You need to temporarily change the permission level on the parent folder, then set the explicit permission, then set the parent folder's permission back to the original value. When a resource and its parent folder have been set to the same permission in this way, the permission dialog still shows the asterisk as if the permission were inherited. But if the parent is later given a different permission, for example Read-Write, the resource retains its explicit Read-Only permission instead of inheriting Read-Write.

- To reset the permission level so that it once more inherits from its parent folder, select a different permissions level and click **Apply**, then select the permission with the asterisk and click **Apply** again.

3.4.7   Testing User Permissions

Once you have configured users, roles and permissions, we recommend that you test the permissions granted to a few representative users. We also recommend testing whenever you add new users, roles, and resources or make any major modifications to your access control configuration.

To test user permissions:

1. Log in as an administrator.
2. Select **Manage > Users**.
3. Browse or search for the user whose permissions you are testing.
4. In the Users panel, select the user.
5. In the Properties panel, click **Login as User**. The selected user's Home page appears. The login information in the upper-right corner shows that you are logged in as that user.
6. Verify that the expected folders and resources are available in the repository. Make a note of any objects that should be there but are not, and any that should be hidden but are displayed.
7. When you have verified the user's permissions, click **Log Out** to exit that user's account.
8. To change the user's permissions, edit the permissions in the repository and modify the user or role definitions.
9. Continue testing until the user's permissions are satisfactory.
10. Repeat these steps with several representative users to ensure that your access control is properly configured. An untested access control configuration can't secure your data adequately.

CHAPTER 4   DATA SOURCES

A data source is a resource in the repository that specifies how and where to obtain the data displayed by reports and OLAP views. Administrators must define data sources before uploading reports that rely on them. Typically, a data source specifies the URI of the database server and the details you need to access it, such as a user name and password.

JasperReports Server provides data source types for relational databases, most flavors of big data, and for specialized data such as Amazon Web Services and JavaBean data. Virtual data sources allow you to combine several data sources into one.

JasperReports Server can access any relational database that supports the SQL query language through the JDBC (Java DataBase Connectivity) API. In this case, you can configure two types of data sources in the repository:

- JDBC data source – Establishes a direct connection to the database server using its JDBC driver. After installation, JasperReports Server includes JDBC drivers to access MySQL and PostgreSQL. If you want to install drivers for other databases, or if you want to use alternate drivers, the administrator can upload and manage JDBC drivers through the UI. With JDBC data sources, JasperReports Server configures and manages the connections to the database. By default, the maximum number of simultaneous connections for each data source is 20.
- JNDI data source – Relies on the JNDI (Java Naming and Directory Interface) service of your application server to access a database connection. You must first configure your application server to install its JDBC drivers and configure its database connections. With JNDI data sources, the configuration of the application server determines the number of shared connections. Note that the application server connects to the database using JDBC, meaning that JNDI data sources are available for all databases that support JDBC.

Big data stores that have custom data sources:

- Cassandra – This data source introduced in JasperReports Server 5.6 is different from the community-contributed data source for Cassandra. This data source supports the Cassandra Query Language CQL 3. JasperReports Server also provides a JDBC driver for Cassandra data sources.
- MongoDB – This custom data source for MongoDB supports Jaspersoft's own MongoDB Query Language. A MongoDB JDBC driver is also available.

JasperReports Server also supports some specialized data sources:

- Amazon Web Services (AWS) data source – Accesses data stored in your AWS data store using JasperReports Server, either on-premises or in the cloud.
- Microsoft Azure SQL data source - Allows you to access data stored in your Azure SQL Server database.
- File data source – Allows you to generate reports based on data in XML and JSON format.
- Bean data source – Allows you to access data encapsulated in JavaBeans.

In the case of analysis data, JasperReports Server supports OLAP data sources (such as Mondrian and XML/A connections). For information about analysis data sources, refer to the *Jaspersoft OLAP Ultimate Guide*.

> You can extend JasperReports Server to support any custom data source. Custom data sources consist of Java implementation classes, a message catalog, and a Spring bean definition. For more information about custom data sources, see the *JasperReports Server Ultimate Guide*.

This chapter contains the following sections:
- **Attributes in Data Source Definitions**
- **JDBC Data Sources**
- **Managing JDBC Drivers**
- **JNDI Data Sources**
- **AWS Data Sources**
- **Azure SQL Data Sources**
- **Cassandra Data Sources**
- **MongoDB Data Sources**
- **File Data Sources**
- **Bean Data Sources**

4.1 Attributes in Data Source Definitions

You have two options for defining the parameters of a data source, such as the host and port number:
- Define the parameters statically, so that they are the same for every user.
- Have the server derive these parameters at run time based on attributes you provide.

Attributes can be used to derive all data source parameters that are not selected from drop down lists. For instructions on how to create attributes, see **2.3, "Managing Attributes," on page 24**.

When referring to a an attribute in a data source definition, you can specify the attribute categorically or hierarchically:
- Categorical reference – If you specify a category for the attribute value, the server attempts to find that particular value of the attribute. If the attribute is not defined where specified, the reports using this data source will fail with an error. You can specify these attribute categories:
  - User – In the attributes defined on the logged-in user.
  - Server – In the attributes defined at the server-level.
- Hierarchical reference – If you don't specify a category for the attribute, the server searches attributes hierarchically and uses the value of the first attribute it finds with the given name. This search starts with the logged in user, then proceeds to the server level. If the specified attribute is not found in either of these, the reports using this data source will fail with an error.

The following figure is an example of attributes used to define data source parameters.

Figure 4-1 Using Attributes for Data Source Parameters

In the example above, the following data source parameters are specified by attributes:

- `{attribute('host','User')}` - Host will be derived categorically from the logged-in user's attributes, because the "User" category is specified.
- `{attribute('port','Server')}` - Port will be derived from the server-level attributes.
- `{attribute('db','Server')}` - Database name will be derived from the server-level attributes.
- The URL is generated automatically from the host, port, and database fields.
- `{attribute('userName')}` - The user name will be derived hierarchically, because no category is specified. JasperReports Server will look for a host first in the logged in user's attributes, then in the server-level attributes.
- `{attribute('password')}` - The password field can also reference an attribute, here a hierarchical attribute.

For information about attributes on users and the server, see **2.3, "Managing Attributes," on page 24**.

4.2   JDBC Data Sources

JDBC data sources are direct connections to your database managed by JasperReports Server. To create one, you must provide the URL and credentials to access your database, along with any database-specific configuration parameters.

JasperReports Server includes JDBC drivers for some databases. If your database is not included, or if you want to use different JDBC drivers, the administrator must upload the appropriate JDBC driver before creating a data source. For more information on JDBC drivers, see **4.3, "Managing JDBC Drivers," on page 52**.
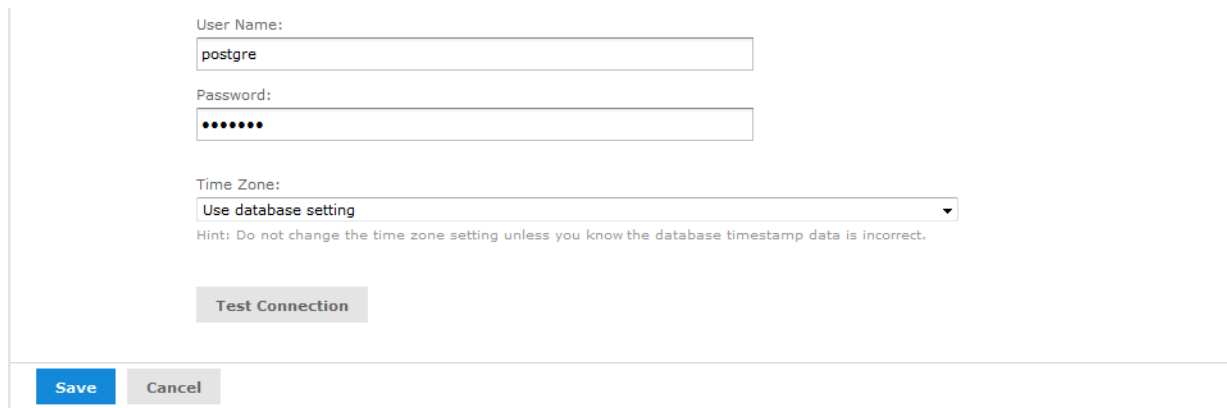
To create a JDBC data source:

1. Log on as jasperadmin.

2. Select **View > Repository**, right-click a folder's name, and select **Add Resource > Data Source** from the context menu. If you installed the sample data, the suggested folder is Data Sources. The New Data Source page appears.

3. In the Type field, select **JDBC Data Source**. The page refreshes to show the fields required for a JDBC data source.

Figure 4-2 **Setting the JDBC Data Source Type**

4. Select the JDBC driver for your database. If your driver is listed as NOT INSTALLED, you must first upload the driver as described in **4.3, "Managing JDBC Drivers," on page 52**.

5. Enter the hostname, port, and database name for your database. The default hostname is localhost, and the default port is the typical port for the specified database vendor. The three fields are combined automatically to create the JDBC URL where the server will access the database. When specifying values for your JDBC data source:

   • The JDBC URLs for some databases allow optional parameters described in **A.10.5, "JDBC Database URLs," on page 167**.

   • You have the option to use attributes in the values of data source parameters. See **4.1, "Attributes in Data Source Definitions," on page 48**.

6. Fill in the database user name and password. These are the credentials the server will use to access the database.

Figure 4-3 **Entering the User Name and Password**

> The database user needs the privileges to run SELECT queries on the tables used in your reports. The server blocks any DROP, INSERT, UPDATE, and DELETE SQL commands through its SQL injection protection. In some cases, additional permissions may be required to execute stored procedures, depending on your configuration and needs. For more information, see **A.10.4, "Database Permissions," on page 167**.

7. If the date-time values stored in your database do not indicate a time zone, set the Time Zone field. When in doubt, leave the default Time Zone value (**Use database setting**).

   When date-time values are stored in a format other than local time zone offset relative to Greenwich Mean time (GMT), you must specify a time zone so that the server can properly convert date-time values read from the target database. Set the Time Zone field to the correct time zone for the data in the database. The list of time zones is configurable, as described in **B.5.2, "Specifying Additional Time Zones," on page 189**.

8. Click **Test Connection** to validate the data source. If the validation fails, ensure that the values you entered are correct and that the database is running. To diagnose JDBC connection issues, you can turn on logging as described in the troubleshooting section **A.10.1, "Logging JDBC Operations," on page 166**.

9. When the test is successful, click **Save**. The Save dialog appears.
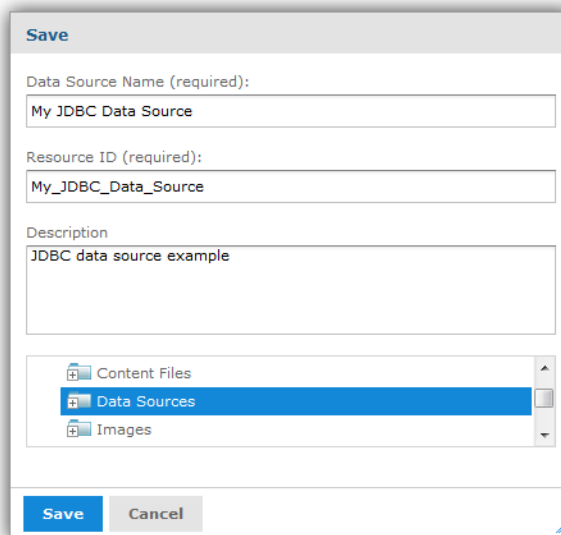
Figure 4-4 Saving the JDBC Data Source

10. Enter a name for the data source and an optional description. The Resource ID is generated from the name you enter. If you haven't already specified a location, expand the folder tree and select the location for your data source.

11. Click **Save** in the dialog. The data source appears in the repository.

4.3   Managing JDBC Drivers

To access a database from JasperReports Server using JDBC you need an appropriate driver that's accessible in the server's classpath. As of JasperReports Server 6.4, the following drivers are installed by default:

- Cassandra (com.simba.cassandra.jdbc4.Driver)
- Hive (tibcosoftware.jdbc.hive.HiveDriver)
- MySQL (org.mariadb.jdbc.Driver)
- Oracle (tibcosoftware.jdbc.oracle.OracleDriver)
- PostgreSQL (org.postgresql.Driver)
- Redshift (tibcosoftware.jdbc.redshift.RedshiftDriver)
- Salesforce (tibcosoftware.jdbc.sforce.SForceDriver)

Drivers for other databases can be downloaded from links on the Jaspersoft community website:

http://community.jaspersoft.com/wiki/downloading-and-installing-database-drivers

The administrator (jasperadmin) can add JDBC drivers for other databases in the following ways:

- During installation. For more information, see the *JasperReports Server Community Project Installation Guide*.
- At any time through the UI. As described in the procedure below, the administrator can add, replace, or remove JDBC drivers through the user interface, without needing to restart the server.

JBoss lacks the flexibility of uploading drivers on the fly. On JBoss, drivers that are not installed don't appear in the list below, and you must configure and restart JBoss to add a driver. For more information, see **A.10.3, "JDBC Drivers on JBoss," on page 166**.

To add a JDBC driver:

1. Log on as administrator (jasperadmin).

2. Select **View > Repository**, right-click a folder's name, and select **Add Resource > Data Source** from the context menu.

3. In the Type field, select **JDBC Data Source**. The page refreshes to show the fields necessary for a JDBC data source.

4. The drop-down selector for the JDBC Driver field shows the available JDBC drivers and those that are not installed.
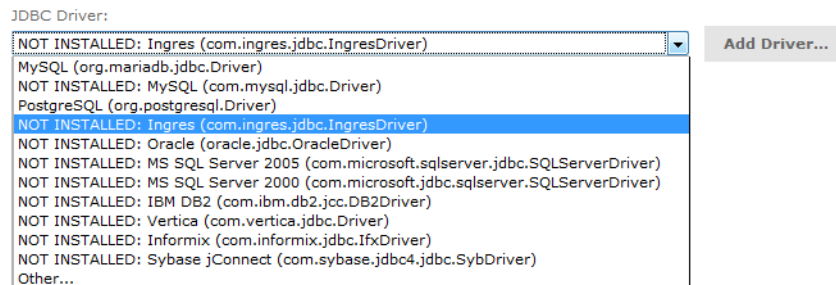
Figure 4-5 **Viewing the List of Available JDBC Drivers**

5. Select the driver that has not been installed, then click **Add Driver**. The Select Driver dialog appears.

6. If you have not yet obtained the driver, click the link to Jaspersoft's community website for Downloading and Installing Database Drivers. That page has links to the most commonly used JDBC drivers. After you download a driver to your file system, you can return to the Select Driver dialog.
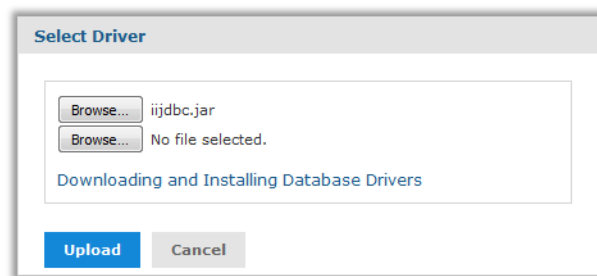
Figure 4-6 **Adding a JDBC Driver**

7. In the Select Driver dialog, click **Browse** to locate the appropriate driver JAR file. If your driver has more than one JAR file, click the **Browse** button that appears after selecting the first file.

8. Click **Upload** to install the driver and make it available immediately.

You can replace any driver that you upload with newer versions of the same driver..

To update a JDBC driver:

1. Log on as administrator (`jasperadmin`).
2. Select **View > Repository**, right-click a folder's name, and select **Add Resource > Data Source** from the context menu.
3. In the **Type** field, select **JDBC Data Source**. The page refreshes to show the fields necessary for a JDBC data source.
4. The drop-down selector for the JDBC Driver field shows the available JDBC drivers and those that are not installed.
5. To update a driver that has already been installed, select it from the list, then click **Select Driver**. The Select Driver dialog appears and notifies you that selecting a driver will overwrite the existing one.
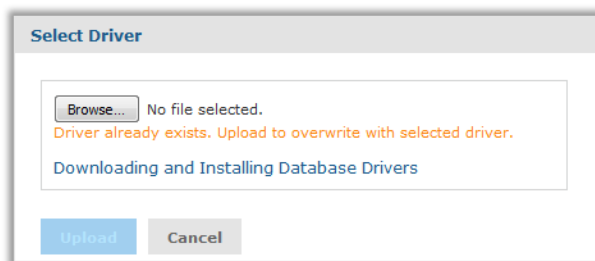
Figure 4-7 Updating a JDBC Driver

6. In the Select Driver dialog, click **Browse** to locate the JAR file for the new driver.
7. Click **Upload** to replace the existing driver and make it available immediately.
8. You can now use this driver to create a data source.

To remove an uploaded JDBC driver:

1. Log on as administrator (`jasperadmin`).
2. Select **Manage > Server Settings** and choose **Restore Defaults** from the left-hand panel.
3. Locate the driver you uploaded in the list of properties. The drivers with the value [SYSTEM] are the default drivers configured at installation time. Do not remove the [SYSTEM] drivers.
4. Click 🗑 beside the driver you want to remove, then confirm your choice.

⊟ Settings	Restore Defaults		
Log Settings	Name	Value	
OLAP Settings	jdbc:com.ingres.jdbc.IngresDriver	com_ingres_jdbc_IngresDriver	🗑
Cloud Settings			
Server Attributes			
● Restore Defaults			
Import			
Export			
	Save Cancel		

Figure 4-8 Removing an Uploaded JDBC Driver

5. Click **Save** to save your changes.

> If the JDBC driver you remove is one that updated a default driver, the default driver will reappear as an installed driver the next time you use the New Data Source wizard.

4.4 JNDI Data Sources

The JNDI data source accesses a database connection previously defined in the application server and published as a resource or service through JNDI (Java Naming and Directory Interface). Instead of specifying a driver and database as you do with JDBC data sources, you need to specify only the JNDI service name in your application server.

> Application servers use JDBC connections themselves to expose a database through JNDI. You must specify the JNDI service name of a JDBC connection. Your application server must also have the appropriate JDBC drivers and be configured to use them.
>
> Ensure the database user in your JNDI definition has the privileges to run SELECT queries on the tables used in your reports. In some cases, additional permissions may be required to execute stored procedures, depending on your configuration and needs. For more information, see **A.10.4, "Database Permissions," on page 167**.

For information about setting up a JNDI connection in your application server, see the following sections:
- **A.10.8, "JNDI Services on Apache Tomcat," on page 169**
- **A.10.9, "JNDI Services on JBoss," on page 170**
- **A.10.10, "JNDI Services on WebLogic," on page 170**

To create a JNDI data source:

1. Log on as an administrator.
2. Click **View > Repository**, expand the folder tree, and right-click a folder to select **Add Resource > Data Source** from the context menu. If you installed the sample data, the suggested folder is Data Sources. The New Data Source page appears.

3. In the Type field, select **JNDI Data Source**. The information on the page changes to reflect what's needed to define a JNDI data source.

4. Fill in the service name. This is the name the application server exposes through JNDI.

   You have the option to use attributes in the service name. See **4.1, "Attributes in Data Source Definitions," on page 48**.

New Data Source

Set Data Source Type and Properties

First, select the type of data source you wish to add, then enter the required property values.

Type: JNDI Data Source

Service Name (required):

jndi/foodmart

Hint: jdbc/jasperserver

Time Zone:

Use database setting

Hint: Do not change the time zone setting unless you know the database timestamp data is incorrect.

Test Connection

Save   Cancel

Figure 4-9 JNDI Data Source Page

5. If the date-time values stored in your database do not indicate a time zone, set the Time Zone field. When in doubt, leave the default Time Zone value (**Use database setting**).

   When date-time values are stored in a format other than local time zone offset relative to Greenwich Mean time (GMT), you must specify a time zone so that the server can properly convert date-time values read from the target database. Set the Time Zone field to the correct time zone for the data in the database. The list of time zones is configurable, as described in **B.5.2, "Specifying Additional Time Zones," on page 189**.

6. Click **Test Connection** to validate the data source. If the validation fails, ensure that the values you entered are correct, that the database is exposed through JNDI, and that the database is running. Also see the troubleshooting section **A.10.8, "JNDI Services on Apache Tomcat," on page 169**.

7. When the test is successful, click **Save**. The Save dialog appears.

8. Enter a name for the data source and an optional description. The Resource ID is generated from the name you enter. If you haven't already specified a location, expand the folder tree and select the location for your data source.

9. Click **Save** in the dialog. The data source appears in the repository.

4.5 AWS Data Sources

Amazon Web Services (AWS) provide computation and data storage on demand in the cloud. We partner with Amazon to deliver business intelligence solutions based on AWS.

JasperReports Server supports two of the AWS database services as data sources for reporting:

• Amazon Relational Database Service (RDS)

- Amazon Redshift data warehouse

JasperReports Server can access either of these services when you define a data source with the correct configuration information and credentials. The AWS data source wizard uses the AWS credentials you provide to discover RDS and Redshift data sources. Then it uses those credentials to properly configure security groups to maintain the connection between JasperReports Server and the AWS data source, even when the IP address changes. You can access AWS data sources from both stand-alone server instances that you maintain on your own computers and virtual server instances that you run on Amazon's Elastic Compute Cloud (EC2). For more information, see https://www.jaspersoft.com/cloud.

To create an AWS Data Source:

1. Log into JasperReports Server as an administrator.
2. Click **View > Repository**, expand the folder tree, and right-click a folder to select **Add Resource > Data Source** from the context menu. If you installed the sample data, the suggested folder is Data Sources. The New Data Source page appears.
3. In the Type field, select **AWS Data Source**. The information on the page changes to reflect what's needed to define an AWS data source.

Figure 4-10 Selecting AWS Credentials

4. Under AWS Settings, specify your Amazon credentials in one of the following ways:
   - If your JasperReports Server is running in Amazon's EC2 service, and it has the proper instance role assigned, the server will detect this and automatically use your EC2 credentials. Using the EC2 instance credentials requires that the role was properly set up and assigned when the instance was created. If you're using the EC2 service, we strongly recommend that you use the EC2 credentials.
   - If your JasperReports Server is not running on Amazon's EC2, enter the AWS credentials associated with the RDS or Redshift service. If you don't have AWS keys, click **Generate credentials**, then look for them on the **Outputs** tab for your Stack on the Amazon console:

Figure 4-11 AWS Access and Secret Keys

5. Under Select an AWS Data Source, specify the connection details of the AWS data source that you want to connect to:

    a. Select your AWS Region from the drop-down.

    b. Click the **Find My AWS Data Sources** button.

       The AWS data source queries your environment and displays your available data sources, as shown in the figure below.

    c. Select your data source.

    d. Enter your user name, password, and database name.

       The AWS data source queries your environment and adds the appropriate driver and URL.

Figure 4-12 Select an AWS Data Source Section

6. When you've entered all the information, click **Test Connection**.

    If your connection is successful, a message appears to the right of the button. Sometimes the process takes a few minutes. In that case you'll see an alert. Try the test again after one or two minutes. The test performs the following actions:

    •   Validates the user name and password.
    •   Creates a database security group.
    •   Adds the IP address of your JasperReports Server instance to the security group to authorize ingress to the data service (RDS or Redshift).

    If you want to control details of the security group name or specify the IP address manually because you have a complex VPC Topology, see **8.5, "Configuring Cloud Services," on page 134**. You can also change the default JDBC driver through the configuration.

7. Click **Save**. The Save dialog appears.

8. Enter a name for the data source and a optional description. The Resource ID is generated from the name you enter. If you haven't already specified a location, expand the folder tree and select the location for your data source.

9. Click **Save** in the dialog. The data source appears in the repository.

4.6   Azure SQL Data Sources

Microsoft Azure provides data storage in the cloud as a service. Microsoft offers different types of storage options for Azure users, but only the Azure SQL Server database service is currently supported. JasperReports Server uses the Microsoft SQL Server JDBC driver (tibcosoftware.jdbc.sqlserver.SQLServerDriver), a management certificate, and your credentials to create a secure connection between JasperReports Server and the Azure SQL data source.

4.6.1   Uploading an Azure Certificate File to the Repository

Before you can create an Azure SQL data source, you will need a management certificate to authenticate JasperReports Server with Azure. The management certificate must be a x.509 certificate and can be either self-signed or signed by a trusted certificate authority. The certificate can use a public or private key. You will need to upload the management certificate (.cer) file to the Azure Management Portal and the key exchange (.pfx) file, which contains the server certificate and the key, to JasperReports Server. You can also store a server certificate (.cer) file in the repository.

To upload a certificate file to the repository:

1. Log into JasperReports Server as an administrator.
2. Click **View > Repository** and expand the folder tree.
3. Browse to the folder where you want to save the certificate.
4. Right-click the folder and select **Add Resource > File > Azure Certificate** from the context menu.
5. Click **Choose File** to locate and upload the certificate key exchange (.pfx) or server certificate (.cer) file.
6. Enter a name and resource ID for the file.
7. Click **Submit** to save the file to the repository.

4.6.2 Creating an Azure SQL Server Data Source

The data source wizard uses the Azure credentials that you provide to discover your Azure SQL Server databases. It then uses those credentials to properly configure access rules to maintain the connection between JasperReports Server and the data source. For information on configuring access rules for Azure, see **"Configuring Cloud Services" on page 134**.

To create an Azure SQL data source:

1. Log into JasperReports Server as an administrator.

2. Click **View > Repository**, expand the folder tree, and right-click a folder to select **Add Resource > Data Source** from the context menu. If you installed the sample data, the suggested folder is Data Sources. The New Data Source page appears.

3. In the Type field, select **Azure SQL Data Source**. The information on the page changes to reflect what's needed to define an Azure SQL data source.

Figure 4-13 Entering Azure User Information

4. Under Azure Settings, enter your Azure subscription ID, user certificate (.pfx) file, and user certificate password. Click **Browse** to select the certificate file from your repository. See **"Uploading an Azure Certificate File to the Repository" on page 59** for instructions on uploading the certificate file.

5. Under Select an Azure Database, specify the connection details of the Azure database that you want to connect to:

   a. Click the **Find My Azure Databases** button.

      JasperReports Server queries Azure and displays your available SQL Server databases.

   b. Select your database.

   c. Enter your server name, database name, user name, and database name.

      The Azure data source queries your environment and adds the appropriate URL.

   d. If you have Microsoft's JDBC driver for SQL Server installed on your JasperReports Server instance, you can choose to use it instead of the existing JDBC driver by checking **Use Microsoft Driver**.

Figure 4-14 **Selecting an Azure SQL Data Source**

6. When you've entered all the information, click **Test Connection**.

   If your connection is successful, a message appears to the right of the button. Sometimes the process takes a few minutes. In that case you'll see an alert. Try the test again after one or two minutes. The test performs the following actions:

   • Validates the user name and password.
   • Creates firewall access rules to authorize ingress to the data service.
   • Adds the IP address of your JasperReports Server instance to the access rule.

   If you want to control details of the access rule name or specify the IP address manually, see **"Configuring Cloud Services" on page 134**.

7. Click **Save**. The Save dialog appears.

8. Enter a name for the data source and an optional description. The Resource ID is generated from the name you enter. If you haven't already specified a location, expand the folder tree and select the location for your data source.

9. Click **Save** in the dialog. The data source appears in the repository.

4.7   Cassandra Data Sources

The Apache Cassandra database provides scalability and high availability for certain applications of big data. In JasperReports Server 6.2.1, we included a JDBC driver for the Cassandra database in addition to the native driver supported since JasperReports Server 5.6. We recommend using the JDBC driver for new Casandra data sources and the native driver for existing Cassandra data sources. See JDBC Data Sources for information on creating a JDBC data source. For more information about Cassandra, see http://cassandra.apache.org/.

The native driver for the Cassandra data source has certain limitations on how your data can be structured and accessed:

- The current version of Cassandra does not support NULL values in the data. All required fields must have non-NULL default values. This also means that input controls cannot be null and must be given a value.
- The current version of the driver does not support aggregate functions (sum, min, max).
- For query parameters, the current version of the driver supports $X(IN...), but no other $X functions.

The Cassandra data source supports queries in the Cassandra Query Language 3 (CQL3). To improve performance, design your Cassandra data using the following guidelines:

- Specify the `ALLOW FILTERING` suffix to speed up queries.
- All fields referenced in `WHERE` clauses of a query should be indexed.

As with all big data stores, Cassandra data sources have the following limitations and usage guidelines within JasperReports Server:

- Cassandra data sources are not supported for OLAP connections.
- You must configure your JVM memory to handle the expected amount of data (see the *JasperReports Server Community Project Installation Guide*).

4.7.1   Creating a Cassandra Data Source with the Native Cassandra Driver

1. Log on as an administrator.
2. Click **View > Repository**, expand the folder tree, and right-click a folder to select **Add Resource > Data Source** from the context menu. If you installed the sample data, the suggested folder is Data Sources. The New Data Source page appears.
3. In the Type field, select **Cassandra Data Source**. The information on the page changes to reflect what's needed to define a Cassandra data source.

    You have the option to use attributes in the values of data source parameters. See **4.1, "Attributes in Data Source Definitions," on page 48**.

Figure 4-15 **Cassandra Data Source Page**

4. Fill in the required fields, along with any optional information you choose.

   Use port 9042 with the Cassandra data source. Cassandra's default port of 9160 is for the Thrift client that is commonly used with Cassandra. To use the Cassandra Query Language (CQL) with our Cassandra data source, you may need to configure your Cassandra instance as follows:

   ```
   start_native_transport: true
   native_transport_port: 9042
   ```

5. If you have configured your Cassandra source to be password protected, specify a valid username and password. Due to compatibility issues, Cassandra authentication is supported only when you use Cassandra 1.12.18 and above.

6. Click **Test Connection** to check the values you entered. Make sure that the port is set to 9042, because the connection test will also work with the wrong port (9160).

7. When done, click **Save**. The Save dialog appears.

8. Enter a name for the data source and an optional description. The Resource ID is generated from the name you enter. If you haven't already specified a location, expand the folder tree and select the location for your data source.

9. Click **Save** in the dialog. The data source appears in the repository.

4.7.2   Increasing File Descriptor Limits for Cassandra

Many users have reported errors when viewing multiple reports from a Cassandra data source. Cassandra generally needs more than the default limit of open file descriptors (1024).

To increase the number of file descriptors, administrators need to change the security limits on the Cassandra nodes and on the operating systems running JasperReports Server.

To test this configuration, you can increase the limits for the current session with the following Linux commands:

```
sudo ulimit -Hn 32768
```

or

```
sudo ulimit -Sn 32768
```

The effects of the commands above will be reset when the computer restarts. To make the changes permanent, edit the file /etc/security/limits.conf to add the following settings:

```
* soft nofile 32768
* hard nofile 32768
root soft nofile 32768
root hard nofile 32768
* soft memlock unlimited
* hard memlock unlimited
root soft memlock unlimited
root hard memlock unlimited
* soft  unlimited
* hard unlimited
root soft unlimited
root hard unlimited
```

4.8   MongoDB Data Sources

MongoDB is a big data architecture based on the NoSQL model that is neither relational nor SQL-based. We provide connectors that allow reports to use a MongoDB data source or a MongoDB JDBC data source. As of version 6.2.1, JasperReports Server also supports Kerberos, SSL, and x509 authentication for MongoDB data sources.

As with all big data stores, MongoDB data sources have the following limitations and usage guidelines within JasperReports Server:

• MongoDB data sources are not supported for OLAP connections
• You must configure your JVM memory to handle the expected amount of data (see the *JasperReports Server Community Project Installation Guide*).

4.8.1   Creating a MongoDB Data Source with the Native MongoDB Driver

To create a MongoDB data source with the native driver:

Follow these steps to create a MongoDB data source with the native MongoDB driver.

1. Log on as an administrator.
2. Click **View > Repository**, expand the folder tree, and right-click a folder to select **Add Resource > Data Source** from the context menu. If you have installed the sample data, the suggested folder is Data Sources. The New Data Source page appears.
3. In the Type field, select **MongoDB Data Source**. The information on the page changes to reflect what's needed to define a MongoDB data source.

    You have the option to use attributes in the values of data source parameters. See **4.1, "Attributes in Data Source Definitions," on page 48**.

Figure 4-16 **MongoDB Data Source Page**

4. Fill in the required fields, along with any optional information.

   The MongoDB URI has the form: `mongodb://<hostname>:27017/<database>`

   To enable SSL include the argument `ssl=true` in the URI. For example:

   `mongodb://<hostname>:27017/<database>?ssl=true`

   To enable x509 authentication for the data source include it as well in the URI. For example:

   `mongodb://<hostname>:27017/<database>?ssl=true&authMechanism=MONGODB-X509`

   To enable Kerberos authentication for the data source include the data source and Kerberos user name in the URI. For example:

   `mongodb://<hostname>:27017/<database>?authMechanism=GSSAPI`

   > Before you can enable x509 authentication you need to set up the Java Secure Socket Extension (JSSE) in your application server. Before you can enable Kerberos authentication you need to perform the steps in **4.8.3, "Using Kerberos Authentication with MongoDB Data Sources," on page 68**.

5. Click **Test Connection** to validate the data source.

6. When the test is successful, click **Save**. The Save dialog appears.

7. Enter a name for the data source and an optional description. The Resource ID is generated from the name you enter. If you haven't already specified a location, expand the folder tree and select the location for your data source.

8. Click **Save** in the dialog. The data source appears in the repository.

4.8.1.1 The Jaspersoft MongoDB Query Language

MongoDB is designed to be accessed through API calls in an application or a command shell. As a consequence, it does not have a defined query language. In order to write queries for MongoDB data sources, we have developed a query language based on the JSON-like objects upon which MongoDB operates. JSON is

the JavaScript Object Notation, a textual representation of data structures that is both human- and machine-readable.

The Jaspersoft MongoDB Query Language is a declarative language for specifying what data to retrieve from MongoDB. The connector converts this query into the appropriate API calls and uses the MongoDB Java connector to query the MongoDB instance. The following examples give an overview of the Jaspersoft MongoDB Query Language, with the equivalent SQL terms in parentheses:

• Retrieve all documents (rows) in the given collection (table):

```
{ 'collectionName' : 'accounts' }
```

• From all documents in the given collection, select the named fields (columns) and sort the results:

```
{
  'collectionName' : 'accounts',
  'findFields' : {'name':1,'phone_office':1,'billing_address_city':1,
                  'billing_address_street':1,'billing_address_country':1},
  'sort' : {'billing_address_country':-1,'billing_address_city':1}
}
```

• Retrieve only the documents (rows) in the given collection (table) that match the query (where clause). In this case, the date is greater-than-or-equal to the input parameter, and the name matches a string (starts with N):

```
{
  'collectionName' : 'accounts',
  'findQuery' : {
    'status_date' : { '$gte' : $P{StartDate} },
    'name' : { '$regex' : '^N', '$options' : '' }
  }
}
```

The Jaspersoft MongoDB Query Language also supports advanced features of MongoDB such as map-reduce functions and aggregation that are beyond the scope of this document. For more information, see the language reference on the Community website.

4.8.2 Creating a MongoDB JDBC Data Source

If you want to use your MongoDB data source with an SQL query, create a MongoDB JDBC data source. The MongoDB JDBC driver can create a default normalized schema for your data or, if you prefer, you can load a schema from the repository or your server file system.

To create a MongoDB JDBC data source:

1. Log on as an administrator.
2. Click **View > Repository**, expand the folder tree, and right-click a folder to select **Add Resource > Data Source** from the context menu. If you have installed the sample data, the suggested folder is Data Sources. The New Data Source page appears.
3. In the Type field, select **MongoDB JDBC Data Source**. The information on the page changes to reflect what's needed to define a MongoDB JDBC data source.

   You have the option to use attributes in the values of data source parameters. See **4.1, "Attributes in Data Source Definitions," on page 48**.

Figure 4-17 **MongoDB JDBC Data Source Page**

4. Fill in the required fields, along with any optional information.

5. Specify your Connection Options. For example, if you're using MongoDB 3.0 and you want to enable SSL, enter:

   `EncryptionMethod=SSL;ValidateServerCertificate=false`

   To enable both SSL encryption and self-signed CA, enter the TrustStore and KeyStore paths and the KeyStore password. For example:

   `EncryptionMethod=SSL;TrustStore=<path>;KeyStore=<path>;KeyStorePassword=<password>;`

   The **Auto Generate Schema Definition** check box is checked by default. With this option selected, When you first connect to a MongoDB server, the driver automatically creates a normalized schema of the data and generates a SchemaDefinition for housing and sharing the normalized schema.

6. To specify a schema you've created, uncheck this box and:

a. Use the **File Source** drop down to select your schema file location: **Repository** or **Server File System**. To create a schema using the schema tool, see **"Creating a Schema with the Schema Tool" on page 69**

b. If your file is in the repository, click the **Browse** button and locate it in the repository. If your file is in your server file system, enter the path in the **Server File Location** text box. To upload a schema to the repository, see **"Uploading a Schema to the Repository" on page 69**

7. Click **Test Connection** to validate the data source.

8. When the test is successful, click **Save**. The Save dialog appears.

9. Enter a name for the data source and an optional description. The Resource ID is generated from the name you enter. If you haven't already specified a location, expand the folder tree and select the location for your data source.

10. Click **Save** in the dialog. The data source appears in the repository.

4.8.3 Using Kerberos Authentication with MongoDB Data Sources

The native MongoDB driver can connect to MongoDB Enterprise using Kerberos Authentication. Kerberos is a network authentication protocol that allows clients and servers on a non-secure network to use "tickets" to identify themselves.

To use Kerberos authentication, you need to set up the following:

1. Install the Kerberos client tools on the JasperReports Server host. The client tools are included in the JDK/JRE for Windows. Client tools packages are available for Linux and OS X. You will be prompted to enter information on your Kerberos system during the installation.

2. Download the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy files from Oracle and install them on the JasperReports Server. The files are available for download at the Oracle Java SE Downloads page.

3. Set these Java system properties for JasperReports Server:

```
java.security.krb5.realm=<Kerberos_realm>
java.security.krb5.kdc=<KDC_server_hostname>
javax.security.auth.useSubjectCredsOnly=false
java.security.auth.login.config=<path>/jaas.conf
```

4. Edit the jaas.conf file to add the following:

```
com.sun.security.jgss.krb5.initiate {
    com.sun.security.auth.module.Krb5LoginModule
    required
    useTicketCache=true
    doNotPrompt=true;
};
```

5. Run the `kinit mongodb/[mongo-db-hostname]@<Kerberos_realm>` command on the JasperReports Server host to create or renew a ticket for the Kerberos user. This is the account you will enter as the user name for the data source.

4.8.4   Creating a Schema with the Schema Tool

To create a schema with the schema tool:

1. Go to <js-install>/buildomatic/tools and double click the schematool.jar file.

2. Follow the instructions in the schema tool documentation found here:

    https://documentation.progress.com/output/DataDirect/jdbcmongohelp/index.html#page/mongohelp/starting-the-schema-tool.html

4.8.5   Uploading a Schema to the Repository

To upload a schema to the repository:

1. Click **View > Repository** and expand the folder tree.

2. Right-click a folder and select **Add Resource > File > MongoDB JDBC Schema** from the context menu.

3. Use the **Upload File From Your Local Computer** page to locate and upload your schema file.

4.9   File Data Sources

JasperReports Server 6.1 introduces file-based data sources. These data sources read a text file in JSON, XML, or XMLA format and allow you to query its contents as a relational table. A text file in JSON format may be stored in the server's repository, in the server's file system, while a text file stored in JSON, XML, or XMLA format can be accessed from a URL. This means you can create a data source based on data retrieved from websites and even REST APIs.

By default file data sources do not appear in the New Data Source dialog and must be enabled first.

To enable file data sources in the UI:

1. Open the file .../WEB-INF/applicationContext-remote-services.xml for editing.

2. Locate the element `<util:set id="customDataSourcesToHide">`.

3. Comment out the file data sources you want to display.

4. Save the file and restart JasperReports Server.

In the following example, the JSON and XML data sources are commented out so they appear in the drop-down menu in the New Data Source dialog:

```
<util:set id="customDataSourcesToHide">
    <value>xmlaQueryDataSource</value>
    <value>sparkQueryDataSource</value>
    <!-- value>remoteXmlDataSource</value -->
    <value>mongoDBQueryDataSource</value>
    <!-- value>jsonDataSource</value -->
    <value>jdbcQueryDataSource</value>
    <value>cassandraQueryDataSource</value>
</util:set>
```

After file data sources are enabled, you can create a file data source.

The headers in data source text files must not be blank and cannot contain spaces or special characters other than underscores.

In order to upload a JSON file to the JasperReports Server repository, you must save it with the .xml extension on your computer. Then create an XML file resource and select your JSON.xml file. Give your resource a name such as MyDataFile_JSON so you can find it. The repository calls it an XML file, but the data source can interpret the JSON contents.

To create a file data source:

1. Log on as an administrator.

2. Click **View > Repository**, expand the folder tree, and right-click a folder to select **Add Resource > Data Source** from the context menu. If you installed the sample data, the suggested folder is Data Sources. The New Data Source page appears.

3. In the Type field, select a file data source that you enabled, for example **JSON Data Source** or **XML Data Source**. In this example, we create a JSON data source. The fields on the page change to reflect what's needed to define a file data source.

   You have the option to use attributes in the values of data source parameters. See **4.1, "Attributes in Data Source Definitions," on page 48**.

Figure 4-18 **File Data Source Page for a JSON Data Source**

4. Enter the URI of the JSON file. You can specify a file in the repository with the `repo:` syntax, followed by the repository path of the file. Specify `ftp:`, `http:` or `https:` to use those Internet protocols. To specify a file on your server's file system, specify its path directly, starting from the root, for example /tmp/MyDataFile.json. The user running the server process must have permission to access the file.

5. You can specify a optional query to limit or select the data accessible through the data source. The query has the following syntax:

   <tableName> or <tableName>/<fieldName>

   where:

   <tableName> corresponds to an XML element or JSON structure name

   <fieldName> corresponds to an XML sub-element or JSON field name

6. Click **Save**. The Save dialog appears.

7. Enter a name for the data source and an optional description. The Resource ID is generated from the name you enter. If you haven't already specified a location, expand the folder tree and select the location for your data source.

8. Click **Save** in the dialog. The data source appears in the repository.

Once you have defined a file data source for your XML or JSON file:

*   You can use SQL queries in reports to access the data as a relational table.
*   You can create a Domain on a file data source, allowing you to alter the visibility and names of the fields extracted from the file. On the Display tab of the Domain Designer you can also specify which fields are measures.
*   You can create Ad Hoc views using the Domain based on your file data source, allowing you to explore and interact with data from the file.
*   You can create virtual data sources that combine several file data sources, even different file formats such as XML and JSON, as long as their data structure is compatible so the tables can be joined. You can then create a Domain based on the virtual data source to join the tables and access the joined data in Ad Hoc views and reports.

4.10   Bean Data Sources

The bean data source type is a key extension because it allows you to make use of any custom or exotic data that you might need to report on. Bean data sources serve as a bridge between a Spring-defined bean and a JasperReport. The Spring bean is responsible for providing the data or parameters that fill the report.

To use a bean data source, you must first configure the underlying Spring bean and make it available in the server's web application context. For example, you would add a bean definition to one of the .../WEB-INF/applicationContext*.xml files.

The bean must resolve to a `ReportDataSourceService` instance, either directly or by way of a factory no-argument method. You can use any Spring instantiation method (for example, a constructor or factory) and bean scope (for example, singleton or prototype) for the data source service bean.

The `ReportDataSourceService` instance is responsible for supplying data source parameters to the JasperReport. Custom `ReportDataSourceService` implementations can follow two approaches:

*   If the implementation can provide the data to be used to fill a report, it needs to wrap the data into a suitable `JRDataSource` implementation and pass the data using the `REPORT_DATA_SOURCE` report parameter.
*   If the data comes from the report query by way of a JasperReports query executor, the data source service must set values for the connection parameters defined by the query executor. The connection parameters are usually obtained from the properties of the data source service instance.

For example, you could implement a Hibernate data source service that would be injected in a session factory. The factory would create a Hibernate session that would be passed as a value for the `HIBERNATE_SESSION` parameter. The JasperReports Hibernate query executor then uses the parameter to run the HQL report query.

The `ReportDataSourceService` interface contains two methods: `setReportParameterValues` and `closeConnection`. The former provides data and connection parameter values; the latter is required to close and release any resources or connections created during the call to `setReportParameterValues`.

As of JasperReports Server 6.0, there have been some changes to bean data sources that might affect your custom code. If your custom data sources generate errors after upgrading, see the troubleshooting section **A.10.12, "Upgrading Bean Data Sources," on page 172**.

Once the data source service bean is available through Spring, you can add the bean data source to the repository.

To create a bean data source:

1. Log on as an administrator.

2. Click **View > Repository**, expand the folder tree, and right-click a folder to select **Add Resource > Data Source** from the context menu. If you installed the sample data, the suggested folder is Data Sources. The New Data Source page appears, as shown in the figure below.

3. In the Type field, select **Bean Data Source**. The information on the page changes to reflect what's needed to define a bean data source.

4. Enter the bean name. If the data source service is to be instantiated through a factory method of the Spring bean, you should also enter the name of the method.

New Data Source

Set Data Source Type and Properties

First, select the type of data source you wish to add, then enter the required property values.

Type: Bean Data Source

Bean Name (required):

Hint: Name of Configured Bean

Bean Method (required):

Hint: Name of Method on Configured Bean

Test Connection

Save   **Cancel**

Figure 4-19 Bean Data Source Page

5. Click **Test Connection** to validate the data source. If the validation fails, ensure that the values you entered are correct and that the bean is in the classpath.

6. When the test is successful, click **Save**. The Save dialog appears.

7. Enter a name for the data source and an optional description. The Resource ID is generated from the name you enter. If you haven't already specified a location, expand the folder tree and select the location for your data source.

8. Click **Save** in the dialog. The data source appears in the repository.

CHAPTER 5   OTHER RESOURCES IN THE REPOSITORY

The previous chapter introduced data sources necessary to create any report. But the repository also stores other resources needed by reports. This chapter goes into detail about how to create queries, input controls, and file resources. These are the resources that users reference when uploading a JRXML file to create a JasperReport.

As an administrator, you need to define the resources required for your users' reports. If you have users who are proficient at creating their own reports in Jaspersoft Studio they can upload them as JasperReports to the server. In this case, you'll work with them to prepare the resources required by their reports. For less proficient users, you'll have to create and upload their JasperReports to the server.

For instructions on creating and uploading JasperReports to the server, see the *JasperReports Server User Guide*.

This chapter contains the following sections:

- **Queries**
- **Datatypes**
- **Lists of Values**
- **Input Controls**
- **Query-based Input Controls**
- **Cascading Input Controls**
- **File Resources**

5.1   Queries

A JasperReport uses a query to select the data to be returned from the data source. You can define the query in the JRXML itself, but you can also create and save queries in the repository. A query saved in the repository can be re-used by multiple JasperReports.

Reusing a query enables you to adapt reports to different audiences. The query returns the same data from the same data source every time, but each report presents the data in a different way. Also, separating the query from the JRXML makes it easier to maintain large numbers of reports when the data source changes and the query needs to be updated.

You can also use queries to populate list input controls, as described in **5.5, "Query-based Input Controls," on page 80** and **5.6, "Cascading Input Controls," on page 85**.

To create a reusable query:
1. Login as an administrator.
2. Click **View > Repository** and locate the folder for the query.

3. Right click the folder's name and select **Add Resource > Query** from the context menu. The Add Query page appears.

Figure 5-1 Add Query - Name the Query Page

4. Enter a name for the query. Resource ID is filled in automatically, and the description is optional. Click **Next**. The Link a Data Source page appears.

Figure 5-2 Add Query - Link a Data Source Page

5. Select the data source and click **Next**. Your options are:
    *   **Do not link a data source**. If no data source is associated with the query, the server uses the data source associated with the report that references the query.
    *   **Create a new data source**. You can define a local data source within this query resource that's not accessible to any other resource. Click the link to create any data source as described in **Chapter 4, "Data Sources," on page 47**. This new data source overrides any data source specified in reports that use the query.
    *   **Select data source from repository**. This creates a reference to a data source in the repository. Click **Browse** to select an existing data source. The data source you select overrides any data source specified in reports that use the query.

Click **Next**. The Define the Query page appears.

Figure 5-3 **Add Query - Define the Query Page**

6.   Select a Query Language. For this example choose **SQL**.

The query language Domain (sl) is selected when opening Domain-based queries created in versions of the server before 3.7. It is used only for backward compatibility and should not be selected for new Domain-based queries.

7.   Enter the text of your query in the **Query String** field, for example:

```
SELECT * FROM orders
```

8.   Click **Save**.

By default, JasperReports Server supports SQL, CQL (Cassandra), and MongoDB, while JasperReports Library supports several more (such as EJBQL, xPath and MDX). However, JasperReports Server can support additional query languages if a query executor implementation is properly configured for each additional language when the server is deployed.

You can use a specialized bean data source to support multiple query languages. For information about bean data sources, refer to **4.10, "Bean Data Sources," on page 71**. Another option to add new types of data sources to the server to extend the reach of the JasperReports Server platform. Custom data sources are described in the *JasperReports Server Ultimate Guide*.

5.2   Datatypes

A datatype defines the format of a single-value input control, for example text or numerical. Datatypes determine what users can enter in the input fields that correspond to parameters in a report:

•   Text
•   Number
•   Date
•   Date/time

To create a datatype:

1. Log on as an administrator.

2. Click **View > Repository**and locate the folder for the datatype.

3. Right click the folder's name's name and select **Add Resource > Datatype** from the context menu. The Add Datatype page appears.

Add Datatype

Set the Datatype Kind and Properties

First, select the kind of datatype you wish to add, then enter the required property values.

Type: Text

Name (required):
My Datatype

Resource ID (required):
My_Datatype

Description:
Sample text datatype

Pattern:

Save   Cancel

Figure 5-4 Add Datatype Page

4. Enter a name and optional description for the datatype. The resource ID is filled in automatically.

5. Select the datatype and provide related information. Your options are:
    - Text – You can specify a regular expression in the **Pattern** field. The expression is used to validate the text the user submits. For instance, you could enter an expression that tests for email addresses.
    - Number – With numerical datatypes, you can control the range of acceptable values by specifying minimum and maximum values and whether the specified values are themselves acceptable (**Minimum is Strict** and **Maximum is Strict** check boxes). If you select a **Strict** check box, the specified value is *not* acceptable.
      For instance, for a percent field, you might specify a minimum of 0 and a maximum of 100. If you do not want to accept 0 percent, you would check **Minimum is Strict**. If you want to accept 100 percent, you would clear **Maximum is Strict**.
    - Date and DateTime – Click the calendar icon to the right of the Minimum and Maximum date or date time fields to choose their values.

6. Click **Save**. The datatype resource appears in the repository.

5.3   Lists of Values

A lists of values resource is a static list of values for single-select or multi-select input controls. Depending on the type of input control, the user selects one or more of these labels using radio buttons, check boxes, or drop-down lists.

To create a list of values:

1. Log on as an administrator.
2. Click **View > Repository** and locate the folder for the new list of values.
3. Right click the folder's name and select **Add Resource > List of Values** from the context menu. The Add List of Values page appears.

Figure 5-5 Add List of Values Page

4. Enter a name for the list of values. Resource ID is filled in automatically, and the description is optional
5. Enter the name and value for each item in the list and click **Add**.

   The name and value are both treated as strings. Users see the label only in an input control that uses the list, and the report receives only the value. To remove an item, click Remove beside the value.
6. Click **Submit**. The list of values resource appears in the repository.

5.4   Input Controls

Any JasperReport can be parameterized so that its generated output is a function of values given at run time (query filters), or so that its layout is changed to accommodate different users (such as changing the title).

When writing JRXML, you can declare parameters and accommodate any run time value that needs to be passed into the query executor, the rendering engine, or the calculation engine. However, the parameter information in a JRXML file does not provide everything JasperReports Server needs to build a complete user interface and prompt users for values. You must also define an input control resource that defines the following:

• The range of possible values or list of discrete values allowed.
• The type of input, for example single-select or multi-select, and the widget to display the possible values, for example drop-down list or check boxes.
• Display options such as labels and whether the value is required.
• The name of the corresponding parameter in the JRXML.

When a user runs the report, the server uses this information to prompt the user for a value and to validate that input. For example, consider a report that returns sales data for all of a company's products; the user might input a product to view by selecting a product name in a list.

JasperReports Server supports several types of input controls, each of which can map to certain types of parameters in the report's JRXML. The input control also determines the kind of widget the user interacts with:

- Boolean – Presented as a check box. These input controls return a `java.lang.Boolean` object to the report engine in response to the user's selection. Boolean input controls return .TRUE or FALSE as values, depending on whether the box is checked.
- Single value – Presented as a free-form text box. You must specify a datatype, for example text or numerical value, and the user's entry is validated against this datatype.
- Single-select – Presented either as a drop-down list or a set of radio buttons. A single-select input control returns a single value.
- Multi-select – Presented as a scrollable list of values or a set of check boxes. A multi-select input control returns a collection of values.

One advanced feature of single-select or multi-select input controls is that the values they present can be the result of a dynamic query. The query retrieves actual values from the data source before presenting them as choices to the user. These queries can contain parameters themselves, for example based on the logged-in user or the selection of a previous input control. Query parameters are described in **5.5, "Query-based Input Controls," on page 80** and **5.6, "Cascading Input Controls," on page 85**.

Input controls rely on other resources in the repository, such as datatypes, static lists of values, or queries. You can manage these resources the same way you manage other resources. You can define them locally (available only to the input control) or reference them externally (reusing a resource in the repository). For more information, see **3.2.3, "Local Resources and External References," on page 34**.

As with other resources, input controls can be created locally as part of a JasperReport, in which case they cannot be seen outside that JasperReport, or they can be created separately in the repository and referenced in multiple reports.

To use an input control in a report, the control must meet two conditions:

- The parameter name in the input control must correspond to the name of the parameter in the report. No error occurs for a mismatch, but at run time `NULL` is passed instead of actual value of the parameter.
- The input control and its corresponding parameter must be of compatible datatypes (for example, both must be text types or date types). If there is a mismatch, the report fails and an exception is returned.

This section explains how to create an input control in the repository. To reference input controls in a JasperReport, see the *JasperReports Server User Guide*.

To create an input control:

1. Log on as an administrator.
2. Click **View > Repository** and locate the folder for the input control.
3. Right click the folder's name and select **Add Resource > Input Control** from the context menu. The Add Input Control page appears.

Figure 5-6 Add Input Control Page

4. Select the type of input control from the **Type** list. In this example, **Single Value** is selected.

5. Enter the prompt to tell users how to use the control. This example, uses the prompt `Select the text for the report title`.

6. In practice, the prompt text is often the same as the parameter, so the parameter name is automatically filled in. If you have used a different prompt, edit the Parameter Name field and enter the exact name of the parameter for your control. Remember, the parameter name must be the same here as in the reports that use this input control.

   For this example, the parameter name is `title`. Description is optional.

7. Select options for the control. Your options are:

   • **Mandatory** – Forces the end user to supply a value.

   • **Read-only** – Displays the value of the parameter without allowing the end user to modify it.

   • **Visible** – Makes the input control visible in the report options dialog.

8. Click **Next**.

   Subsequent pages depend on what type of input control you chose:

   • Boolean types do not require any further information.

   • Single-value types require a datatype the user can enter.

   • Single-select and multi-select types based on static lists require a list of values.

   • Single-select and multi-select types based on queries require a query.

9. In this single-value example, the Locate Datatype page appears. Choose the option to select a datatype from the repository and click Browse. In the repository dialog, select `/datatypes/TextGeneralDatatype`, which is similar to the datatype we created in **5.2, "Datatypes," on page 75**.

   📝 If you choose to define a datatype, the wizard takes you through the same procedure as in section **5.2, "Datatypes," on page 75**. You can then define any datatype you need, but it's local to the input control and not reusable in other input controls.
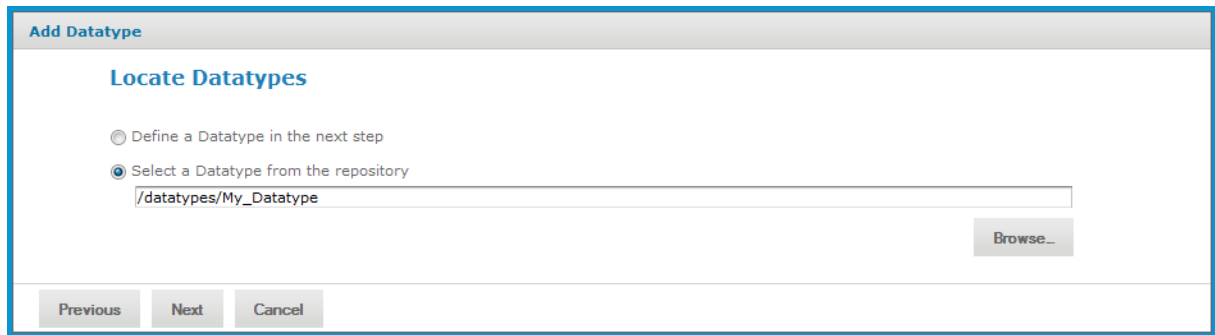
Figure 5-7 **Locate a Datatype for an Input Control**

10. Click **Next**. The input control resource is created in the repository.
11. Locate the input control in the repository manager. Notice that the text of the prompt that you entered in **step 5** is also used as the name for the resource.

5.5 Query-based Input Controls

Query-based input controls display a dynamic set of values for the user to choose from. They are input control resources in the repository, but instead of being based on a datatype or a static list of values, they perform a query to retrieve a list of values. For example, a report could have a city parameter, and the query-based input control could display the list of cities in your data. Because the queries use standard syntax, you can include filters in a WHERE clause, for example, to restrict the list of cities to a certain country.

5.5.1 Creating a Query-based Input Control

In this first example, we create a query-based input control that returns a list of all cities the user can choose from.

1. Log in as an administrator.
2. Browse the repository and select the folder for the query-based input control.
3. Right click the folder's name and select **Add Resource > Input Control**. The Add Input Control dialog appears.

Figure 5-8 **Adding an Input Control - Naming**

4. Select the type of query-based input control from the type drop-down list. This determines how the input control appears to users, either as a drop-down list, a set of radio buttons, a multi-select list, or a set of check boxes. In this example, we choose a single-select query-based input control type.

5. Specify the prompt text, parameter name, optional description, and appearance options in the same manner as when defining a regular input control.

6. Click **Next**. Because we selected one of the query-based types, the Locate Query page appears:

Figure 5-9 **Adding an Input Control - Locating the Query**

If you have a suitable query defined in the repository, you can select it here as an external reference. In this example, we'll define a query locally inside the input control.

7. Click **Next** to define the local query resource. The query naming dialog appears:

Figure 5-10 Adding an Input Control - Naming the Query

Although the query resource is not visible in the repository, it may still have a name, ID and optional description within the query resource. However, the values for these fields are not important.

8. Enter any name, and the ID is filled in automatically. Then click **Next**. The link data source page appears:

Figure 5-11 Adding an Input Control - Linking to a Data Source

Like all queries, the query inside the input control may optionally link to a data source, either in the repository or its own internally defined one. If no data source is linked, the query in the input control uses the same data source as the report. In this example, we use the default of not linking to a data source.

9. Click **Next**. The query definition page appears:

Figure 5-12 **Adding an Input Control - Defining the Query**

10. Select the query language, in this example SQL, and enter a query string. The SELECT statement should contain the names of all fields used in the display, value, or filter for the input control. In this example, the query returns three fields, country, state, and city. Country limits the values to a single country. The ORDER BY clause ensures that the values from the query are sorted alphabetically when they appear in the input control.

11. Click **Save** to complete the query definition. The parameter values page appears:

Figure 5-13 **Adding an Input Control - Setting Parameter Values**

On the parameter values page, you specify which fields in the query result are displayed, and which field contains values that become the parameter value. when chosen.

a. First, specify the value column. This is the field whose value is passed to the report. The datatype of the field must match the type of the corresponding parameter in the report.

b. Next, specify the visible columns. These are the fields whose values appear in the input control the user chooses from. In the simplest case, enter the same field as the value column. If you add multiple fields to the visible columns, the input control displays the fields together, in the order listed, separated by a vertical bar (|). In this example, the user can see and choose from:

Los Angeles | CA
San Francisco | CA
Denver | CO

Only the city value (without the state) is passed to the report. Showing additional fields in this way can help users find the value they want in long lists of results.

The value and display columns may also be entirely different, for example, displaying the full name of a sales representative, but using the employee ID as the value returned by the input control. The only restriction is that all fields used in the value or display list must be selected by the query.

5.5.2 Built-in Parameters for Query-based Input Controls

The `LoggedInUser` and `LoggedInUsername` parameters are always available for query input controls; they're always available to reports, as well, even if an input control isn't defined for them. The standard parameters are also provided for reports if they're defined as parameters in the JRXML.

Table 5-1 Built-in Parameters for Query-based Input Controls

Parameter Name	Type	Notes
LoggedInUser	User	The currently logged in user. This parameter isn't available in query input controls, but is used as a parameter to the report.
LoggedInUsername	String	The user name of the current user.
LoggedInUserFullName	String	The full name of the current user.
LoggedInUserEmail Address	String	The email address of the current user.
LoggedInUserEnabled	Boolean	Indicates whether the current user is enabled.
LoggedInUserExternally Defined	Boolean	Indicates whether the current user is authenticated externally.
LoggedInUserRoles	Collection <String>	The roles assigned to the current user. This is helpful for parameters that use $X.

Attributes defined on users or at the server-level can also be used in reports and query-based input controls. For more information, see **2.3, "Managing Attributes," on page 24**.

Table 5-2 Attribute-based Parameters for Queries and Reports

Parameter Name	Type	Notes
LoggedInUserAttributes	Map<String, String>	The attributes of the logged-in user. This parameter isn't usable in query input controls, but it is used as a parameter to the report. If the user has no attributes, the parameter is an empty map.
LoggedInUserAttribute Names	Collection <String>	The names of the attributes of the logged-in user. This is helpful for parameters that use $X. If the user has no attributes, the parameter is an empty map.
LoggedInUserAttribute Values	Collection <String>	The values of the attributes of the logged-in user. This is helpful for parameters that use $X. If the user has no attributes, the parameter is an empty map.
LoggedInUserAttribute_ <attribute-name>	String	For the logged-in user, the value of the attribute matching the name passed as <attribute-name> (such as att1). If there's no attribute with this name for this user, the parameter is empty. This parameter is available only if it's defined in a query or as a report parameter.
ServerAttribute_ <attribute-name>	String	The value of the named attribute defined at the server level. If there's no attribute with this name at the server-level, the parameter is empty. This parameter is available only if it's defined in a query or as a report parameter.
Attribute_ <attribute-name>	String	The value of the named attribute is determined hierarchically. The first value found for the named attribute in the following order: • On the logged-in user. • At the server level. If there's no attribute with this name defined in those locations, the parameter is empty. This parameter is available only if it's defined in a query or as a report parameter.

5.6 Cascading Input Controls

A cascading input control is one whose values depend on the selection made in a previous input control. You create a cascading input control using parameters in the query string of a related input control. In other words, the parameter you defined for an input control can be used in another query-based input control.

In a query-based example of cities and states such as:

Los Angeles | CA
San Francisco | CA
Denver | CO

the query may still generate a list of hundreds of cities to scroll through. Even though each city is easy to identify with the state, scrolling through a long list is time consuming. With cascading input controls, this example can have two input controls, one for the state and one for the city: The city input control is empty until the user selects a state.

- When input controls are displayed, the query for the state input control returns an alphabetical list of unique state names.
- When the user selects a state, the query for the city input control is triggered and returns the list of cities in that state. When the user selects one and submits it, the city name is passed as a parameter to the report.

For an especially large number of cities, you can use more cascading input controls, such as region of state, to reduce the list.

The parameter values determined by each cascading input control may or may not be used in the report. For example, if the report only shows data about a city, the country input control exists only to speed up the choice of city. However, if the report also shows information such as city average compared to country average for a given measure, the country parameter is also used in the report.

5.6.1   Parameters in Input Control Queries

Parameter substitution in query input controls follows the same approach as for JasperReports queries. Queries of all types of data sources can use parameter substitution, and $P, $P! and $X (for SQL queries) parameters are supported. The $X notation has two principal forms explained in the following list:

- `$P{parameter_name}`
  The value of the parameter is substituted into the query. In cases where the parameter contains a string, the substitution inserts the proper escape characters to create valid SQL. Use this for single-select input controls and simple comparison operators such as greater-than or less-than. For example:

  ```
  select name from EMPLOYEES where deals_closed > $P{DEALS}
  ```

  Do not use $P{parameter_name} with equality because the parameter value can be null, and `field = NULL` is not valid SQL. Instead use $X{EQUALS, ...} as explained below.

- `$P!{parameter_name}`
  The value of the parameter is treated as raw text. The server replaces the parameter with the value of the input control without performing extra checking or value escaping. This is used in complex cases where the input control provides a piece of the query or sometimes the entire query.

- `$X{EQUALS, column, parameter_name}` or `$X{NOTEQUAL, column, parameter_name}`
  This syntax performs equality verification and handles the case when the parameter value is null. Use this everywhere instead of the old `column = $P{parameter_name}` syntax. The $X{EQUALS...} syntax performs the following substitution before submitting the query:

  ```
  column = parameter_value -- when parameter_value is non-null

  column IS NULL -- when parameter_value is NULL
  ```

- `$X{IN, column, parameter_name}` or `$X{NOT IN, column, parameter_name}`
  Use this parameter for cascading with multiple-select input controls. The $X{IN...} operator is true when the field value matches any one of the multiple values of the input control. In the country/cities example, we can allow the user to pick any number of countries, and show all the cities in the selected countries. The query-based input control would have the following query:

  ```
  select city from ACCOUNTS where $X{IN, country, COUNTRIES}
  ```

If the user selects the values Canada, Mexico, and USA in the COUNTRIES multi-select input control, the `$X{}` syntax translates into the following query for the CITIES input control:

```
select city from ACCOUNTS where country IN ('USA','Canada','Mexico')
```

When defining these parameters in a report, don't use a `defaultValueExpression` element. Due to a limitation in JasperReports Server, these parameters are null when a `defaultValueExpression` is provided.

The $X syntax also supports the following operators. They are all designed to handle null input by generating 0 = 0 when the parameter value is null:

Parameter Syntax	Meaning
$X{GREATER, column, parameter}	column > parameter
$X{[GREATER, column, parameter}	column >= parameter
$X{LESS, column, parameter}	column < parameter
$X{LESS], column, parameter}	column <= parameter
$X{BETWEEN, column, start_param, end_param}	start_param < column < end_param
$X{BETWEEN], column, start_param, end_param}	start_param < column <= end_param
$X{[BETWEEN, column, start_param, end_param}	start_param <= column < end_param
$X{[BETWEEN], column, start_param, end_param}	start_param <= column <= end_param

For more information on using $P, $P! and $X to build dynamic queries, see the *JasperReports Library Ultimate Guide* and the *Jaspersoft Studio User Guide*.

Any number of parameters can be used in a query, just as any number of input controls can be defined in a JasperReport. In addition to the standard input control parameters, a cascading input control query can use the built-in parameters described in **Table 5-1, "Built-in Parameters for Query-based Input Controls," on page 84**.

5.6.2   Creating a Cascading Input Control

In this example, we'll create a cascading input control to allow users to report on orders for a specific city by selecting a country and then a city within that country. Our example uses Jaspersoft Studio to edit a JasperReport on the server. The report uses sample data from the SugarCRM database shipped with JasperReports Server.
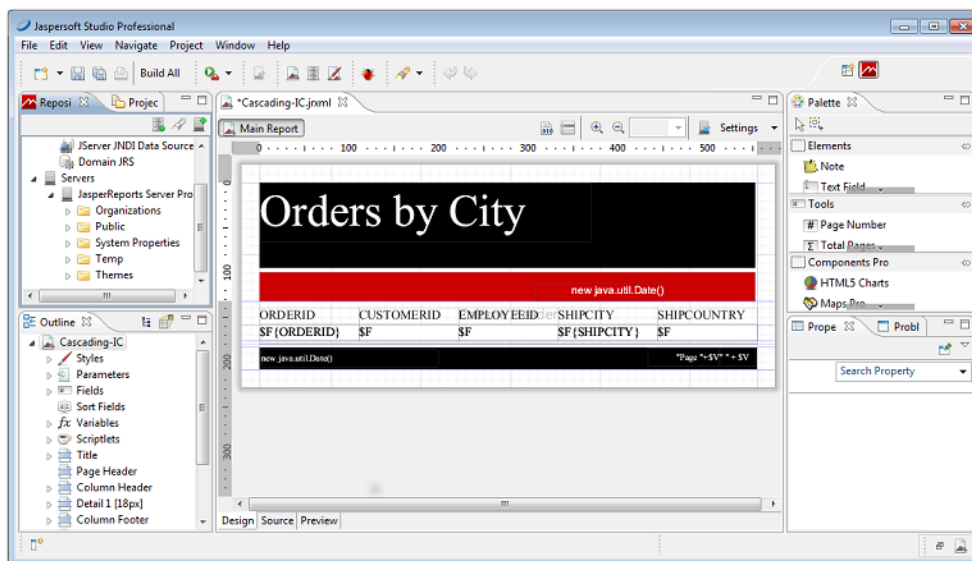
Figure 5-14 Simple Report Filtered by City

You can connect Jaspersoft Studio to JasperReports Server. Once connected, the reports on the server will appear in the repository tree in Jaspersoft Studio. For more information see the *Jaspersoft Studio User Guide*.

We'll start with a report containing the following dataset fields: `ORDERED`, `CUSTOMERID`, `EMPLOYEEID`, `SHIPCITY`, and `SHIPCOUNTRY`.

Our sample report is based on the following query that requires the user to enter a city:

```
select * from orders where SHIPCITY=$P{City}
```

Creating the Country input control:

1. On the repository tab, right click your report's folder and select **New** from the context menu. The Add JasperReports Server Resource wizard appears.

2. Select **Input Control** and click **Next**.

3. Enter Country for the **Name** of the input control and click **Next**.

4. Select the query type **Single Select Query**.

5. Click the ellipsis button next to the **Local Resource** field. The Query window opens.

6. Enter the SQL query:

```
select distinct SHIPCOUNTRY from orders order by SHIPCOUNTRY
```
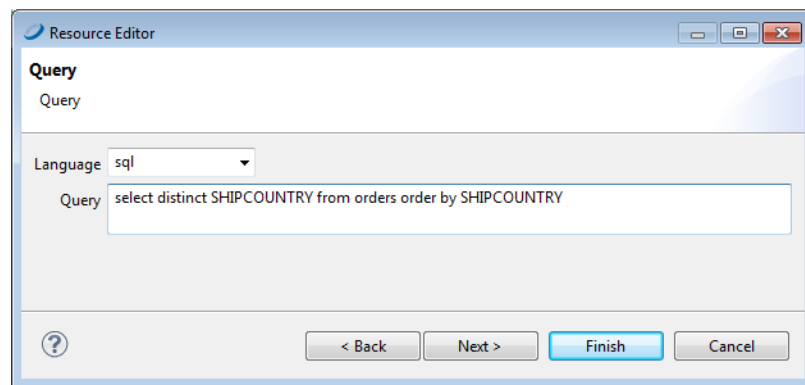
Figure 5-15 Query for the Country Input Control

7. Click **Next**. The Data Source window opens.

8. Click the **Data Source from Repository** option, select your data source, then click **Finish**.

9. On the **Value and Visible Columns** tab, enter SHIPCOUNTRY in the **Value Column** field and click **Add**. This is the column whose value is returned as the value of the Country input control.

10. Under **Visible Query Columns**, enter SHIPCOUNTRY. This is the column whose values populate the list of cities the user can choose from. In our case it is the same as the value column.

11. Click **Finish**.
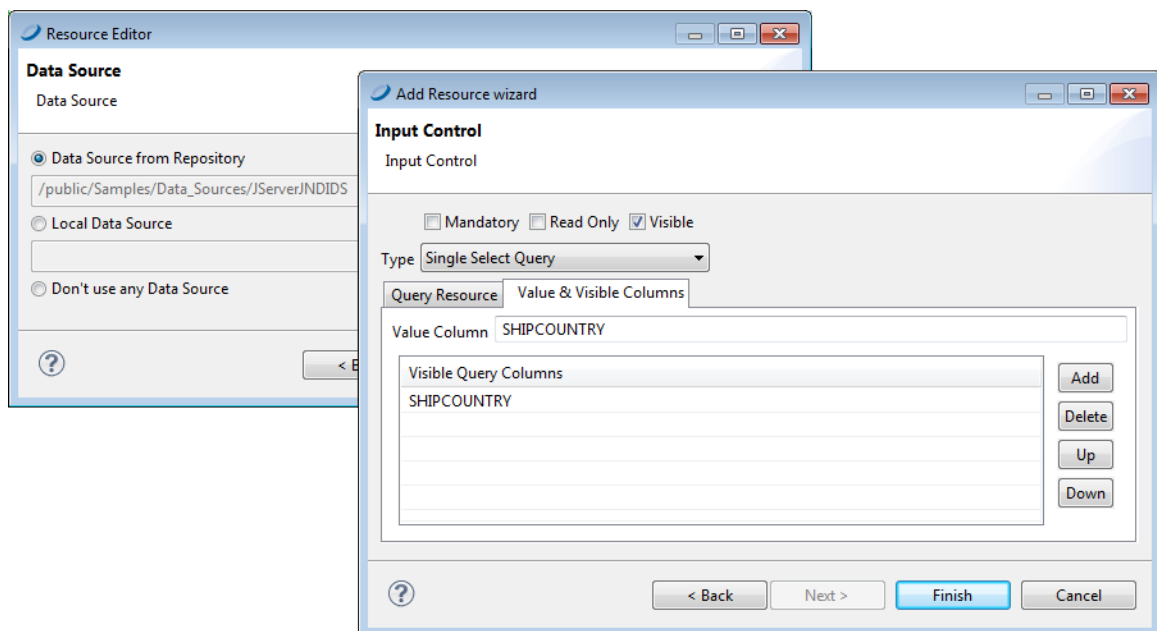
Figure 5-16 Country Control Value and Visible Columns

Creating the City input control:

1. Create a new City input control the same way you created the Country input control, using **Single Select Query** type.

2. Define an SQL query that uses the value of Country to derive valid values for City.

```
Select distinct SHIPCITY from orders where SHIPCOUNTRY = $P{Country} order by SHIPCITY
```
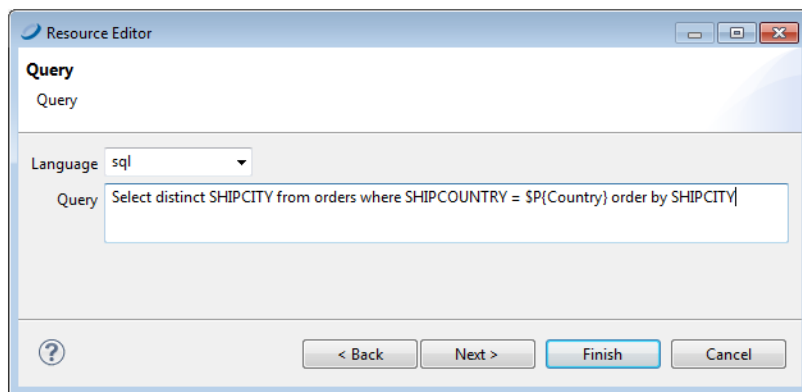

Figure 5-17 Query Using Country to Select Cities

3. Click **Next**. The Data Source window opens.
4. Click the **Data Source from the Repository** option, select your data source, then click **Finish**.
5. On the **Value and Visible Columns** tab, enter SHIPCITY in the **Value Column** field and click **Add**. This is the column whose value is returned as the value of the City input control.
6. Under **Visible Query Columns**, enter SHIPCITY. This is the column value used to create the list of cities the user can choose from.
7. Click **Finish**.
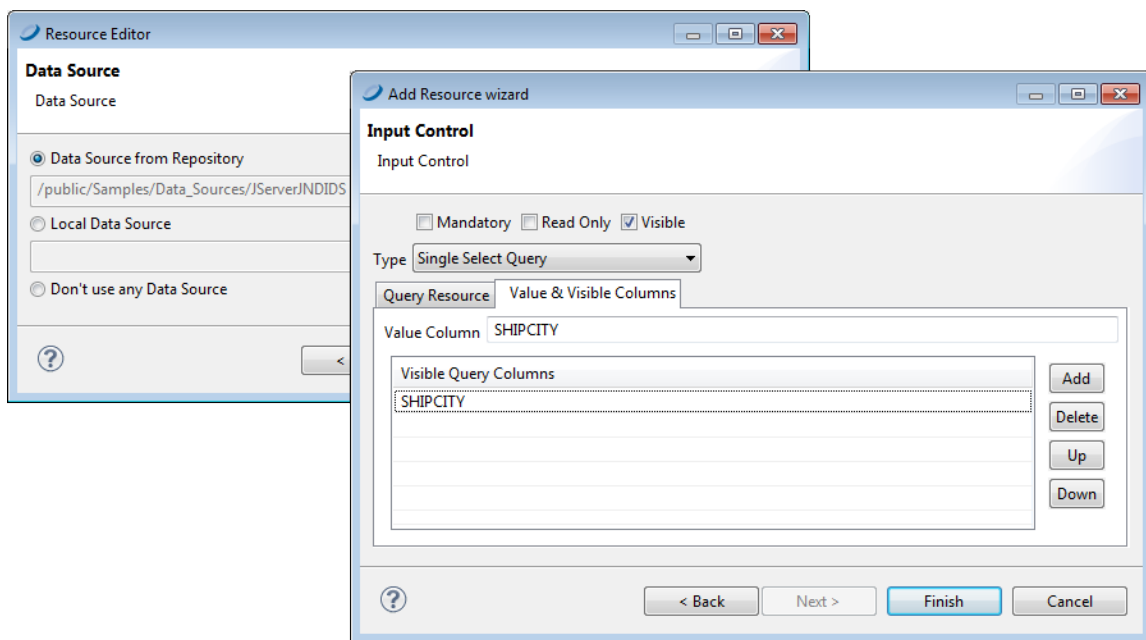
Figure 5-18 City Control Value and Visible Columns

8. Publish your report to JasperReports Server.

Now when you run your report in JasperReports Server, You'll see an input control for country and another for city. After you select the country, the values available for city are those within that country. Notice that the Country value is not passed to the report. It is used only to help the user select a city.
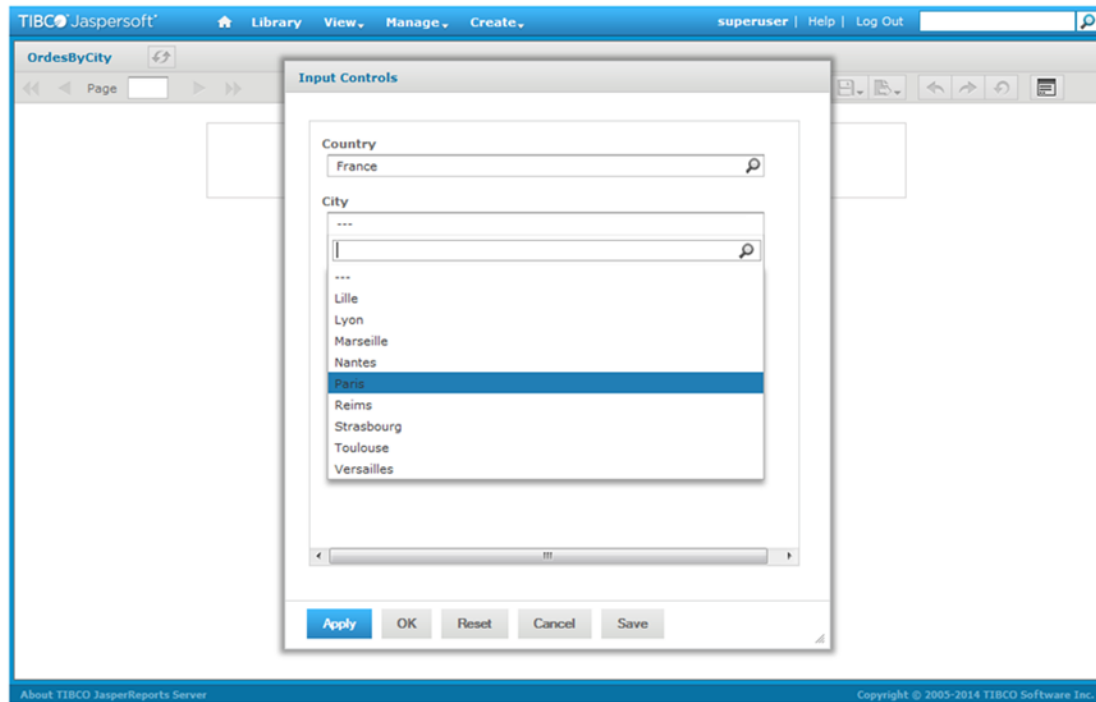
Figure 5-19 **A Report with a Cascading Input Control**

5.7   File Resources

You create file resources by uploading files so they can be referenced by Jasper Reports. JasperReports Server supports the following file types:

Table 5-3 File Resource Types

File Type	Description
CSS	Cascading Style Sheet file that helps define the user interface as part of a theme.
Font	True Type font (.ttf) file to extend the set of fonts available in a report and allow embedding of fonts in the PDF output, if needed (see **5.7.1, "Fonts," on page 92**).

File Type	Description
Image	Any image format supported by the JVM (Java Virtual Machine), such as JPEG, GIF, and PNG. Image files can be referenced in JasperReports, and also in CSS files.
JAR	Libraries that provide functionality for your reports (see **5.7.2, "JAR Files," on page 93**).
JRXML	The definition of a report in JasperReports' XML-based report definition language. A JRXML file can be uploaded separately for use in multiple JasperReports.
OLAP Schema	Defines the data in an OLAP cube, including how to aggregate the dimensions.
Resource Bundle	A Java .properties file containing key-value pairs for localization of reports (see **5.7.3, "Resource Bundles," on page 93**).
Style Template	A JRTX file containing a style template that can be shared among JasperReports.
XML	XML file used in Domains and analysis to define data-level security. Can also be used for XML and JSON data files (see **4.9, "File Data Sources," on page 69**).
Azure Certificate	A x.509 server certificate (.cer) or key exchange (.pfx) file used to authenticate JasperReports Server with Microsoft Azure (see **"Uploading an Azure Certificate File to the Repository" on page 59**).
Secure File	A SSH private key file for SFTP file transfers that require a SSH key (see **"Uploading an SSH Private Key File to the Repository" on page 95**).

The way in which fonts, JAR files and resource bundles are associated with reports is further explained in the following sections.

5.7.1   Fonts

The server relies on the JasperReports Library as its content rendering engine, which enables it to produce high-quality, pixel-perfect documents. The server can use any of the fonts available to its JVM as logical or physical fonts. This solution is perfect for HTML reports that are stored in the server.

However, when exporting the report to PDF, you may need to take additional steps if the report includes fonts that the PDF viewer doesn't recognize, or if the report requires fonts that your users do not have on their computers. In this case, you must embed the font in the PDF file itself. To embed a font, you must edit the report's main JRXML file; the TTF (True Type Font) file that the report references must be available to the server at run time. One way to ensure that the server has the correct font is to upload it to the repository by creating a file resource. Then, the report can refer to the font's URI in the repository.

For details about working with fonts and PDF export, refer to the JasperReports documentation.

5.7.2   JAR Files

JasperReports can leverage third-party APIs. When run, reports can make direct API calls to third-party code using JRXML expressions. This provides enormous flexibility for incorporating business logic or other utility code into report generation.

In some cases, you can make the third-party code available to the report generating process by adding the necessary libraries to the server's application classpath when it's deployed. In other cases, upload the third-party or additional JAR files to the repository by creating a file resource. Then the report can refer to the code by referencing them as additional file resources.

5.7.3   Resource Bundles

When a single JRXML template is used to generate documents in multiple languages, it needs a resource bundle to accommodate the locale-specific content. If you upload such resource bundles by creating a file resource, your JRXML files can refer to them.

The name of the resource bundle created as a file resource in the repository must have .properties as its file extension. For example, the default resource bundle might be named MyReport.properties, and its French translation MyReport_fr.properties. For more information about resource bundles for reports, refer to the *JasperReports Server User Guide*.

5.7.4   Creating a File Resource

Administrators should organize file resources into folders in the repository to make them easier to find when creating references.

To add a file resource:

1. Log in as an administrator and select **View > Repository**.
2. In the Folders panel, right-click the parent folder's name and select **Add Resource > File** from the context menu , and select a resource type. In this example, **Font**. The Add File dialog appears.
3. Enter the required information for the file resource. In additions to the name and ID, file resources only require you to enter the path to a file. Click **Browse** to locate a file on your file system.

   The figure below shows the dialog for adding a Font file. All file resources are created by uploading a file in this way.

Figure 5-20 Adding a File Resource

4. When done, click **Submit**. The new file resource appears in the selected folder in the Repository panel.

5.7.5 Editing a File Resource

The following example shows how to edit a file resource.

To edit a file resource:

1. Log on as an administrator.
2. In the repository, browse or search for the resource.
3. Right-click the resource and select **Edit** from the context menu. In this example, we edit the font resource created in **5.7.4, "Creating a File Resource," on page 93**.

Figure 5-21 **Editing a File Resource**

4. Use the Edit dialog to view or modify the resource definition and its values. In the figure above, you can see how the Description field was changed. You can also change the contents of the file resource by specifying another file to upload. The Path to File field is not required unless you want to reload the file from disk.

5. Click **Submit** to save any changes.

5.7.6 Uploading an SSH Private Key File to the Repository

JasperReports Server allows you to export scheduled reports or dashboards over SFTP to an FTP server. If the FTP server uses SSH key authentication, you will need to upload a SSH private key file to the repository as a Secure File resource.

While the SSH private key is being used in a scheduled report or dashboard job, it cannot be deleted from JasperReports Server.

To upload an SSH key file to the repository:

1. Log into JasperReports Server as an administrator.
2. Click **View > Repository** and expand the folder tree.
3. Browse to the folder where you want to save the SSH key.
4. Right-click the folder and select **Add Resource > File > Secure File** from the context menu.
5. Click **Choose File** to locate and upload the SSH key file.
6. Enter a name and resource ID for the file.
7. Click **Submit** to save the file to the repository.

CHAPTER 6 THEMES

A theme is the set of all CSS (Cascading Style Sheets) and associated images that define the appearance of the user interface. Themes are stored in the Themes folder in the repository, with special menus on theme folders for activating, uploading, and downloading a theme. You can store any number of themes in the repository and switch among them, to change the UI's appearance.

> The set of files in the default theme was updated in 6.1. Custom themes developed before 6.1 may require upgrading to work with the new set of files. For more information see the upgrade procedures in the *JasperReports Server Community Project Upgrade Guide*.

This chapter contains the following sections:
* **Introduction to Themes**
* **How Themes Work**
* **Administering Themes**
* **Creating Themes**
* **Working With CSS Files**

6.1   Introduction to Themes

The default appearance of the JasperReports Server user interface (UI) can easily be modified to suit your needs.

The UI's appearance is determined by CSS (Cascading Style Sheets). A theme is a collection of CSS files and associated images that specify the appearance for all or part of the UI. A theme controls only the UI's looks, such as fonts, colors, lines, spacing, and images. It has no effect on content or functionality.

Themes are defined globally at the repository root. Only administrators can set the theme. Administrators can add, upload, edit, copy, and delete the files that make up the theme, just like other resources in the repository. The repository allows special actions on theme folders for downloading and uploading themes as ZIP (compressed archive) files, and for activating the theme. Themes are active immediately without needing to restart the server.

Themes are very flexible, allowing administrators to easily change the global appearance. For example, all of the following scenarios are possible with themes:

Scenario	Description
Use the default theme unchanged.	If the default theme suits your needs, there's no need to customize it or develop new themes.
Quickly modify the default theme.	You can specify overrides of individual CSS rules or replace images in the default theme. After creating and uploading the new files, your theme is active immediately.
Create an entirely new theme.	With CSS experience or Jaspersoft Professional Services, you can change the entire look and feel of the server. You can create a custom theme to match or blend with nearly any other web design.

6.2   How Themes Work

Themes are stored in a special folder named Themes at the root of the repository. The Themes folder contains a default theme that can't be edited and any number of custom theme folders. Each theme is stored in its own folder and is known by the name of the folder.

The folder named default in the Themes folder is a special theme whose contents are controlled by the server. The default theme contains the complete definition of every style that makes up the default theme shipped with JasperReports Server. The default theme folder cannot be modified, even by administrators.

> You cannot modify the files of the default theme through the repository. If you try to do so by circumventing the repository, you could inadvertently change rules and make the UI unusable. If this happens, you'll need to re-install JasperReports Server to recover.

This chapter uses the following terminology to distinguish between themes.

Name	Folder	Description
Default theme	root > Themes > default	The unmodified UI as it appears at installation. The default theme is defined in the default folder in the Themes folder at the root of the repository.
Active theme	root > Themes > *active-theme*	The active theme set at the root level. Users see a combination of the active theme and the default theme, depending on the files in the active theme and the inheritance rules.

If no custom theme is made active, the default theme is the active theme.

The following figure shows the files of the default theme in the Themes folder at the root of the repository. The name of the folder and its subfolders are bold indicating that it's the active theme.

Figure 6-1 Contents of the default Theme

6.2.1   Theme Files

A complete theme consists of the files listed for the default theme, as shown in the previous figure, along with all referenced images. The file samples.css is used only by the **View > UI Samples** page described in section **6.5.5, "User Interface Samples," on page 107**. The files overrides_ie7.css and overrides_ie8.css are only loaded with the style sheets when the user's browser is Internet Explorer 7 or 8, respectively.

> The set of files in the default theme was updated in release 6.1. Custom themes developed before 6.1 may require upgrading in order to work with the new set of files. For more information see the upgrade procedures in the *JasperReports Server Community Project Upgrade Guide*.

The images associated with a theme include all the icons in the user interface and backgrounds for buttons and borders. Several icons and backgrounds can be stored in the same file called a sprite. The theme also includes the favicon.ico file that appears on browser tabs. There are over 70 image files in the default theme.

The image files for the default theme are stored in a folder named images. In a custom theme, there are two ways to change an image of the default theme:

- Create a folder named images and an image file with the same name as the one you want to replace.
- Modify the corresponding CSS rules to specify the name and location of a different image.

When you modify the CSS rules, you can use any of the following ways to reference image files or any other helper file:

- Directly in the theme folder. In this case the file is referenced without a path, for example "myfile.png" in CSS.
- In any folder path located in the theme folder. For example, your custom CSS file could refer to "MyImages/myfile.png" if you create a folder named MyImages in the theme folder and upload your images there.
- Anywhere on the Internet. Following the CSS standard, your custom CSS can refer to images or any helper file with a regular URL.

6.2.2 Inheritance

In order to render the user interface, JasperReports Server must load each of the CSS files listed in **Figure 6-1**. Because each file can be stored in multiple themes, inheritance determines which file to load. To locate the file, the server looks in the following locations, in the order listed below.

1. The active theme folder: /Themes/<active-theme>
2. The default theme stored in /Themes/default.

When one of the CSS files references an image file or a helper file, including any path to that file, the server looks for that path and filename in the same two locations, in the same order. In this way, each file and image is resolved first in the active theme, and if not found, then in the default theme.

The active theme does not need to contain all the files because the default theme is guaranteed to contain all the files.

6.2.3 CSS Priority Scheme and Custom Overrides

Once inheritance determines which files to load, the standard CSS priority scheme determines which rules are visible, based on the order in which files are loaded.

This leads to two general ways of developing custom themes:

- The quickest way is to copy individual CSS rules from the default theme files, modify the rules to change the UI, and save them in the overrides_custom.css file. This is the only file in your new theme. Because overrides_custom.css is always the last CSS file to be loaded, its rules override the same rules in other files. This allows you to easily change any number of rules, and manage them all in a single file.
  For example, if you want to increase the size of text on all the buttons in the default theme, you can do this with a few rules in the overrides_custom.css file. You may need to adjust the spacing for certain buttons, but the idea is you only need to change a limited number of rules.
- If you modify the user interface extensively, you can use the existing structure of CSS files in the default theme. In this case, copy the relevant files from the default theme, make your modifications, and save the files in your new theme. The new files are inherited when you activate the theme.

An example of these extensive changes would be if you want to increase the size of the buttons themselves in the default theme. You would need to rewrite the majority of the rules in the buttons.css file and create images for the new buttons. In this case, it is much easier to copy the buttons.css file than to copy dozens of rules into the overrides_custom.css file. You could still use the overrides_custom.css file to adjust the spacing of elements around the buttons, because there would be fewer of those rules to modify.

We recommend using the custom overrides method for most custom themes. A custom theme that changes simple appearances such as colors, fonts, and spacing has relatively few rules and is easily manageable in a single file. And many changes can be made by copying and modifying image files in the custom theme, without writing any CSS rules. Only if you change the fundamental layout or appearance of the user interface, should you consider copying and modifying the other CSS file.

Copying and modifying CSS files is more prone to error, and is slightly less flexible due to file-based inheritance. Your copy of the file must contain all of the CSS rules as the original. If any rules are accidentally deleted or modified, even by a single character, the theme may not work properly.

6.3   Administering Themes

Themes are sets of CSS and image files stored in a folder in the repository. The root of the repository has a Themes folder where default and active themes are stored. In the repository browser, the Themes folder and individual theme folders have special actions for administrators to manage them. You can also use the repository search to find CSS and image files.

The folders and actions for managing themes are visible only to administrators. The Themes folder has execute-only permission for ROLE_USER so that all users can load the theme files and see the user interface, but not access the folders and files in the repository.

This section gives the basic procedures for administering existing themes. To create theme folders and files, see **6.4, "Creating Themes," on page 103**. For information about how to work with CSS in themes, see **6.5, "Working With CSS Files," on page 105**.

> The Easy Access theme is specifically intended to improve the web UI's accessibility. It increases color contrast and highlighting in the web UI. It can improve the user experience of those with visual impairment.

This procedure assumes you have already created and uploaded your theme:

1. Log into JasperReports Server as administrator (`jasperadmin`).
2. Click **View > Repository** and expand the root/Themes folder if necessary.
3. Right-click the chosen theme folder and select **Set as Active Theme**.

   For example, the sample data includes a theme called jasper_dark that you can set as active.
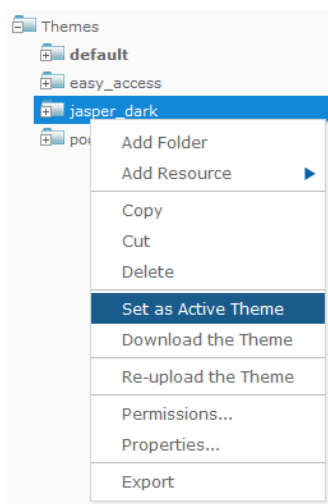
Figure 6-2 Setting a Theme

As soon as the screen is refreshed, you see the effect of the new theme. Notice how the jasper_dark theme changes the colors in the user interface with just the overrides_custom.css file and some image files.
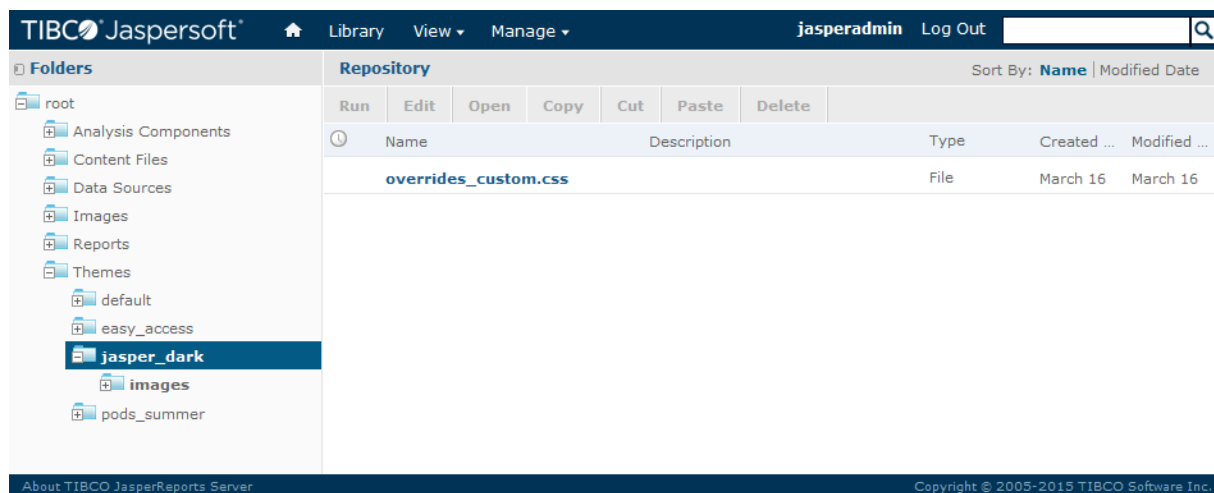
Figure 6-3 The Sample Theme jasper_dark

When you change the theme, it applies immediately to all users on the server. Also, the new theme applies to the login page, as shown in the following figure.

Figure 6-4 **The Login Page as Seen With a New Theme**

6.4   Creating Themes

There are two ways to create the folders and files that make up a theme:

- **Creating Theme Folders and File Resources** – Create them individually as resources in the repository.
- **Downloading and Uploading Theme ZIP Files** – Copy and modify existing themes as ZIP (archive) files.

This section explains only how to store CSS files in the repository. For information about creating CSS file contents, see **6.5, "Working With CSS Files," on page 105**.

6.4.1   Creating Theme Folders and File Resources

A theme is simply a folder in the repository that contains CSS and image files, with optional sub-folders. Administrators can use the repository menus to create theme folders.

To create theme folders and file resources:

1. Log in as an administrator.
2. Click **View > Repository** and expand the folder tree to view the Themes folder at the root.
3. Right-click the Themes folder and select **Add Folder**. Give your folder a name and optional description as you would when creating any folder. The folder name is used as the name of the theme.

Theme folders and files can be created, copied or moved anywhere in the repository, but they can only be made active, uploaded, or downloaded when properly placed in the Themes folder.

4. Right-click your new folder and select **Add Resource > File > CSS**, and use the dialog to upload an individual CSS file. In order to be used as part of a theme, it must be one of the file names shown in **Figure 6-1**.

5. To add images to your theme, create any image folders and upload image files with **Add Resource > File > Image**.

6. Repeat **step 4** and **step 5** to create all the files and images you need. If several themes use the same files or images, you can copy-paste the file resources or entire image folders from one theme to another.

7. If you need to change the contents of a CSS or image file, you can right-click it and select **Edit** to specify another file to upload and replace the current file.

If you upload CSS and image files into the active theme, the changes are visible after reloading the page in your browser.

Interacting with theme folders and files through the repository is a convenient and flexible way to create a theme. However, this method suffers from the limitation that, like other repository resources, you cannot download the files or images to edit them. For this purpose, the repository provides special download and upload actions on theme folders.

6.4.2   Downloading and Uploading Theme ZIP Files

The process of creating a theme often starts with the files of an existing theme that you modify with CSS and image editors on your computer. To support this workflow, the Themes folder has special commands for downloading and uploading themes.

Because a theme is composed of any number of files and folders, JasperReports Server uses the ZIP archive format to store a theme in a single file.

To download a theme ZIP file:

1. Log in as an administrator.

2. Click **View > Repository** and expand the Themes folder if necessary.

3. Right-click the theme folder you want to download and select **Download the Theme**. This menu selection appears only on theme folders inside the Themes folder.

4. The server prompts you to save the file named <theme-name>.zip. Save it anywhere on your computer.

5. Use an archiving or compression utility to extract the files from the ZIP file and save them on your computer.

Once you have the theme files extracted on your computer, you can view the individual CSS and image files that make up the theme. For example, to create your own theme, start by downloading the default theme from the root/Themes folder. Save the extracted file on your computer and create your custom theme in another folder by copying and editing the CSS files and images of the default theme. See **6.2.3, "CSS Priority Scheme and Custom Overrides," on page 100** for an explanation of how to edit theme files.

When you have created all the files you need in your theme, upload it with the following procedure.

To upload a ZIP file as a theme:

1. Place the CSS files, optional folders, and images files that constitute your theme in a folder on your computer.

2. Use an archiving or compression utility to create a standard ZIP file of the contents of your theme folder.

> The ZIP file should include only the contents of your theme, not the theme folder itself.

3. Log in as an administrator.
4. Click **View > Repository** and expand the Themes folder if necessary.
5. Right-click the Themes folder and select **Upload a Theme**.
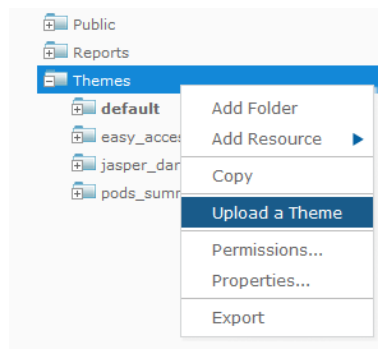
Figure 6-5 Uploading a Theme ZIP File

6. In the dialog that appears, enter a name for your theme, and browse to find the ZIP file on your computer. Click **Upload**. The theme name becomes the name of the theme folder.

> You cannot use the ZIP upload dialog to overwrite an existing theme. You must specify a theme name that doesn't already exist in the Themes folder.

The server uploads your ZIP file and extracts it contents. Then it creates a folder for the new theme and creates file resources in the folder for each of the CSS and images in your ZIP file. If you had sub-folders in your theme, they are created as well. After uploading your theme ZIP file, you can make it active to see effect of your theme on the user interface.

Creating a theme is an interactive process where you often need to make changes until you have the look and feel you want. To support this process, uploading ZIP files can be combined with the uploading of individual file resources that is described in **6.4.1, "Creating Theme Folders and File Resources," on page 103**. In fact, after an initial upload, it is much easier to update individual files in this way than to create the ZIP file and upload it again.

6.5   Working With CSS Files

This section is not a CSS tutorial but rather a collection of tips and tricks for working with the CSS that makes up the themes in JasperReports Server. This section focuses on how to test the themes you develop and match the CSS to its behavior in the JasperReports Server UI. There are many different editors for CSS and tools for testing it, so the recommendations in this section are just one way of developing a theme.

6.5.1 Theme Development Workflow

The major choice to make when developing a theme is whether to use simple theme overrides or to duplicate and modify theme files, as described in section **6.2.3, "CSS Priority Scheme and Custom Overrides," on page 100**. Usually, the extent of your modifications determines which method to use.

Once you have made that determination, you are ready to create your theme. The main steps in a theme development workflow are as follows:

Step	Reference
1. Download the default theme so you have a copy of the files and CSS rules that you want to modify.	**6.4.2, "Downloading and Uploading Theme ZIP Files," on page 104**
2. Create your new CSS rules, CSS files, and image files.	**6.5.2, "Firefox Web Developer Tools," on page 106**
3. Upload your new files to a test platform, and activate the theme or place them in an active theme.	**6.4.1, "Creating Theme Folders and File Resources," on page 103**
4. Verify your changes wherever they occur in the UI.	**6.5.3, "Test Platform," on page 107** and **6.5.5, "User Interface Samples," on page 107**
5. Repeat **step 2** through **step 4** for all your changes until the theme is finalized.	
6. Deploy your theme to your users.	**6.3, "Administering Themes," on page 101**

6.5.2 Firefox Web Developer Tools

To help you find, view, and modify CSS rules in **step 2** above, you can use the web developer tools in the Mozilla Firefox browser. In particular, the Page Inspector tool displays the HTML and CSS rules of web pages as you browse. It has a dynamic interface that lets you select an element on the web page, and it displays the specific CSS rules that apply to the element. It also allows you to modify those rules in the browser memory and immediately see the effect on the web page.

The web developer tools are ideal for modifying themes in JasperReports Server. Once you locate the pages and elements that you want to modify, you can prototype your changes directly within the tool. For example, you can see overall effect of changing a color or modifying the spacing.

If you are implementing your theme through custom overrides, you can copy the CSS rules from the tools directly into the overrides_custom.css file. Firefox displays the entire rule from its original file, so the copy overrides it exactly. If you are modifying other files from the default theme, the Page Inspector shows you the filename and line number of the rule, so that you can easily find it in your copy of the file.

And when you are testing a theme that uses overrides, the Page Inspector displays both the active CSS rule from overrides_custom.css and the original rule in the regular theme file of the inherited theme. The original rule is displayed in strike-through, so you can easily tell which rule is active and which rule it overrides, according to the CSS priority scheme.

For more information, see the Firefox Page Inspector tool overview and documentation.

6.5.3 Test Platform

When you upload a theme and make it active, it is immediately visible to every user on the server. Even editing or uploading a file into an active theme is reflected immediately in the user interface. Because developing a theme requires many iterations of uploading, activating, and testing CSS rules, you shouldn't develop themes on a production server.

In the simplest case, you can develop and test your themes before putting your JasperReports Server into production. As you test your server during the deployment, you can develop your themes without impacting real users. Alternatively, you can test your themes on a second installation of the server, locally on the same computer where you develop the theme.

When your theme is well-tested and nearly complete, you should test it on the production server. Upload your theme to the Themes folder, but do not activate it. Log in as a test user and add the following parameter to any URL, for example the home page URL:

```
&theme=<theme-name>
```

This activates your theme for the test user on all pages that you access until the user session times out. This allows you to navigate the entire application and see the effect of your theme in the production environment, without affecting other users.

> To set the theme back to the default append the &theme parameter to the URL with the string default (`&theme=default`). This is especially useful if a problem with the current theme has inadvertently disabled any functionality.

On all of these test platforms, you should look at the user interface generated by your theme with the same browsers and browser versions that your users have. If you see errors, you can still use the Firefox web developer tools to look at the CSS rules that are involved, even if the errors do not show up in Firefox.

6.5.4 Modifying the Appearance of Jaspersoft OLAP

Jaspersoft OLAP relies on a module called jPivot to display data when performing OLAP analysis. The jPivot module does not use all of the features of the new UI framework, but it supports some customizations through themes. For example, it does not use panels that can be hidden, and images for icons are not stored in a theme. However, some display characteristics of the analysis table are controlled by the theme, through the use of the `analysisView` ID in the theme file dataDisplays.css.

For example, you can change the lines between cells in the analysis table with the following rule in your overrides_custom.css file:

```
#analysisView td { border: thin solid black; }
```

6.5.5 User Interface Samples

When testing your theme, you should look at its effect across all pages and dialogs of JasperReports Server. Your test users should access all the features of the server to view the user interface under all conditions. An additional test is to look at the user interface samples with your theme.

The user interface sample page is accessible only to administrators:

1. Log in as administrator in your test environment.
2. If you haven't already done so, upload your theme to the Themes folder.
3. Select **View > UI Samples** from the main menu on any page.

4. Look at all the sample components in each of the sample galleries. For example, the buttons gallery shows all the different types of buttons in every possible action state.
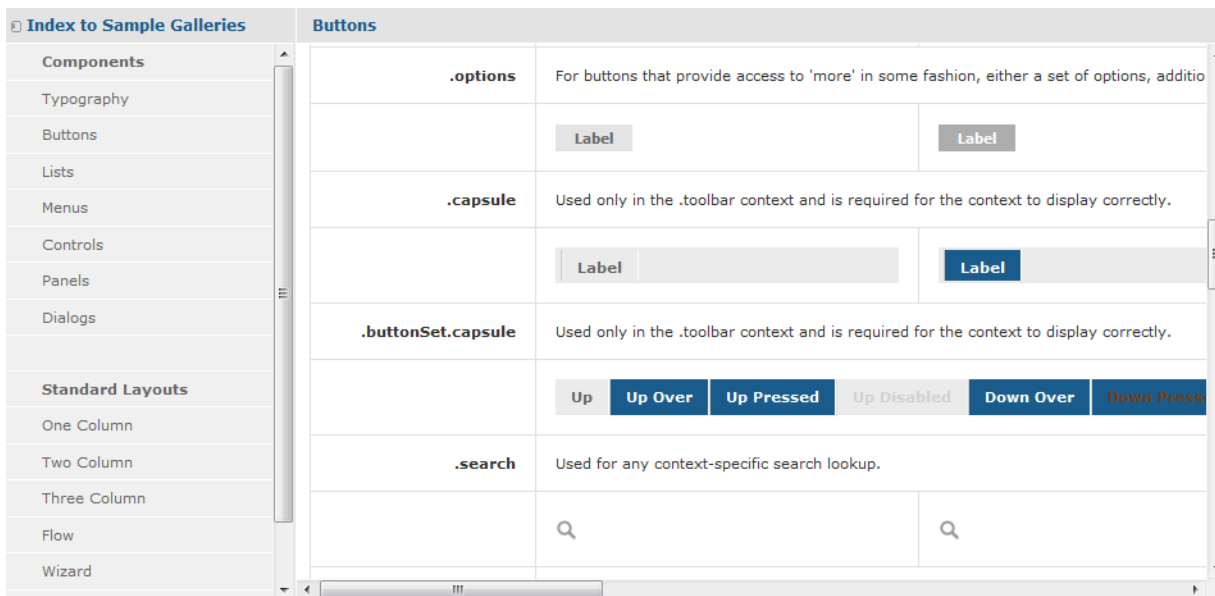
Figure 6-6 **Button Components in the Sample Galleries**

5. When you click on the standard layouts, the sample replaces the samples page. Select **View > UI Samples** from the main menu again to return to the galleries.

> The samples page relies on an extra CSS file that is not required in a theme, but that can be included. The file samples.css is located in the default theme in the system-level Themes folder. If the sample elements do not appear as you expect, add this file to your theme and customize its rules as necessary. The rules in this file are not used anywhere else in the user interface, so it should not be included in your final theme.

Viewing the sample galleries can help you quickly find errors in your theme, especially if you are changing many rules and replacing entire files in your theme. Using these samples along with the testing procedures and tools described previously, you can verify that your theme properly implements the custom user interface that you intend. Having a well-tested theme minimizes the chances of errors when you activate the theme in your production server.

CHAPTER 7   IMPORT AND EXPORT

The import and export tools enable you to move resources into and out of the JasperReports Server repository. These tools can back up the entire contents of repository selected folders, or specific resources. These utilities also handle scheduled jobs, users, and roles that the server stores internally. Import and export can be helpful when migrating between versions of JasperReports Server or when moving between test and production environments.

JasperReports Server provides both a user interface and command-line utilities to perform import and export. The functionality in the UI is available only to administrators (`jasperadmin`) and the command-line utilities require access to the file system where the server is installed.

This chapter contains the following sections:

- **Import and Export Catalogs**
- **Import and Export Through the Web UI**
- **Import and Export Through the Command Line**
- **Alternate Import-Export Scripts**

7.1   Import and Export Catalogs

The output of the export command and the input to the import command is called a catalog. It's a set of the folders and files that comprise the server's internal database, including users, roles, scheduled jobs, and repository resources such as reports and associated files. When you don't need the entire database, you can specify options to export only the contents you need, for example one role and its users.

The catalog can be exported either as a hierarchy of folders and files, or as a single zip file (compressed archive). Regardless of the catalog format, the content of the catalog is not intended for external access. Objects in the database, such as users, roles, and folders are described in XML files, and repository resources are stored in various private formats consisting of data files and subfolders. The XML syntax of the catalog files is not publicly defined, and the data files aren't meant to be accessed.

To access and interact with the server's internal objects, use the REST v2 API. This web service has well defined datatypes and resource descriptors in XML or JSON formats and a complete set of methods for reading and writing objects on the server. For more information, see the *JasperReports Server REST API Reference*.

As of JasperReports Server 5.5, user passwords and data source passwords are encrypted in exported catalogs as well as in the server's internal database. You should still take appropriate measures to secure the catalog file from unauthorized access. Catalog files may contain sensitive metadata such as user names, database URLs, and customer information. Catalog files may also contain data in the form of report output such as the PDF of an executive report.

7.1.1 Dependencies During Import and Export

The resources in the repository often have dependencies on other resources, for example a report that relies on images, input datatypes, and a data source. When exporting a resource, you have the option of including all of its dependencies, even if they are stored in folders that were not selected in the repository. Importing a catalog that contains such dependencies will re-create the same folder structure in the target repository. Once imported, you can move and redefine the dependencies of these resources.

7.1.2 Setting the Import-Export Encryption Key

As of JasperReports Server 5.5, all sensitive passwords in exported catalogs are encrypted for security. In order for two servers to share the encrypted contents of an exported catalog, they must share a private key. The default key is an AES 128-bit string stored in a configuration file.

This encryption is separate from the server's own internal encryption. All user passwords are stored encrypted in the internal database, as described in the *JasperReports Server Security Guide*. The import-export encryption applies only to export catalogs.

Setting the Import-Export Encryption Key		
Configuration File		
.../WEB-INF/applicationContext-security.xml		
Property	**Bean**	**Description**

`<property name="keyBytes">`
`<value>0x2b 0x6c 0x34 0x22`
`0x44 0x42 0x6f 0xb5 0x7f`
`0x34 0xd3 0x5a 0x1f 0x92`
`0xcd 0xdc</value>`
`</property>`
	`importExport`
	`Cipher`
		Set the value of the `keyBytes` property to the same hexadecimal value (16 bytes = 128 bits) on all servers that will exchange export catalogs.

When you change a private key on a server, all previous exports become unusable. Therefore, you must configure your new server soon after installing it, and you should configure it with the key from an existing server, if you have one. This way all your servers and all your export catalogs will use the same key and be mutually compatible.

7.1.3   Importing Unencrypted Catalogs

Versions before 5.5 did not encrypt the user passwords upon export. These passwords appear in plain text within the files of the export catalog. For backwards compatibility, unencrypted catalogs are still supported and can be imported into a newer server, even when an encryption key is configured.

Passwords that are encrypted by an export operation in JasperReports Server 5.5 or later have encryption markers, so older passwords without the markers can be detected and imported as plain text. Once plain-text passwords are imported, they are stored internally with encryption, and will be encrypted in any future export.

7.2   Import and Export Through the Web UI

JasperReports Server allows administrators to import and export resources and users through the user interface:

Section	Actions
Exporting From the Repository	Export entire folders Export selected resources
Exporting From the Settings	Export everything Export selected users or roles Export resources by type
Importing From the Settings	Import any catalog into the server

7.2.1   Exporting From the Repository

> If you're exporting to a different server, you must configure an encryption key on both servers, as described in **7.1.2, "Setting the Import-Export Encryption Key," on page 110**.
>
> Catalogs may be very large and take a long time to generate and then download. During this time, the export operation may affect server performance.

To export resources (non-organization) from the repository:

1. Log in as an administrator that has access to the resources you want to export.
2. Select **View > Repository**.
3. Select one or more non-Organization resources in the main panel, or when viewing the repository, select a folder in the left-panel.
4. Right-click the selected folder or resources and select **Export** from the context menu. The Export Resources dialog appears:
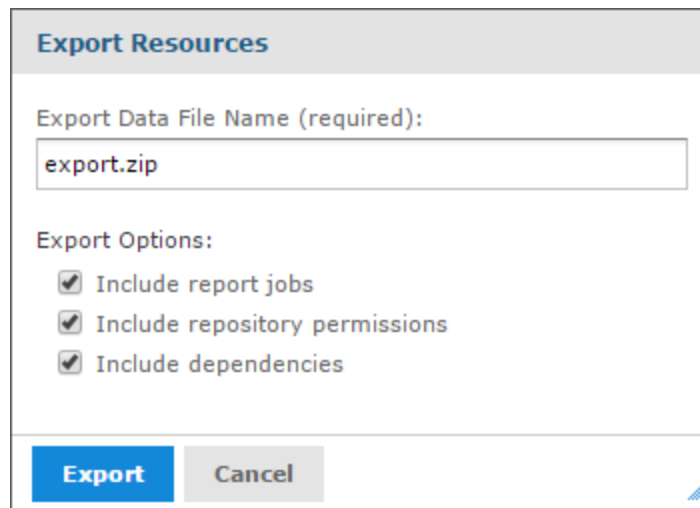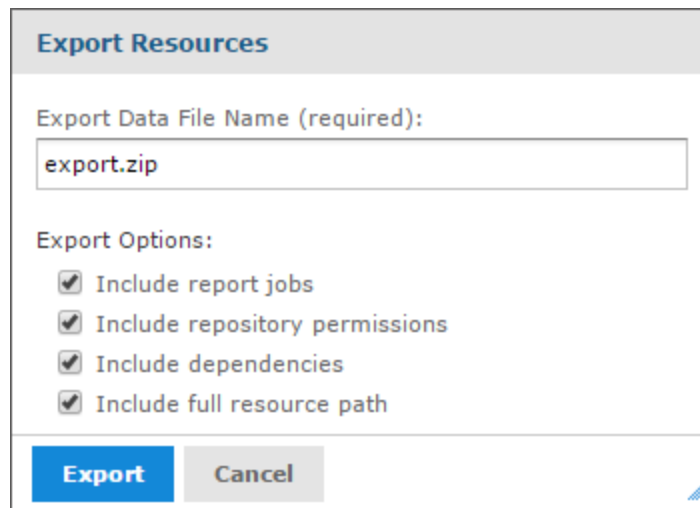
Figure 7-1 Export Resources Dialog in the Repository

5. If desired, change the default name of the zip file for the exported catalog. This dialog allows only the zip archive format.

6. Choose one or both of the export options:

   • **Include report jobs** – When checked, the export includes scheduled report jobs with any reports in your repository selection.

   • **Include repository permissions** – When checked, the export includes any explicit permissions on all items in your repository selection. Clear this check box if you want the exported items to inherit the permissions of the destination repository.

   • **Include dependencies** – When checked, the export includes all dependencies for your resources, even if they are not included in your selected folders. For more information, see **7.1.1, "Dependencies During Import and Export," on page 110**.

7. Click **Export**. The server generates the catalog zip file and your browser prompts you to save the file. Depending on the size of your repository and the options you've selected, it may take several minutes to generate the catalog file.

To export organization resources from the repository:

1. Log in as an administrator that has access to the organization resources you want to export.

2. Select **View > Repository**.

3. Select a folder under **Organization** in the left-panel or one or more Organization resources in the main panel.

4. Right-click the selected folder or resources and select **Export** from the context menu. The Export Resources dialog appears:

Figure 7-2 **Export Resources Dialog in the Repository**

5. If desired, change the default name of the zip file for the exported catalog. This dialog allows only the zip archive format.

6. Choose one or both of the export options:
    - **Include report jobs** – When checked, the export includes scheduled report jobs with any reports in your repository selection.
    - **Include repository permissions** – When checked, the export includes any explicit permissions on all items in your repository selection. Clear this check box if you want the exported items to inherit the permissions of the destination repository.
    - **Include dependencies** – When checked, the export includes all dependencies for your resources, even if they are not included in your selected folders. For more information, see **7.1.1, "Dependencies During Import and Export," on page 110**.
    - **Include full resource path** – When checked, the export includes the full file structure, starting with the Organizations root folder, for the resource in the ZIP file. If you want to import the resource into another organization, uncheck this box and the export includes only the folders within the parent organization's branch in the ZIP file.

7. Click **Export**. The server generates the catalog zip file and your browser prompts you to save the file. Depending on the size of your repository and the options you've selected, it may take several minutes to generate the catalog file.

7.2.2   Exporting From the Settings

For more export options, use the server settings page. The settings page lets you export the following resources:
- Everything – The entire repository, including all resources, as well as all users, roles, and other settings stored internally. With the proper options, this creates a backup of the server.
- Any combination of users and roles – Lets you choose from lists of users and roles, with options to include users by role or roles by user.
- All resources of a given type – For example all reports or all dashboards.

> If you're exporting to a different server, you must configure an encryption key on both servers, as described in **7.1.2, "Setting the Import-Export Encryption Key," on page 110**.

To export resources from the settings page:

1. Log in as administrator (`jasperadmin` by default).
2. Select **Manage > Server Settings**, then click **Export** in the left-hand panel.

Figure 7-3 **User Interface for Export**

3. If desired, change the default name of the zip file for the exported catalog. This dialog allows only the zip archive format.
4. Use the check boxes and radio buttons to choose the contents of your exported catalog file.

   Select **Export Everything** (default) to export the entire repository, including all users and roles, and all types of assets. Select the check boxes under Events to Export to include the different types of events in your export catalog.

5. Clear **Export Everything** to select users and roles or resource types to export.
   a. To export users and roles, choose one of the radio buttons, then select individual users and roles from the lists.
      • **Selected roles and users** – Only the roles and users you select explicitly are exported.
      • **Users with selected roles** – Select one or more roles, and all users with those roles are exported, along with the selected roles.
      • **Roles with selected users** – Select one or more users, and all roles assigned to those users are exported, along with the selected users.
   b. If you only want users and roles, clear all check boxes under Resources to Export.

c.   Or if you only want resources, do not select any users and roles, then select the resource types you want to export.

d.   Select the check boxes under Assets to Export to include these various assets in your export catalog.

e.   Select the check boxes under Events to Export to include the different types of events in your export catalog.

6.   Click **Export**. The server generates the catalog zip file and your browser prompts you to save the file. Depending on the size of your repository and the options you've selected, it may take several minutes to generate the catalog file.

> Resources are exported along with any dependencies, even if they're not included in your repository selection. For more information, see **7.1.1, "Dependencies During Import and Export," on page 110**.
>
> Catalogs may be very large and take a long time to generate and then download. During this time, the export operation may affect server performance.

7.2.3   Importing From the Settings

The Settings pages for administrators include a user interface to simplify the import procedure.

This import operates on a running instance of the server, and all imported resources are visible immediately. In addition, any configuration or security settings in the imported catalog take effect immediately, with no need to restart the server.

> If you're importing from a different server, you must configure an encryption key on both servers, as described in **7.1.2, "Setting the Import-Export Encryption Key," on page 110**. . You'll need to enter the keystore password when prompted by the import operation.

To import data from the Settings page:

1.   Log in as administrator (`jasperadmin` by default).

2.   Select **Manage > Server Settings** and choose **Import** in the left-hand panel.

Figure 7-4 User Interface for Import

3. Click **Browse** to choose the catalog zip file to import. The dialog allows only the zip archive format.

4. Use the check boxes to change the behavior of the import operation:

   • When checked, the **Update** option will import only resources that are newer than ones with the same URI in the current repository. The **Skip user updates** option allows you to keep the current definition of any users that also exist in the imported catalog.

   • When checked, the **Include access events** option imports the modification times of resources from the catalog. When cleared, resources keep their existing access times if they already exist.

   • The **Include server settings** option determines whether the system configuration is updated from the catalog. There are two prerequisites in order for the catalog to contain configuration settings:

      • The originating server settings must be modified through the UI. Thus, only Log Settings, OLAP Settings, and Cloud Settings are affected. For more information, see **8.1, "Configuration Settings in the User Interface," on page 126**

      • The catalog must be exported with the "everything" option or the specific **Server Settings** option.

   When server settings are imported, they take effect immediately and appear in the Settings UI.

5. Click **Import**.

   The server uploads the catalog zip file and imports its contents into the repository. Depending on the size of the catalog and the options you've selected, it may take several minutes to perform the import.

   Resources are imported along with any dependencies, unless you do not have permission to write at the dependency's location. Broken dependencies may block the import operation. For more information, see **7.1.1, "Dependencies During Import and Export," on page 110**.

   Catalogs may be very large and take a long time to upload and then process. During this time, the import operation may affect server performance.

7.3   Import and Export Through the Command Line

> If you installed JasperReports Server from the binary installer, the command-line utilities were configured by the installer. If you installed the WAR file distribution, you must follow the instructions in **7.3.3, "Configuring Import-Export Utilities," on page 121** before you can run the utilities.

The import and export utilities are shell scripts located in the <js-install>/buildomatic folder:

Windows: <js-install>/buildomatic/js-import.bat
                   <js-install>/buildomatic/js-export.bat

Linux: <js-install>/buildomatic/js-import.sh
                   <js-install>/buildomatic/js-export.sh

The examples in this chapter use the shortened Windows commands without the optional .bat extension on the command line. If you're running JasperReports Server on Linux, be sure to add the `.sh` file extension.

When using the import and export utilities, keep the following in mind:

• JasperReports Server should be stopped when using the import and export utilities. This is very important for the import utility to avoid issues with caches, configuration, and security.
• All command line options start with two dashes (--).
• You must specify either a directory or a zip file to export to or import from.
• If you're importing to a different server, configure an encryption key on both servers, as described in **7.1.2, "Setting the Import-Export Encryption Key," on page 110**. Then enter the keystore password when prompted by the import command.
• Make sure the output location specified for an export is writable to the user running the command.
• All URIs are repository paths originating at the root (/).

> The import and export scripts provide access to the repository and internal database of the server. Even though all passwords are encrypted during export, a catalog may still contain sensitive URLs and data. You should set permissions on the host file system and operating system to secure the scripts and any catalogs you export.

7.3.1   Exporting From the Command Line

Usage: `js-export [OPTIONS]`

> We recommend you stop your server instance before running the export utility. For instructions see the *JasperReports Server Community Project Installation Guide*.

Specifies repository resources such as reports, images, folders, and scheduled jobs to export. You can also export the internal definitions for scheduled jobs, users, roles, and audit data. The export output is known as a repository catalog. It's either a zip archive file or a set of files in a folder structure.

Table 7-1 Options in `js-export` Command

Option	Explanation
`--everything`	Exports everything: all repository resources, permissions, report jobs, users, and roles. If any server settings have been modified in the UI, those are also included. This option is equivalent to: `--uris --repository-permissions --report-jobs --calendars --users --roles`
`--help`	Displays brief information about the available options.
`--include-access-events`	Exports repository events (date, time, and user name of last modification).
`--output-dir`	Path of a location to export the catalog in a folder structure.
`--output-zip`	Path and filename to export the catalog as a zip file.
`--report-jobs`	Comma separated list of repository report unit and folder URIs for which report unit jobs should be exported. For a folder URI, this option exports the scheduled jobs of all reports in the folder and all subfolders.
`--calendars`	When specified, the export includes any and all calendars of all types (holiday, recurring, ...) defined in the scheduler. When calendars are present in an export catalog, they're always processed and added upon import.
`--uris`	Comma separated list of folder or resource URIs to export from the repository. If the URI specifies a folder, the export operation exports all resources and folders contained in the folder. In addition, it recurses through all its subfolders.
`--resource-types`	Comma separated list of resource types to export. The available resource types are: adhocDataView, awsDataSource, beanDataSource, customDataSource, dashboard, dataType, file, folder, inputControl, jdbcDataSource, jndiJdbcDataSource, listOfValues, mondrianConnection, mondrianXmlaDefinition, olapUnit, query, reportUnit, virtualDataSource, xmlaConnection.
`--repository-permissions`	This option exports repository permissions with each exported resource or folder. This option should only be used in conjunction with `--uris`.
`--skip-dependent-resources`	Exports resources without their dependencies, for example a report without its external data source, input control definitions, or image files.
`--roles`	Comma separated list of roles to export. If no roles are specified with this option, all roles are exported.

Option	Explanation
`--role-users`	Use only with `--roles`. This option exports all users belonging to each exported role.
`--users`	Comma separated list of users to export; if no users are specified with this options, all users are exported. Exporting a user includes all user attributes and all roles assigned to each user.
`--users-roles`	Use only with `--users`. This option exports all roles belonging to each exported user.
`--include-attributes`	Specify this flag to export attributes on any user or root level that is exported.
`--skip-attribute-values`	When used with `--include-attributes`, specifies that only attribute names are exported, values will be null.

> User passwords are encrypted during the export by default, but exported catalogs may contain sensitive data. Take appropriate measures to secure the catalog file from unauthorized access.

Examples:

- Export everything in the repository:

      js-export --everything --output-dir myExport

- Export the /reports/interactive/CustomersReport report unit to a catalog folder:

      js-export --uris /reports/interactive/CustomersReport --output-dir myExport

- Export the /images and /reports folders:

      js-export --uris /images /reports --output-dir myExport

- Export all resources (except users, roles, and job schedules) and their permissions to a zip catalog:

      js-export --uris / --repository-permissions --output-zip myExport.zip

- Export all resources and report jobs:

      js-export --uris / --report-jobs / --output-dir myExport

- Export the report jobs of the /reports/interactive/CustomersReport report unit:

      js-export --report-jobs /reports/interactive/CustomersReport --output-dir myExport

- Export all roles and users:

      js-export --roles --users --output-dir myExport

- Export the ROLE_USER and ROLE_ADMINISTRATOR roles along with all users belonging to either role:

      js-export --roles ROLE_USER, ROLE_ADMINISTRATOR --role-users --output-dir myExport

7.3.2   Importing From the Command Line

See **7.3, "Import and Export Through the Command Line," on page 117** for guidelines when running the command-line utilities.

⚠ When using the js-import command line utility, the server must be stopped to avoid issues with caches, configuration, and security. For instructions see the *JasperReports Server Community Project Installation Guide*.

Usage: `js-import [OPTIONS]`

Reads a repository catalog from your file system and creates the resources in the JasperReports Server repository. The repository catalog must be one created by the export interface or the `js-export` command, either as a ZIP archive file or a folder structure.

As of JasperReports Server 5.5, all exports contain encrypted passwords and if you're importing to a different server, configure an encryption key on both servers. See **7.1.2, "Setting the Import-Export Encryption Key," on page 110** for details.

Table 7-2 Options in `js-import` Command

Option	Explanation
`--help`	Displays brief information about the available options.
`--input-dir`	Path for importing a catalog from a directory.
`--input-zip`	Path and filename for importing a catalog from a zip file.
`--update`	Resources in the catalog replace those in the repository if their URIs and types match.
`--skip-user-update`	When used with `--update`, users in the catalog are not imported or updated. Use this option to import catalogs without overwriting currently defined users.
`--brokenDependencies`	Specifies the action to take when importing a resource with a broken dependency. One of the following values: • `skip` – Does not import the resource with the broken dependency, but continues to import other resources. • `include` – Attempts to import the resource with the broken dependency. The import succeeds if there is already a resource in the destination that satisfies the dependency. If the dependency is not satisfied in the destination, the resource is skipped and the import continues. • `cancel` – Stops the import operation.
`--include-access-events`	Restores access events (date, time, and username of last modification) on imported resources.

Option	Explanation
`--include-server-settings`	Determines whether the system configuration is updated from the catalog. There are two prerequisites for the catalog to contain configuration settings: • The originating server settings must be modified through the UI (Log Settings, Ad Hoc Settings, Ad Hoc Cache, and OLAP Settings). For more information, see **8.1, "Configuration Settings in the User Interface," on page 126**. • The catalog must be exported with the "everything" option from the user interface or the command-line utility. Imported server settings take effect when the server is started.
`--skip-themes`	This flag is required when importing a catalog that includes a theme from a server version 5.2 or before to version 5.5 or later. If you have a custom theme to import, you can use the Theme UI to download it from the source server and upload it to the target server. If your theme contains the file pageSpecific.css, you must remove it from the ZIP file before uploading, and then redo your changes to the file based on pageSpecific.css in the target server from 5.5 or later. For more information, see **6.4.2, "Downloading and Uploading Theme ZIP Files," on page 104**.

Examples:

- Import the myExport.zip catalog archive file:

  ```
  js-import --input-zip myExport.zip
  ```

- Import the myDir catalog folder, replacing existing resources if their URIs and types match those found in the catalog:

  ```
  js-import --input-dir myDir --update
  ```

- Import the myExport.zip catalog archive file but ignore any users found in the catalog:

  ```
  js-import --input-zip myExport.zip --update --skip-user-update
  ```

- Import the myDir catalog folder with access events:

  ```
  js-import --input-dir myDir --include-access-events
  ```

When a resource in the target repository has the same URI as on that you're importing, the default behavior is leave the existing resource unchanged (no overwriting occurs).

To delete the existing resource and replace it with a new one (of the same type and with the same URI), use the `--update` option. Note that, if the resource in the export catalog is a different type than the existing resource, the server returns an error and skips the update operation.

When you import a user whose roles exist in the repository, the user is given those roles. User properties are imported with the user.

When you import access events, the date and time of the last modification before export is restored on import for every resource. The catalog folder has to be created with access events. If you don't import access events, or if they don't exist in the imported files, the date and time of the import are used.

7.3.3 Configuring Import-Export Utilities

If you installed JasperReports Server from the binary installer, the import-export utilities were configured by the installer. If you installed the WAR file distribution, you must configure several files before you can use the import-export utilities.

Alternatively, see **7.4, "Alternate Import-Export Scripts," on page 122** because the alternate scripts don't require any configuration, regardless of the installation method.

To configure the import-export utilities:

1. Depending on the database you use, copy the installation configuration file:

    from: <js-install>/buildomatic/sample_conf/<database>_master.properties

    to: <js-install>/buildomatic/default_master.properties

2. Edit the default_master.properties file to set values specific to your installation. For more information about the settings in this file, see the *JasperReports Server Community Project Installation Guide*.

> Oracle users can set the `sysUsername` and `sysPassword` to the same name as `dbUsername` and dbPassword in the default_master.properties. The system user name and password are not required because js-import and js-export do not make changes to the database schema.

3. Run the following command:

    ```
    js-ant clean-config gen-config
    ```

    This command will generate the following files with the values you added to the default_master.properties file:

    • <js-install>/buildomatic/build_conf/default/js.jdbc.properties
    • <js-install>/buildomatic/build_conf/default/js.quartz.properties (only for DB2 and PostgreSQL)

4. Make sure the JDBC driver for your database is located in the following folder:

    <js-install>buildomatic/conf_source/iePro/lib

    If necessary, you can find links for downloading JDBC drivers from the Jaspersoft Community website:

    http://community.jaspersoft.com/wiki/downloading-and-installing-database-drivers

7.4 Alternate Import-Export Scripts

Regardless of your installation method, JasperReports Server provides a third way to run import-export commands. Buildomatic is another command-line script that is based on the Apache Ant tool to automate installations. It includes targets (sub-commands) to perform import and export operations with the same options as the scripts. The following examples compare the two commands:

Shell Scripts:	`js-export.sh --everything --output-zip=js-catalog-exp.zip`
Buildomatic:	`js-ant export-everything -DexportFile=js-catalog-exp.zip`

Both types of scripts are located in the `<js-install>/buildomatic` folder.

7.4.1 Running Import from Buildomatic

The `import` target for ant has the following syntax:

Windows:	`js-ant import -DimportFile=<filename> [-DimportArgs="<import-options>"]`
Linux and Mac OSX:	`./js-ant import -DimportFile=<filename> [-DimportArgs=\"<import-options>\"]`

The imported file is handled as a ZIP archive if its name ends in .zip, otherwise it's handled as a directory. The `importArgs` argument is optional and can contain more than one import option. On Linux, all double quotation marks (`"`) must be escaped with a backslash (\).

> When performing a large import using js-ant, the server should be stopped (or put into a mode with reduced load) to avoid issues with caches, configuration, and security.

The following examples are typical import commands on Windows:

```
js-ant import-help-pro
js-ant import -DimportFile=my-reports.zip
js-ant import -DimportFile=my-datasources -DimportArgs="--update"
```

The following examples are typical import commands on Linux:

```
./js-ant import-help-pro
./js-ant import -DimportFile=my-reports.zip
./js-ant import -DimportFile=my-datasources.zip -DimportArgs=\"--update\"
```

7.4.2   Running Export from Buildomatic

The `export` target for ant has the following syntax:

Windows:	`js-ant export -DexportFile=<filename> -DexportArgs="<export-options>"`
Linux and Mac OSX:	`./js-ant export -DexportFile=<filename> -DexportArgs=\"<export-options>\"`

The export file format is a ZIP file or a set of files under a new directory name. If you specify the .zip extension for your output filename, a ZIP archive is created automatically. Otherwise, an uncompressed directory with files and sub-directories is created.

The `exportArgs` argument requires double quotation marks (") and can contain more than one export option, as shown in these Windows examples:

```
js-ant export-help-pro
js-ant export -DexportFile=my-domains.zip
    -DexportArgs="--uris /datasources"
js-ant export -DexportFile=my-reports-and-users.zip
    -DexportArgs="--uris /reports
    --users jasperadmin,joeuser"
js-ant export -DexportFile=my-datasources
    -DexportArgs="--uris /datasources --roles ROLE_USER"
js-ant export -DexportFile=js-everything.zip -DexportArgs="--everything"
```

On Linux, all double quotation marks (") must be escaped with a backslash (\). Also, when listing user names, enclose the list in single quotation marks ('), as shown in this Linux example:

```
./js-ant export-help-pro
./js-ant export -DexportFile=my-reports-and-users.zip
    -DexportArgs=\"--uris /reports
    --users 'jasperadmin,joeuser'\"
```

CHAPTER 8   SYSTEM CONFIGURATION

You can change the default behavior of JasperReports Server by editing the system's configuration. The configuration is defined by a set of properties and their values.

The properties are stored in configuration files located in various folders under the <js-install> directory, which is the root of your JasperReports Server installation. To change the configuration, you edit these files and then restart the server.

A few of the most commonly edited properties are available to the administrator through the user interface (UI). Changes to these properties take effect immediately, are stored in the repository, and override the equivalent values stored in files, even after the server restarts (as of version 5.0).

This chapter describes a subset of the properties in the configuration files. Settings that affect security are covered in *JasperReports Server Security Guide*. More options are described in the *JasperReports Server Community Project Installation Guide*.

Because the locations of files described in this chapter vary with your application server, the paths specified here are relative to the deployed WAR file for the application. For example, the applicationContext.xml file is shown as residing in the WEB-INF folder; if you use the Tomcat application server bundled with the installer, the default path to this location is:

> C:\Program Files\jasperreports-server-6.4\apache-tomcat\webapps\jasperserver-pro\WEB-INF

> ⚠ Use caution when editing the properties described in this chapter. Inadvertent changes may cause unexpected errors throughout JasperReports Server that may be difficult to troubleshoot. Before changing any files, back them up to a location outside of your JasperReports Server installation.
>
> Do not modify settings that are not described in the documentation. Even though some settings may appear straightforward, values other than the default may not work properly and cause errors.

> 📝 In addition to configuration described in this chapter, you can configure Liferay Portal or JBoss Portal to display the reports stored in your JasperReports Server instance. You can download the JasperReports Server portlets for these environments from the Jaspersoft Support Portal. For information about how to deploy the portlet, refer to the documentation in the portlet download package.

This chapter contains the following sections:
* **Configuration Settings in the User Interface**
* **Configuration for Using Proxies**
* **Configuration for Session Persistence**
* **Enabling Data Snapshots**
* **Configuring Cloud Services**

- **Configuring JasperReports Library**
- **Configuring Input Control Behavior**
- **Configuring the Scheduler**
- **Configuring Report Thumbnails**
- **Configuring the Heartbeat**
- **Configuring the Online Help**

8.1   Configuration Settings in the User Interface

As of JasperReports Server 5.0, changes to configuration settings in the user interfaces are persistent through server restarts. Previously, upon restart configuration settings would revert to values stored in the configuration files.

To make persistent configuration changes through the JasperReports Server user interface:

1. Log in as administrator (`jasperadmin` by default).

2. Select **Manage > Server Settings**:
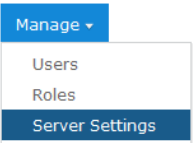
3. Choose a category of settings or administrator actions from the left-hand Settings panel.

Figure 8-1 The User Interface for Configuration Settings

4. Find the configuration setting you want to change and edit its value. In the case of log levels, the new value takes effect immediately. In the case of other settings, click **Change** beside the individual setting.

   The settings and administrator actions are documented in their respective sections:

Settings	Documentation
Log Settings	**Chapter 9, "Configuring System Logs," on page 155**
OLAP Settings	*Jaspersoft OLAP User Guide*
Cloud Settings	**8.5, "Configuring Cloud Services," on page 134**
Server Attributes	**2.3, "Managing Attributes," on page 24**
Restore Defaults	**8.1.2, "Restoring Default Settings," on page 127**
Import	**7.2.3, "Importing From the Settings," on page 115**
Export	**7.2.2, "Exporting From the Settings," on page 113**

8.1.1   Understanding Persistent Settings

When making changes through the Settings UI, you should understand how persistent settings made in the UI are stored internally and relate to the settings in configuration files:

- The Settings pages display a subset of the settings available in configuration files. Therefore, all settings in the UI also exist in a configuration file.
- By default, the Settings pages display the values of settings that exist in the corresponding configuration file. If you modify only the files and restart the server, your new file settings take effect on the server and are visible in the UI.
- When you change a value on the Settings pages, the new setting takes effect immediately, but the new value is *not* written to the corresponding configuration file. Instead, it's stored in the server's internal database so the value is persistent when the server is restarted.

> Only configuration settings that have a value modified on the Settings pages of the UI are stored and made persistent in the database.

- When the server restarts, any stored values take precedence over values for the same settings in the configuration files. However, each setting is independent, so a value that's not modified in the Settings UI is read from the configuration files.
- The Settings pages display the values of the settings in effect on the server.

> Be aware that the configuration values that appear on the Settings pages are possibly a mixture of values loaded from configuration files and from the persistent storage.

- Changing a setting that has already been modified updates its value stored internally, even if it is set to the same original value stored in a configuration file. The stored value continues to take precedence over any changes to this setting in the configuration file.

8.1.2   Restoring Default Settings

If a setting has been modified in the UI, it will remain in persistent storage and always override the corresponding setting in a configuration file. If you want to reset a value so that it is read from the configuration

file instead, use the Restore Defaults page.

To restore a default setting:

1. Log in as administrator (`jasperadmin` by default).

2. Select **Manage > Server Settings** and choose **Restore Defaults** from the left-hand panel.

Settings	Restore Defaults	
Log Settings	Name	Value
OLAP Settings	log4j.com.jaspersoft.jasperserver.war.cascade.CachedEngineService	WARN
Cloud Settings	aws.db.security.group.suppressEc2CredentialsWarnings	true
Server Attributes		
• Restore Defaults		
Import		
Export		

Save   Cancel

Figure 8-2 The Restore Defaults Page Containing Persistent Configuration Settings

The configuration values on the Restore Defaults page represent the settings that have been modified through the UI and are stored in persistent storage.

> The Restore Defaults page also includes JDBC drivers configured during the installation or through the data source creation wizard. Do not remove the drivers with `[SYSTEM]` values. For more information, see **4.3, "Managing JDBC Drivers," on page 52**.

3. To restore a setting to its configuration file default, click the 🗑 icon beside its current value and confirm.

4. Click **Save** to make the change permanent.

   The setting is removed from persistent storage and the value of the setting is restored to its default value from the corresponding configuration file. The next time the server restarts, its value will be read from the configuration file.

> The settings listed on the Restore Defaults page are those that can be exported to different servers or re-imported after a server upgrade. For more information, see **7.2.3, "Importing From the Settings," on page 115**.

8.2   Configuration for Using Proxies

When setting up JasperReports Server to use a proxy server, there are two additional settings to make in the server configuration. JasperReports Server exposes some URLs to itself through the UI and when using the scheduler to send emails. When using a proxy, those URLs must be configured so that they expose the proxy URL, not actual URL of the server instance.

Change the following settings so that JasperReports Server exposes the proxy URL.

Configuration for Using Proxies	
Configuration File	
.../WEB-INF/js.config.properties	
Property	**Description**
deploy.base.url	Set this property to the full URL for exposing the JasperReports Server UI through the proxy. This URL must include the application name, for example: http://bi.example.com/jasperserver-pro
Configuration File	
.../WEB-INF/js.quartz.properties	
Property	**Description**
report.scheduler.web.deployment.uri	This is the base URL used by the scheduler to generate links to reports in emails. Set this property to the full URL, including application name, that you expose through your proxy. It should be the same as deploy.base.url.

8.3   Configuration for Session Persistence

Many application servers can store user sessions while a web app is offline, for example when changing JasperReports Server configuration files. The app server remembers all the information about a user's session, such as the session ID and what page was being viewed; and when the web app restarts, the user session is restored. The user doesn't have to log in again, and often will not even notice that the server was temporarily unavailable. This is called session persistence.

Sessions do not persist when redeploying a web app in the application server, only when restarting the web app.

JasperReports Server supports a limited form of session persistence. When session persistence is enabled in the app server, sessions can be restored in the following cases:

- Browsing the repository, expanding folders in the repository tree and viewing folder contents.
- Searching the repository, including all search filters and results.
- Repository permissions dialog, including state and selections.
- Add folder dialog.
- Add resource dialogs, including adding or editing a data source, JasperReport, and other repository objects.
- Copy, cut, and paste resources in the repository.
- Scheduling a report, including all information such as a schedule and notifications.

If the server becomes unavailable when using the pages or dialogs above, the user will see a pause only when performing an action on these pages, such as submitting. When the server has finished restarting, the user can continue interacting with these UI elements. If the user takes no action while the server is unavailable, he may not even notice the server restart.

However, for other interactive dialogs in JasperReports Server, the state is too large to store in the user session. These features do not support session persistence and unsaved data can be lost:

- Interactive report viewer - The data in the report, as well as the state of column sorting and filtering can't be saved.
- OLAP viewer - The data in the OLAP view and current MDX expression can't be restored.
- Administration dialogs - When creating or editing users or roles, the information entered in a dialog can't be restored if it was not submitted.

In the cases listed above, the user's work is interrupted, and any unsaved work is lost. However, when the server restarts, the user does not have to log in again, the server displays a message about the session that could not be fully restored, and redirects the user to the home page. The user must relaunch the interactive feature and recreate any unsaved work.

Session persistence also affects web service calls. The REST API supports a login to store a session ID, and with persistence enabled, that session ID will still be valid when the application server restarts. This simplifies the code you need to handle timeouts. In general, web service calls do not support interactive work like designing an Ad Hoc view, a dashboard, a Domain, or exploring data in OLAP, so they aren't affected by the lack of session persistence in those cases. However, web service calls are affected in the following case:

- Report execution - All asynchronous API calls for running and exporting reports rely on the large JasperPrint object that can't be persisted. When the server restarts, the asynchronous calls will return errors because the reports could not be saved in the session. Your application needs to detect this error and include code for re-running the report.

> JasperReports Server also supports session replication among multiple instances of the server in a cluster. However, session replication has the same limitations because it is based on session persistence. For more information, see the *JasperReports Server Ultimate Guide*.

The following procedure gives the configuration in JasperReports Server and the Apache Tomcat application server to enable session persistence. For another application server, refer to that server's documentation.

To configure JasperReports Server and Apache Tomcat for session persistence:

1. Edit the file .../META-INF/context.xml to comment out the Manager property as follows:

```
<!-- Manager pathname="" /-->
```

2. Edit the file .../WEB-INF/web.xml to make the following changes.

   a. Locate the listener of class `RequestContextListener` and replace it with the listener of class `TolerantRequestContextListener`. The new listener class is given in comments that you need to uncomment as follows:

```
<listener-class>com.jaspersoft.jasperserver.core.util.TolerantRequestContextListener
    </listener-class>
<!--listener-class>org.springframework.web.context.request.RequestContextListener
    </listener-class-->
```

   b. Locate the `ClusterFilter` given in comments and uncomment it as follows:

```
    <filter>
        <filter-name>ClusterFilter</filter-name>
        <filter-class>com.jaspersoft.jasperserver.war.TolerantSessionFilter</filter-class>
    </filter>
```

c. Locate the corresponding mapping for the `ClusterFilter` and uncomment that, too. You must also uncomment the `<distributable>` element below it as follows:

```
    <filter-mapping>
        <filter-name>ClusterFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <distributable/>
```

3. Add the following property to your JVM environment:

```
-Dorg.apache.catalina.session.StandardSession.ACTIVITY_CHECK=true
```

4. Restart your Apache Tomcat application server.

8.4   Enabling Data Snapshots

The data snapshot feature was introduced in JasperReports Server release 4.7 to store report data in the server. Data snapshots create a significant change in the user experience:

*   Without data snapshots – Whenever users run a report, the server queries the data source and displays the latest data. When the same report is run over and over, the data source is often returning the same data every time. This is the behavior of all releases before 4.7, and the default behavior of release 4.7.
*   With data snapshots – The first time a report runs, it queries the data source and stores a copy of the data with the report in the repository. Users who view the report later see the data from the saved snapshot, not from querying the data source. Reports accessed through web service APIs are also based on the saved snapshot. For large reports or frequently viewed reports, the persisted snapshot provides a significant performance gain and reduces load on your data sources. Every user who has access to the report will see the data from the same snapshot. For users who require it, the report viewer provides a button to manually refresh the data snapshot anytime. In addition, when the scheduler runs a job on a report it always updates the snapshot. Data snapshots are implemented in JasperReports Server 4.7 and later, but must be enabled manually.

We encourage enabling data snapshots with the following recommendations:

*   If you have a new installation of JasperReports Server, enable snapshots to get the full server functionality. In the future, persistent data snapshots may be enabled by default.
*   If you are upgrading from a release earlier than 4.7, first proceed with the upgrade procedure and verify the outcome, as instructed in the *JasperReports Server Community Project Installation Guide*. Then, before enabling data snapshots, notify your users about this new functionality.
*   Data snapshots are stored in the server's repository, which must be sized accordingly. If you have a large number of reports, or very large reports, consider the performance of your repository database before enabling snapshots. If your users rely on data that changes frequently or if they expect to see real-time data when opening a report, do not enable snapshots. Alternatively, you can enable snapshots selectively as described below.

8.4.1   Global Data Snapshot Configuration

The server-level settings determine whether the snapshot feature is available on the server.

Data Snapshots Server-Level Configuration		
Configuration File		
.../WEB-INF/applicationContext-data-snapshots.xml		
Property	**Bean**	**Description**
`snapshot Persistence Enabled`	`dataSnapshot Service`	When set to `true`, it allows the JasperReports Server report viewer to save data snapshots in the repository and open them the next time the report is run. By default, this is set to `false`.

> There is also a property named `snapshotRecordingEnabled` that caches a snapshot in the report viewer memory when sorting and filtering columns interactively. This allows the report viewer to refresh the display without querying the database every time. Regardless of persistence, `snapshotRecordingEnabled` improves report viewing performance and decreases database load, so it should remain set to `true`.

8.4.2   Report-level Data Snapshot Configuration

You can disable snapshots on a specific report by setting the following property in the report's JRXML:

```
net.sf.jasperreports.data.cache.persistable=false
```

> This report-level property depends on the snapshot mechanism in JasperReports Server. This property has no effect in other report viewers without such a mechanism, like the viewer integrated in Jaspersoft Studio.

There are two ways to control snapshots at the report level. In the case above:
*   Data snapshots are enabled on the server, so most reports use them.
*   Reports that don't benefit from data snapshots can explicitly disable snapshots in their own JRXML.

As with all report-level properties, you can set server-wide default values, as described in **8.6, "Configuring JasperReports Library," on page 136**:

Data Snapshots Default Report-Level Configuration		
Configuration File		
.../WEB-INF/classes/jasperreports.properties		

Data Snapshots Default Report-Level Configuration	
Property	Description
`net.sf.jasperreports.data.cache.` `persistable=false`	When set to `false`, the server-wide default for reports is to not use data snapshots, however, they are still available if a report overrides this value to true in its own JRXML.

Because the report-level property takes precedence over the server-level property, this enables a second way to control snapshots:

- Data snapshots are enabled on the server.
- But the server-wide default is set to false, so most reports don't use them.
- Reports that benefit from data snapshots can explicitly enable snapshots in their own JRXML with:
  `net.sf.jasperreports.data.cache.persistable=true`

8.4.3   Data Snapshots in the Scheduler

Scheduled jobs always run the report by accessing the data source, so the output has up-to-the-minute data. When data snapshots are enabled on reports, the job always updates the data snapshot with this new data after it runs. This way, when you schedule a report, it also refreshes the data snapshot periodically: hourly, daily, weekly, or whatever suits your data requirements.

When data snapshots are enabled on the server, the scheduler interface has an extra option to output the data snapshot. This option, as shown in the following figure, generates a copy of the report with the new data snapshot. This copy is stored in the repository as a JasperReport, identical to the report being run. Over time, this will create an archive of your report data.

If you clear the Data Snapshot Output Format option, no copy of the report is saved with the new data snapshot, but the data snapshot on the original report is still updated when the job runs. Also, you must select at least one other output format to schedule the report.
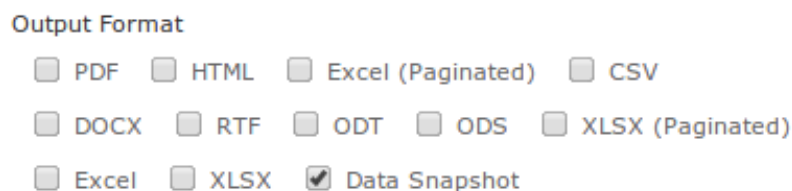
Figure 8-3 The Data Snapshot Output Option in the Scheduler

Finally, when data snapshots are enabled, you can also update them through REST web services calls. When specifying the report to run with the rest_v2/reportExecutions service, you can add arguments to explicitly update or not update the associated data snapshot. For more information, see the *JasperReports Server REST API Reference*.

8.5   Configuring Cloud Services

The following settings control how JasperReports Server interacts with a cloud service provider, such as AWS and Microsoft Azure:

- The Cloud Settings page enables you to create and change firewall rules without restarting the server. For the Microsoft Azure service, these rules are called "access rules." For AWS, they are called "security groups."
- The AWS configuration file allows you to change the JDBC driver used for AWS data sources.
- The Azure configuration file allows you to change the JDBC driver and URL template used for Azure data sources.

For more information about AWS and Azure data sources, see **4.5, "AWS Data Sources," on page 56** and **"Azure SQL Data Sources" on page 59**.

8.5.1   Changing Cloud Services Settings

To change cloud services settings:

1. Log in as administrator (`jasperadmin` by default).
2. Click **Manage > Server Settings**. The **Log Settings** page appears.
3. Click **Cloud Settings** in the left-menu.

    The Cloud Settings panel appears.

Figure 8-4 Cloud Settings Page

> We set up one AWS DB Security Group (using IP address) in each RDS region, per JasperReports Server instance. The security group allows connections from the specific JasperReports Server instance to the specified AWS database instance.

4. Modify the following settings and click **Change** after each modification. Changes are effective immediately on the server:

   • **Automatically Set Up an Access Rule for JasperReports Server**: This check box is generally left checked. When checked the JasperReports Server will automatically create and update an access rule that allows connections from JasperReports Server to the database hosted by the cloud service provider. If you want to manage access rules manually, uncheck this box.

   • **Access Rule Name**: When JasperReports Server creates access rules to support cloud-based data sources on this instance, it will use this name as the basis of the access rule name. When the JasperReports Server instance is running on AWS EC2, the EC2 instance ID will be appended. When running outside of AWS EC2, you must make sure that name is unique among JasperReports Server instances (*i.e.*, each instance should have its own name), so the IP addresses are properly granted access to the appropriate database instances.

   • **Access Rule Description**: This text will be used as the description for the access rule.

   • **JasperReports Server Public IP:** Enter the public IP address for JasperReports Server. Most users on AWS EC2 should leave the this field empty and let JasperReports Server determine the IP address automatically. It is possible with complex EC2 topology involving Virtual Private Clouds (VPCs) that you need to provide your IP address manually.

   • **Suppress EC2 Credentials Warning**: If your JasperReports Server instance was created with no IAM role, when you go to the data source wizard to add an AWS data source with EC2 credentials there will be a warning message saying there is no proper role set. Checking this box suppresses the warning and disables the option.

8.5.2   Changing the Default JDBC Driver for AWS Data Sources

When adding an AWS data source, JasperReports Server uses the JDBC driver specified in the .../WEB-INF/applicationContext-webapp.xml file. You can configure JasperReports Server to use a different JDBC driver.

To change the JDBC driver used with AWS data sources:

1. Open the file .../WEB-INF/applicationContext-webapp.xml for editing.

2. Locate the `jdbcConnectionMap` bean and the key of your AWS database type within it. Modify this key to specify a different JDBC driver. For example, the default driver for MySQL databases is set to the MariaDB driver:

```
<entry key="mysql">
<util:map>
    ...
    <entry key="jdbcUrl" value="jdbc:mysql://$[dbHost]:$[dbPort]/$[dbName]"/>
    <entry key="jdbcDriverClass" value="org.mariadb.jdbc.Driver"/>
    ...
</util:map>
```

3. Save the file and restart JasperReports Server.

8.5.3   Changing Azure SQL Data Source Defaults

When creating an Azure SQL data source, JasperReports Server uses the JDBC driver and JDBC connector URL template specified in the .../WEB-INF/applicationContext-azure-sql-datasource.xml configuration file. You have the choice of changing the URL string and using the Microsoft JDBC driver for SQL Server when creating the data source or you can change the default driver and URL string in the configuration file.

Azure Data Source Defaults	
Configuration File	
.../WEB-INF/applicationContext-azure-sql-datasource.xml	
Bean	**Description**
`defaultAzureJdbcDriverClassName`	The JDBC driver used for Azure SQL Server data sources. The default is the native JDBC driver for SQL Server (`tibcosoftware.jdbc.sqlserver.SQLServerDriver`).
`defaultAzureJdbcUrlSyntax`	The template for the JDBC connector URL for Azure SQL Server data sources. The connector string contains properties required for the JDBC driver to access the Azure data source. The default string specifies the server name, database name, and the use of SSL as an encryption method.
Configuration File	
.../WEB-INF/applicationContext-webapp.xml	
Bean	**Description**
`defaultAzureKeyStoreType`	The file format for storing the certificate key exchange files used for Azure data sources. The default file format is pkcs12.

8.6   Configuring JasperReports Library

JasperReports Server's reporting features are built on the JasperReports Library, which is embedded in the server. Many of the options you can configure to change the server's functionality are actually JasperReports Library options. The configuration options can control many aspects of the server's behavior, from the way reports are exported into different file formats, to the default font.

These options can be set at different levels:

- Global – Applies to all reports generated by the server. Global JasperReports properties are defined in the .../WEB-INF/classes/jasperreports.properties file.
- Report – Defined in the JRXML of the report and applies to that specific report.
- Element – Defined in the JRXML and applies to specific elements of the report.

For more information about JasperReports Library configuration, see http://jasperreports.sourceforge.net/config.reference.html.

The following sections highlight a few of the available options:

- • **Extending JasperReports Library**
- • **Changing the Crosstab Limit**
- • **Setting a Global Chart Theme**
- • **Disabling Interactivity in the Report Viewer**
- • **Disabling Chart Types in Dashboards**
- • **Enabling the XHTML or HTML Exporters**
- • **Changing the Pro Charts Rendering Engine**
- • **Configuring a JavaScript Engine for Graphical Report Rendering**
- • **Static Export Properties for High Charts**
- • **Enabling PDF Accessibility Features in Tables**

8.6.1   Extending JasperReports Library

You can extend JasperReports Library by implementing the public interfaces it exposes.

Such an implementation is usually stored in a JAR (Java Archive) that contains a file called jasperreports_extension.properties, and specifies a factory class used to instantiate an extension registry. The extension registry specifies one or more extension objects, each of which corresponds to a JasperReports Library extension point represented by a Java interface.

Place this JAR on the JasperReports Library classpath, and your extension is automatically available.

For more information, refer to *JasperReports Library Ultimate Guide*.

8.6.2   Changing the Crosstab Limit

If you use crosstab reports, you may experience Out of Memory errors if the reports are very large or complex. You can configure JasperReports Server to return a message instead of memory errors when users run such crosstabs. To do so, enable the `net.sf.jasperreports.crosstab.bucket.measure.limit` property and set its maximum value in the following configuration file:

Crosstab Report Configuration Option	
Configuration File	
.../WEB-INF/classes/jasperreports.properties	
Property	**Description**

`net.sf.jasperreports.`
`crosstab.bucket.`
`measure.limit`
	This value effectively limits the number of cells in a crosstab, which can be computed as follows:
	
	(number of crosstab rows) x (number of crosstab columns) x (number of user-defined measures + 1)
	
	The default value is `100000`.
	
	Enter large values to allow your users to create larger, more complicated crosstabs; enter small values to restrict them. If you experience `OutOfMemoryExceptions` after changing this value, try setting it to a smaller number, or configure your JVM to allow more memory to be used.

8.6.3 Setting a Global Chart Theme

Chart themes control the look and feel of the charts generated by JasperReports Server. Chart themes can be applied at the level of either the server or the individual report:

- To apply a theme at the report level, select it when designing the report in Jaspersoft Studio. Note that you can also apply a theme to individual chart elements. Note that a chart theme can be included in a report unit as a resource. In this case the theme is available only to charts in that report unit.
- To apply a theme at the server level, copy the chart theme JAR to the correct location and edit its configuration file.

A chart theme is a JAR file that defines the look and feel of a chart. Once you have created the chart theme JAR file, copy it to the .../WEB-INF/lib directory. Chart themes in this location are available to any chart in the instance of the server. They can also be set as the global chart theme.

To set a theme as the default chart theme, edit the following configuration file:

Global Report Theme	
Configuration File	
.../WEB-INF/classes/jasperreports.properties	
Property	**Description**
`net.sf.jasperreports.` `chart.ChartTheme`	The name of a chart theme that is in the .../WEB-INF/lib directory.

We recommend that you create your chart themes in Jaspersoft Studio. Click **File** > **New** > **Other > Chart Theme**, then use Jaspersoft Studio to archive the new chart theme as a JAR.

Chart themes do not apply to Ad Hoc chart views.

8.6.4 Disabling Interactivity in the Report Viewer

By default, the report viewer's interactivity is enabled, and reports with interactive elements (such as the table component) are interactive when run in the web server and displayed in the viewer. If you don't want your reports to be interactive, you can disable interactivity across all reports by editing the following configuration file.

Interactivity in the Report Viewer
Configuration File
.../WEB-INF/classes/jasperreports.properties

Interactivity in the Report Viewer	
Property	**Description**
`net.sf.jasperreports.` `components.table.interactive`	By default, this property is set to `true`; in this case, interactivity is enabled in the report viewer. Set it to `false` to disable interactivity.

Changing this setting in this configuration file changes the behavior for the entire server. To configure this behavior at the report, table, or column level, edit the report's JRXML properties in Jaspersoft Studio.

8.6.5   Disabling Chart Types in Dashboards

By default, interactivity is enabled in charts that appear in dashboards, and users can change the chart type by clicking on the chart type selector (gear icon). If you don't want users to modify the chart types, you can disable the chart type selector on all dashboard charts by editing the following configuration file.

Chart Type Selector in Dashboards	
Configuration File	
.../WEB-INF/classes/jasperreports.properties	
Property	**Description**
`com.jaspersoft.jasperreports.` `highcharts.interactive`	By default, this property is set to `true`; in this case, chart types can be modified in dashboards. Set it to `false` to prevent users from changing dashboard charts.

Changing this setting in this configuration file changes the behavior for the entire server. To configure this behavior at the report level, edit the report's JRXML properties in Jaspersoft Studio.

8.6.6   Enabling the XHTML or HTML Exporters

By default, JasperReports Server exports HTML format using an HTML-based exporter. Unlike the default exporters from previous versions of the server (`html` and `xhtml`), the new HTML exporter (`html2`) is more forgiving when exporting reports that have overlapping elements while still preventing text from being cut off due to font metrics issues.

This setting affects all cases when HTML is exported, including when reports are exported from the report viewer and when they are scheduled to produce HTML output.

To use an older HTML exporter in JasperReports Server, edit the following configuration file:

HTML Exporters
Configuration File
.../WEB-INF/classes/jasperreports.properties

Property	Description
`com.jaspersoft.jasperreports.export.html.type`	Determines which of the HTML exporters is used. Valid values are: • `html2` is the default HTML exporter. It handles overlapping report elements more gracefully than the other exporters. • `xhtml` was the default HTML exporter in versions from 4.5 to 5.0. We continue to support this exporter. It handles overlapping report elements more gracefully that the `html` exporter. However, it is subject to font metric mismatches between client browsers, which can result in text being cut off. • `html` was the default HTML exporter in versions of JasperReports Server before 4.2. Jaspersoft continues to support this exporter. It doesn't handle overlapping elements as gracefully as the other exporters.

Note that the properties are mutually exclusive; you can have only one uncommented at a time.

> As of JasperReports Server version 5.5, if your reports include interactive elements such as the table component (which supports sorting and filtering in the HTML viewer), you must use the `html2` exporter to enable the interactive features; the `html` and `xhtml` exporters don't support them.

8.6.7 Changing the Pro Charts Rendering Engine

By default, JasperReports Server renders Pro Charts (those based on Fusion Charts) using HTML5. If your browser doesn't support HTML5, the chart isn't be rendered unless you change the configuration. Though we don't recommend it, you can instead have these elements rendered by Adobe Flash; in some circumstances, Fusion's Flash solution may pose a security risk.

Note that Pro Charts are available only in the JasperReports Server Professional edition.

To render Pro Charts using HTML5, edit the following configuration file:

Pro Charts Renderer
Configuration File
.../WEB-INF/classes/jasperreports.properties

Pro Charts Renderer	
Property	**Description**
`com.jaspersoft.jasperreports.` `fusion.charts.render.type`	Determines which of the following renderers is used: • `html5` is the default renderer for Pro Charts. It is our preferred renderer. • `flash` is a deprecated renderer for Pro Charts.

Note that this property applies only to reports that rely on Pro Charts and affects only the HTML preview and export.

Typically, this property is set at the server level; to override the server-level setting for a specific Pro Chart report, set this property at the report level, and also specify a second property as shown:

```
net.sf.jasperreports.print.transfer.fusion=com.jaspersoft.jasperreports.fusion
```

This allows the reporting engine, JasperReports Library, to recognize the Fusion settings. If this property isn't set, the `com.jaspersoft.jasperreports.fusion.charts.render.type` property is ignored at the report level.

8.6.8 Configuring a JavaScript Engine for Graphical Report Rendering

Depending on the circumstances, a given graphical element (such as a chart, a map, or a widget) in a report can be rendered in two ways:
• When run directly in the web UI, the browser itself renders the chart.
• When scheduled to run later or run in the background, an internal engine renders the chart.

By default, JasperReports Server's internal JavaScript engine is Rhino, which is an excellent solution for most cases; most JasperReports Server users can accept this default. However, you may want to investigate other engines if you encounter any of the following issues when scheduling chart-based reports or running them in the background:
• Poor performance when generating complex charts or charts that contain large volumes of data.
• Out of memory messages.
• Incorrect scaling when certain Pro Chart reports are printed.
• Results that don't match those generated when the report is run directly in the web UI. For example, text elements incorrectly sized or placed.

An alternative engine is provided in our installers; PhantomJS executes JavaScript when generating graphical reports that are run in the background or scheduled. PhantomJS is a headless WebKit with JavaScript API. To use PhantomJS when JasperReports Server is installed from the WAR file, download the correct version for your environment from PhantomJS and install it on the computer hosting JasperReports Server. For installation instructions, refer to the documentation provided with PhantomJS.

Once PhantomJS is installed, point JasperReports Server to its location. You can configure several processes to use PhantomJS: HighCharts generation, Pro Charts generation (including Pro Widgets and Pro Maps), and exporting dashboards.

> 📝 These are a server-wide settings. In a given server, all charts of the same type (HighCharts or Fusion (Charts Pro, Maps Pro, or Widgets Pro) use the same JavaScript engine.
>
> You can't use PhantomJS to render JFreeCharts. Such reports are always generated by Rhino when run in the background or scheduled.

To configure JasperReports Server to use PhantomJS for HighCharts and Pro Charts, including Pro Widgets and Pro Maps, edit the following properties:

JavaScript Engine Configuration for HighCharts and Pro Charts	
Configuration File	
.../WEB-INF/classes/jasperreports.properties	
Property	**Description**
`net.sf.jaspersoft.jasperreports.phantomjs.executable.path`	This property points to the engine the server should use to generate HighCharts-based and Pro Charts-based charts in reports run in the background or scheduled. For example, if you're using Windows and you expanded the PhantomJS 2.0 ZIP file into the root of your C: drive: `net.sf.jaspersoft.jasperreports.phantomjs.executable.path=C:\\phantomjs-2.0-windows\\phantomjs.exe`
`com.jaspersoft.jasperreports.components.customvisualization.phantomjs.executable.path`	This property points to the engine the server should use for exporting Jaspersoft Studio reports with custom visualization components. See the *Jaspersoft Studio User Guide* for more information.
`com.jaspersoft.jasperreports.phantomjs.tempdir.path`	The temporary directory where PhantomJS stores its output. By default, JasperReports Server expects this output in the location defined by Java's `java.io.tmpdir` system property.
`com.jaspersoft.jasperreports.phantomjs.executable.timeout`	The maximum number of milliseconds to wait for output from PhantomJS before the chart times out. The default is 3000.

PhantomJS can also be used to render dashboards when exporting them to a PNG, PDF, ODT, or DOCX file. To support Japanese and Chinese fonts, as well as certain icons in dashboard output, Jaspersoft recommends using PhantomJS 2.0 or later.

> 📝 If you are using JasperReports Server and PhantomJS on a Windows platform, exporting large Fusion reports may cause issues. In those cases, we recommend editing the properties as follows:
>
> • Change `net.sf.jaspersoft.jasperreports.phantomjs.executable.path` to `com.jaspersoft.jasperreports.fusion.phantomjs.executable.path`.
> • Set `com.jaspersoft.jasperreports.fusion.phantomjs.executable.timeout` to `600000` to increase the timeout period.

To configure JasperReports Server to use PhantomJS for exporting dashboards, edit the following property:

JavaScript Engine Configuration for Dashboards	
Configuration File	
.../WEB-INF/js.config.properties	
Property	**Description**
`phantomjs.binary`	This property points to the engine the server should use to generate dashboards when exporting. For example, if you are using Windows and you expanded the PhantomJS 2.0 ZIP file into the root of your C: drive: `phantomjs.binary=C:\\phantomjs-2.0-windows\\phantomjs.exe`

After setting this property, restart JasperReports Server to enable it.

8.6.9 Static Export Properties for High Charts

When you create interactive JasperReports that use Highcharts, and those reports are exported to HTML, the `<script>` elements in the HTML must reference the correct JavaScript libraries. The following JasperReports Library properties define the library paths:

Static Export Properties for Highcharts
Configuration File
.../WEB-INF/classes/jasperreports.properties
Property

`# Highcharts static export properties`
`com.jaspersoft.jasperreports.highcharts.render.require.js$context.url`
`com.jaspersoft.jasperreports.highcharts.jquery.js$context.url`
`com.jaspersoft.jasperreports.highcharts.highcharts.js$context.url`
`com.jaspersoft.jasperreports.highcharts.highcharts.more.js$context.url`
`com.jaspersoft.jasperreports.highcharts.highcharts.heatmap.js$context.url`
`com.jaspersoft.jasperreports.highcharts.data.service.js$context.url`
`com.jaspersoft.jasperreports.highcharts.default.service.js$context.url`
`com.jaspersoft.jasperreports.highcharts.item.hyperlink.service.js$context.url`
`com.jaspersoft.jasperreports.highcharts.y.axis.service.js$context.url`
`com.jaspersoft.jasperreports.highcharts.dual.pie.service.js$context.url`
`com.jaspersoft.ji.adhoc.highcharts.setting.service.js$context.url`
`com.jaspersoft.jasperreports.highcharts.highcharts.chart.producer.js$context.url`

Each property that ends in `$context.url` may also have a similar property ending in `$url`. The `$context.url` property is used when a request object is available to resolve a context path for the HTML export, and the `$url` property is used as a fallback.

Not all library paths are defined in the jasperreports.properties file. Some are provided by the environment (such as `com.jaspersoft.jasperreports.highcharts.jquery.ui.js`), and others are set programmatically by the bean `com.jaspersoft.ji.adhoc.jr.AdhocHighchartsSetting-ServiceBundle` (in the .../WEB-INF/applicationContext-adhoc.xml file).

8.6.10   Enabling PDF Accessibility Features in Tables

JasperReports Library supports properties and metadata that allow people to create accessible PDF documents that can be read by screen readers. These properties help meet the requirements specified in the Section 508c of the United States Rehabilitation Act of 1973.

In release 6.2, JasperReports Library now provides the ability to automatically add 508c metadata to table components. When this features is enabled, tables such as those in Ad Hoc reports and your own JasperReports contain the necessary metadata when exported to PDF.

Enabling PDF Accessibility Features in Tables	
Configuration File	
.../WEB-INF/classes/jasperreports.properties	
Property	**Description**
`net.sf.jasperreports.components.table.generate.pdf.tags`	When set to true, metadata for accessibility is automatically generated in JasperReports tables.

This property can also be set at the report level or table component level to control the automatically generated metadata.

8.7   Configuring Input Control Behavior

When defining text input controls, the default server behavior allows empty strings, even if you've configured a regular expression and made the input control mandatory. Use this setting to enforce the regular expression even on empty strings. This forces the user to provide a conforming value for the input control.

Input Control Behavior
Configuration File
.../WEB-INF/applicationContext-cascade.xml

Input Control Behavior	
Bean	**Description**
`applyRegexpToEmptyString`	The default value of `false` gives the traditional behavior: even if a regular expression is defined, it is not applied to empty strings. If you want to strictly enforce the regular expression, even on empty input strings, set this property to `true`.

You can also configure the default value that appears in each type of input control. This is the value that is displayed when the input control is not given any value. By default, the display value is ~NULL~.

Edit the file .../WEB-INF/applicationContext-cascade.xml to change the following entries. The examples in comments show how you can use the default value to suggest a pattern for the input. To make an input control appear blank when no value is given, set value="" (an empty string).

```
<util:map id="globalDefaultValues" value-type="java.lang.String" key-type="java.lang.Byte">
    <!-- if DataType isn't defined-->
    <entry key="-1" value="~NULL~"></entry>
    <!--TYPE_TEXT = 1-->
    <!--<entry key="1" value="Enter value"></entry>-->
    <entry key="1" value="~NULL~"></entry>
    <!--TYPE_NUMBER = 2-->
    <!--<entry key="2" value="0"></entry>-->
    <entry key="2" value="~NULL~"></entry>
    <!--TYPE_DATE = 3-->
    <!--<entry key="3" value="2020-03-12"></entry>-->
    <entry key="3" value="~NULL~"></entry>
    <!--TYPE_DATE_TIME = 4-->
    <!--<entry key="4" value="2015-09-22T05:26:16"></entry>-->
    <entry key="4" value="~NULL~"></entry>
    <!--TYPE_TIME = 5-->
    <!--<entry key="5" value="13:37:54"></entry>-->
    <entry key="5" value="~NULL~"></entry>
</util:map>
```

8.8   Configuring the Scheduler

The scheduler runs reports in the background according to a user-defined schedule (also called a job). You can configure the following aspects of the scheduler:

- **Configuring the Scheduler Misfire Policy**
- **Configuring Scheduler Failure Notifications**
- **Restricting File System Output**
- **Removing Report Scheduling Interval Options**
- **Adding a Holiday Exclusion Calendar**
- **Changing the Default Output Folder**

8.8.1   Configuring the Scheduler Misfire Policy

A scheduler misfire occurs when the scheduler cannot run a report at the designated time, for example because JasperReports Server is offline, its database is offline, or the number of threads is limited. In this case, you can configure the behavior of the scheduler to retry the report or skip the scheduled run.

You can set a different misfire policy for each kind of job schedule: single job, repeating job, and calendar job. Misfire policies are defined in the Quartz Scheduler documentation and other online resources:

http://quartz-scheduler.org/documentation/quartz-2.x/tutorials/tutorial-lesson-05

http://quartz-scheduler.org/documentation/quartz-2.x/tutorials/tutorial-lesson-06

http://nurkiewicz.blogspot.com/2012/04/quartz-scheduler-misfire-instructions.html

Configuring Scheduler Misfire Policy	
Configuration File	
.../WEB-INF/js.quartz.properties	
Property	**Description**
`report.quartz.misfirepolicy.` `singlesimplejob`	Sets the misfire policy for single jobs to one of the following: • `SMART_POLICY` • `MISFIRE_INSTRUCTION_FIRE_NOW` • `MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY`

`report.quartz.misfirepolicy.`
`repeatingsimplejob`
	Sets the misfire policy for repeating jobs to one of the following values:
	• `SMART_POLICY`
	• `MISFIRE_INSTRUCTION_FIRE_NOW`
	• `MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY`
	• `MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_`
	`EXISTING_COUNT`
	• `MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_`
	`EXISTING_REPEAT_COUNT`
	• `MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_`
	`REMAINING_REPEAT_COUNT`

`report.quartz.misfirepolicy.` `calendarjob`	Sets the misfire policy for jobs with calendar recursion to one of the following values: • `SMART_POLICY` • `MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY` • `MISFIRE_INSTRUCTION_FIRE_ONCE_NOW` • `MISFIRE_INSTRUCTION_DO_NOTHING`

8.8.2   Configuring Scheduler Failure Notifications

By default, if a scheduled report runs but causes an error, the scheduler sends an email to the schedule owner and to all JasperReports Server administrators. This is in addition to any failure notification addresses specified on the **Notifications** tab of the scheduler wizard. To receive these scheduler failure alerts, administrators must have valid email addresses defined in their user accounts.

You can also configure the scheduler to send failure notifications to different users based on roles, or turn off failure notifications.

Configuring Scheduler Failure Notifications		
Configuration File		
.../WEB-INF/applicationContext-report-scheduling.xml		
Entry Key	**Bean**	**Description**
administrator Role	quartz Scheduler	This setting determines the role to which the scheduler failure notifications will be sent. All users with this role and a valid email address defined in their user profile will receive the email notification. By default, this setting is ROLE_ ADMINISTRATOR.
disableSending AlertToAdmin	quartz Scheduler	Disables or allows failure notifications to be sent to the role in the previous setting. By default, this setting is false, meaning that notifications are sent. Set this value to true to disable scheduler failure notifications being sent to administrators (or the role defined above).
disableSending AlertToOwner	quartz Scheduler	Disables or allows failure notifications to be sent to the schedule owner. By default, this setting is false, meaning that notifications are sent. Set this value to true to disable scheduler failure notifications to the schedule owner.

8.8.3   Restricting File System Output

The scheduler outputs reports through several channels. Most reports are emailed, but reports can also be written to FTP folders. You can also configure the scheduler to write reports to the server's local file system. This option is disabled by default for security reasons.

If you turn on scheduler file system output, make sure you have configured user and folder access rights to make sure that malicious files cannot be written to your file system. The process that writes the files is the same user that runs the application server hosting JasperReports Server.

Scheduler File System Output
Configuration File
.../WEB-INF/applicationContext.xml

Property to Update	Description
enableSaveToHostFS	Set the value from "false" (the default) to "true". When true, the user interface for the scheduler displays active fields that allow the schedule creator to specify a folder in the server's file system. The scheduler will write files to this location every time it runs the schedule for this report. **Output Destination** ☑ Output To Repository /public/Samples/Reports Browse ☑ Output To Host File System /Archives/reports/daily This property also determines the scheduler's overall access to the file system. When true, any schedule configured with a file system folder will write to the file system. When false, no scheduled reports will write output to the file system (FTP and email output are not affected). However, any file system output specified in a schedule remains defined and will again trigger file system output when this property is true again.

8.8.4 Removing Report Scheduling Interval Options

Users can schedule reports to run at regular intervals. For simple recurrence, the default interval can be expressed in days, hours, or minutes. To prevent users from scheduling frequent reports, you can limit the intervals to days or hours by editing the following configuration file:

Scheduling Interval Options
Configuration File
.../WEB-INF/flows/reportJobBeans.xml

Section to Update	Description
recurrenceIntervalUnits	Comment out the intervals you want to disable.

To remove a temporal interval, enclose the corresponding bean in comment characters. For example, to keep users from scheduling reports at minute intervals, comment out the bean containing the INTERVAL_MINUTE field:

```
<!--
```

```
<bean class="com.jaspersoft.jasperserver.war.dto.ByteEnum">
    <property name="code">
        <util:constant static-field="com.jaspersoft.jasperserver.api.engine.scheduling.
            domain.ReportJobSimpleTrigger.INTERVAL_MINUTE"/>
    </property>
    <property name="labelMessage">
        <value>job.interval.unit.minute.label</value>
    </property>
</bean>
-->
```

8.8.5 Adding a Holiday Exclusion Calendar

The scheduler supports exclusion calendars to specify days or times when no report should be run, even if scheduled. For example, you might not want a report to run on a bank holiday when the financial data would be meaningless.

The scheduler maintains a list of named calendars, and the user interface allows the schedule creator to select a calendar whose dates will be excluded from the schedule.

Currently, the only method to define a holiday calendar is through the REST API. You can use any browser plug-in that acts as a REST client and can send PUT requests to JasperReports Server. Using such a plug-in, compose and send the following REST request (header and body) to your server:

```
PUT http://<host>:<port>/jasperserver[-pro]/rest_v2/jobs/calendars/2014FrenchHolidays
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<reportJobCalendar>
    <calendarType>holiday</calendarType>
    <description>2014 French Holidays</description>
    <excludeDays>
        <excludeDay>2014-01-01</excludeDay>
            <excludeDay>2014-04-18</excludeDay>
        <excludeDay>2014-04-21</excludeDay>
        <excludeDay>2014-05-01</excludeDay>
        <excludeDay>2014-05-08</excludeDay>
        <excludeDay>2014-05-29</excludeDay>
        <excludeDay>2014-06-09</excludeDay>
        <excludeDay>2014-07-14</excludeDay>
        <excludeDay>2014-08-15</excludeDay>
        <excludeDay>2014-11-01</excludeDay>
        <excludeDay>2014-11-11</excludeDay>
        <excludeDay>2014-12-24</excludeDay>
        <excludeDay>2014-12-25</excludeDay>
    </excludeDays>
    <timeZone>GMT+01:00</timeZone>
</reportJobCalendar>
```

For example, using the Poster plug-in for Firefox, you can submit this request as shown in the following figure. The figure also shows the successful reply from the server.
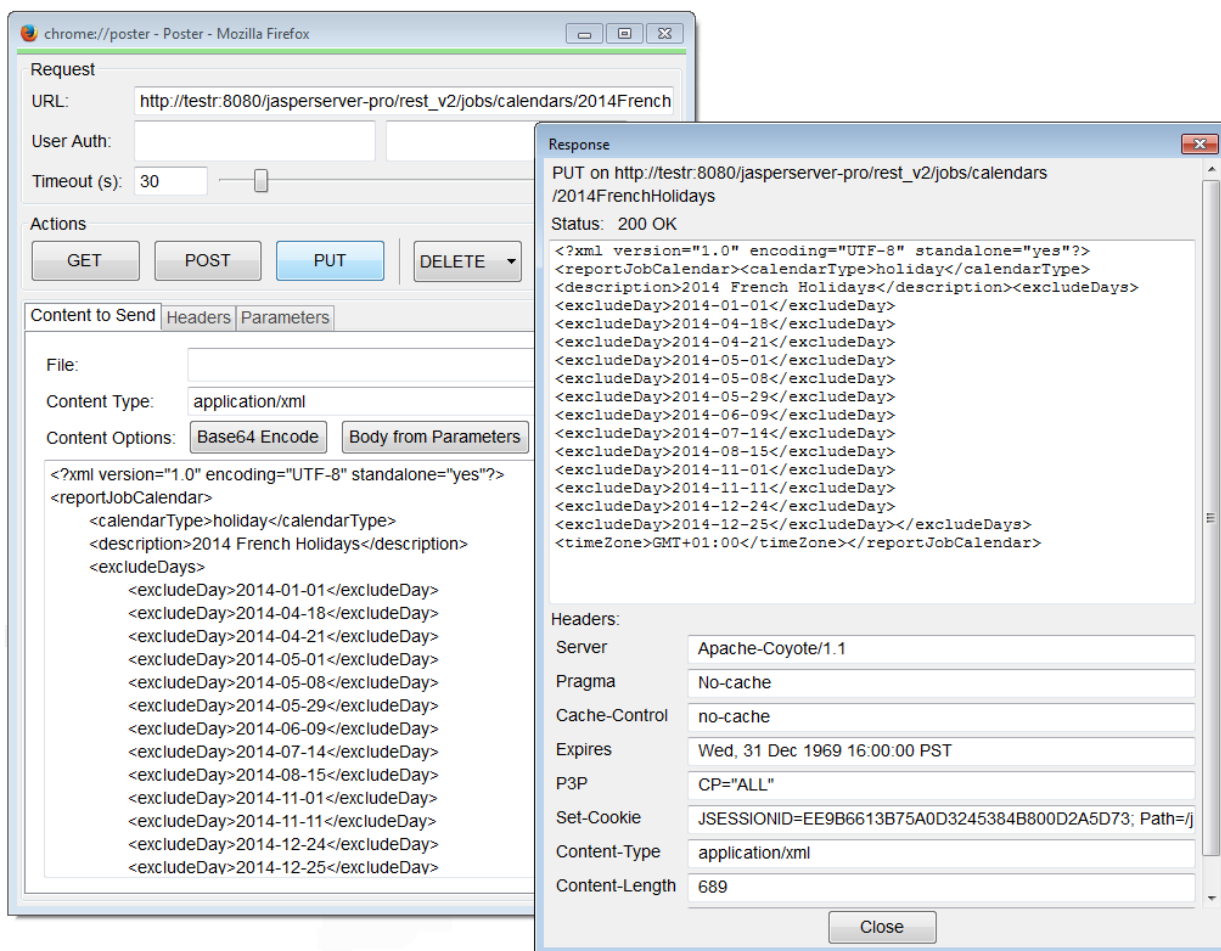
Figure 8-5 Creating a Holiday Calendar with REST Web Services

Then you should see your new calendar in the list of calendars in the Schedule tab.
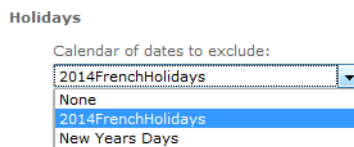
Figure 8-6 Selecting a Custom Holiday Calendar in the Scheduler

The REST API supports other types of calendars, however, the user interface lists only calendars of type holiday. Using the REST API, you can create and manage any number of calendars and update any schedule to use them. For more information, see the *JasperReports Server REST API Reference*.

8.8.6   Changing the Default Output Folder

Be default, the scheduler will save the output of scheduled reports to the /public/Samples/Reports folder in the repository and scheduled dashboard exports to the /public/Samples/Dashboards folder. You can change this default location to another folder in the repository by editing the applicationContext-report-scheduling.xml configuration file on the server.

To enable file data sources in the UI:

1. Open the file <js-install>/WEB-INF/applicationContext-report-scheduling.xml for editing.
2. Locate the element `<util:map id="reportJobDefaults">`.
3. Update `<entry key="scheduler.job.repositoryDestination.folderURI" value="/job_output" />` and replace `"/job_output"` with the URI for the new default folder in the repository.
4. Restart the server or redeploy the JasperReports Server web app. The new default folder appears on the Output Options tab when you try to schedule a report or dashboard.

8.9   Configuring Report Thumbnails

The JasperReports Server REST API has the ability to generate and export thumbnail images (small preview images) of reports and dashboards. The JasperMobile apps for Android and iOS use this feature to display small tiles with the report or dashboard image.

For more information about the REST API, see the *JasperReports Server REST API Reference*.

By default, report thumbnails are not active. If you use the JasperMobile apps with your server, you should turn on the report thumbnails:

Report Thumbnails	
Configuration File	
.../WEB-INF/js.spring.properties	
Property	**Description**
`property.reportThumbnailServiceEnabled=false`	When this property is set to `true`, JasperReports Server generates and sends report thumbnails in response to REST API requests. When it is set to `false`, report thumbnails are not generated. The default is false.

8.10   Configuring the Heartbeat

During installation (or the first time you log in as administrator), you're prompted once to participate in the Heartbeat program, which reports technical information to the JasperReports Server product team about your implementation, such as the operating system, JVM, application server, database (type and version), data source types, and server edition and version number.

If you change your mind, you can change the heartbeat behavior by editing the following configuration file:

Heartbeat Options	
Configuration File	
.../WEB-INF/js.config.properties	
Property	**Description**
`heartbeat.enabled=true`	When this property is set to `true`, JasperReports Server reports information about your environment to us once a week. When it is set to `false`, information is not sent.
`heartbeat.askForPermission.enabled`	Determines whether the administrator is prompted (the next time he logs into the web UI) about whether to allow heartbeat data to be sent. Typically, there is never cause to edit this property directly.
`heartbeat.permissionGranted.enabled`	Indicates whether a user has granted the server permission to send Heartbeat data. Setting this property to `false` prevents data from being sent.

All of these settings are properties that are substituted into the `heartbeatBean` in the .../WEB-INF/applicationContext-heartbeat.xml file.

8.11   Configuring the Online Help

JasperReports Server professional edition includes an online help system that describes the web interface. If your users don't have Internet connectivity, or if you don't want to provide access to this system, you can configure the server to hide the help links completely.

Online Help Configuration Options		
Configuration File		
.../WEB-INF/applicationContext-webHelp.xml		
Property	**Bean**	**Description**
`showHelp`	`webHelp`	Determines whether the help links are displayed in the JasperReports Server web UI. Valid values are `true` and `false`. The **Help** link appears at the top right corner of the web UI's pages.
`hostURL`	`webHelp`	Indicates the name of the computer hosting the web server where the help is running. The value depends on the version of JasperReports Server. Do not change this value.

Online Help Configuration Options		
`pagePrefix`	`webHelp`	Defines the default page name to pass to the web server hosting the help system. The only valid value is `Default_CSH.htm` for this property.
`helpContextMap`	`webHelp`	Maps contexts in the application to topic identifiers in the help system. Many pages in the web application are configured for context-sensitivity. When a user clicks **Help** on such a page, JasperReports Server loads a specific topic in the help system. The topic that appears is determined by a map in the applicationContext-webHelp.xml file. The only valid values are the defaults.

CHAPTER 9   CONFIGURING SYSTEM LOGS

System logs capture any errors in the code running the server, but can be configured to capture more information for debugging. If you suspect a problem or cannot pinpoint an error, you can turn log collectors on and off to capture logs while you test your issue.

This chapter contains the following sections:

- **Log File Location**
- **Managing Log Settings**
- **Log Configuration Files**
- **Adding a Logger to the Log Settings Page**

9.1   Log File Location

JasperReports Server uses the Apache log4j package to generate log files. Jaspersoft uses the slf4j facade to wrap log4j.

- The default log file is .../WEB-INF/logs/jasperserver.log.
- The default log configuration file is .../WEB-INF/log4j.properties.

To view the log file, you must have access to the file system where JasperReports Server is installed. This chapter describes the settings that control the information JasperReports Server writes to its logs.

9.2   Managing Log Settings

To set the current logging levels:

1. Log in as administrator (`jasperadmin` by default).
2. Select **Manage > Server Settings** and choose **Log Settings** in the left-panel.
3. In the Log Settings panel, use the drop-down selectors to change the log level for each class being logged.

Figure 9-1 System Log Settings

The page lists some of the currently-enabled loggers with their logging level. Any change to the logging levels takes effect immediately, without restarting JasperReports Server.

> These logging levels override the levels in the log4j.properties file. As of JasperReports Server version 5.0, these settings persist even when the server is restarted. Therefore the values on the Log Settings page that are in effect on the server may differ from settings in the log4j.properties file. For details, see **8.1, "Configuration Settings in the User Interface," on page 126**.

The four logging levels indicate the type of event recorded by a logger:

Setting	Level of Information
ERROR	Writes minimal information to the log, only when describing serious program faults.
WARN	Writes error and warning messages to the log. Warning messages contain cautionary information to help you to decide whether the logged events require your attention.
INFO	Writes error, warning, and informational messages to the log describing significant events, such as those that affect application performance.
DEBUG	Writes error, warning, informational, and additional messages to the log. Debug messages are very detailed and often voluminous. Use this setting only to diagnose a problem. DEBUG can impact system performance and should not be used in production environments. If several loggers are set to DEBUG, the server may generate huge logs, and performance can suffer.

JasperReports Server's default root logger setting is ERROR, as configured in log4j.properties. A logger without an assigned value inherits the setting of its parent in log4j.properties.

The following table lists each logger name as it appears on the Log Settings page, the identifier used to find it in the log file, and a description of the logger.

Logger Name	Identifier in Log	Description
SQL query executer	JRJdbcQueryExecuter	Logs SQL text and parameter values for queries that are run by the SQL query executer.
Input control value queries	valueQueryLog	Logs SQL text and parameter values for queries associated with input controls.
Cascading input control parameter resolution	FilterCore	Logs activity associated with cascading input controls. Query-driven input controls can cascade when a query has a parameter whose value comes from another input control. When the parameter value is changed, the query is automatically rerun, possibly changing the list of values for its input control.
Cascading input control query result caching	TokenControlLogic	Logs use of the cache for results of cascading input control queries.
Profile attributes resolver	ProfileAttributesResolverImpl	Determines the values of attributes when they are referenced.
Hibernate SQL	SQL	Logs SQL run by the Hibernate layer to access the JasperReports Server repository database. This logger generates a large volume of logging that could affect performance.

You can add other loggers to the Log Settings page if you know their classnames.

To add a logger to the page from the web interface:

1. Log in as administrator (`jasperadmin` by default).
2. Select **Manage > Server Settings** and choose **Log Settings** in the left panel.
3. Scroll to the bottom of the page.
4. Enter the logger's classname in the text field. See the other properties on the page for guidance, for example:
   `com.jaspersoft.ji.adhoc.action.AdhocCrosstabAjaxController`
5. Use the drop-down to set the logging level.

The logger setting is persistent even when the server is restarted. However, the logger setting may not appear on the Log Settings page again. For information about adding loggers to this page permanently, see **9.4, "Adding a Logger to the Log Settings Page," on page 158**.

9.3 Log Configuration Files

You can edit the log configuration file to set loggers, logging levels, and log output, but you must restart the server for your changes to take effect.

If you've made modifications in the Log Settings UI, those settings are persistent in the repository, and take precedence over the configuration files. However, these changes are *not* written to the configuration files. Each setting is independent, so a value that's not modified in the Log Settings UI is taken from the corresponding file. For more information, see **8.1, "Configuration Settings in the User Interface," on page 126**.

Logger names are defined in the Java source. Loggers can have any name, but the Jaspersoft convention is to give them their full class names. In the log4j.properties file, the classname must be preceded by `log4j.logger.` For example, the classname `org.acegisecurity.intercept` is represented in the log4j.properties file `as log4j.logger.org.acegisecurity.intercept`. If you want to add a new logger, find its classname in the source.

The file you edit depends on whether you're configuring server logging or logging during import and export.

Functionality to Log	File Location
Import or Export	<js-install>/buildomatic/conf_source/iePro
JasperReports Server	.../WEB-INF/log4j.properties

If the logger is defined in the configuration file but commented out, simply remove the comment character (#) to add the logger. Otherwise, add the logger's classname and set the logging level.

The form of a logger definition should be:

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

where:
- `<logger-classname>` is the name of the class you want to monitor.
- `<log-level>` is ERROR, WARN, INFO, or DEBUG
- `<output-type>` is a standard output type, such as stdout. For example:
  ```
  log4j.logger.org.springframework.webflow=DEBUG, stdout, fileout
  ```

Restart the server for your changes to take effect.

9.4   Adding a Logger to the Log Settings Page

If you know of a log4j logger that JasperReports Server uses, you can add it to the Log Settings page available to the administrator. To add a logger, edit a configuration file.

> Because editing text files can be error-prone, we recommend that you add loggers from the web interface by entering them into the text field on the Log Settings page. Edit the configuration file only if you need to permanently add the logger.

To edit the list of loggers to be displayed on the page:
1. Edit the file .../WEB-INF/bundles/logger_descriptions_pro.properties.
2. Add a new line and specify the logger's classname and a brief description of it.
   Entries should be in the form `<logger-classname> = <description>`.
   See the other properties in the file for guidance, for example:
   ```
   com.jaspersoft.ji.adhoc.action.AdhocCrosstabAjaxController = Crosstab controller
   ```
3. Restart the server for your changes to take effect.

The logger_descriptions_pro.properties file controls the labels for the English locale. You can specify labels for other locales by editing the logger description property files for those locales. For example, to add the label in French, add an entry to the logger_descriptions_pro_fr.properties file. For more information on supporting other languages, refer to **Appendix B, "Localization," on page 177**.

APPENDIX A   TROUBLESHOOTING

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.

This appendix contains the following sections:

- **Number of Users Exceeded**
- **Running Out of Database Connections**
- **Custom URLs Not Loading in Dashboards**
- **Print View Not Displaying in Dashboards**
- **Scheduler Sending Multiple Emails**
- **Scheduler Not Sending STARTTLS Emails**
- **Scheduler Running Deleted Jobs**
- **Scheduler Timezones in Excel Output**
- **Charts Not Appearing in Excel Export**
- **Working With Data Sources**
- **Special Characters in Database Schemas**
- **Cassandra Reports Not Running**
- **Reverting to the Old Home Page**

A.1   Number of Users Exceeded

When you have more users defined in the server than your license allows, the login page displays a warning; users can still log in. After a 24-hour grace period, an email is sent to the administrator and users can no longer log in. Most server functionality is disabled. To re-enable server functionality:

- Contact Jaspersoft sales to purchase additional user licenses. When you install the new license, the server becomes fully functional for all users.
- Delete user accounts until the number of user accounts on the server is in accord with your license. When server functionality is disabled, administrators can still log on and select **Manage > Users** to delete user accounts. For more information see **2.1, "Managing Users," on page 17**.

A.2   Running Out of Database Connections

JasperReports Server manages a pool of connections for each JDBC data source. The default number of connections is 20, but if you run many reports concurrently against the same data source you may reach the connection limit and see degraded performance. In particular, using the web service APIs, REST clients can easily launch many report executions at the same time and reach the limit.

The connection pool size is limited to avoid having too much memory permanently allocated to connections. But if you need more concurrent connections on a regular basis, you can increase the limit with the following configuration:

Reducing the Size Limit for Ad Hoc Dimensions		
Configuration File		
.../WEB-INF/applicationContext.xml		
Property	**Bean**	**Description**
`<constructor-arg type="int" value="20"/>`	`dataSource ObjectPool Factory`	Change the default value to match your concurrent connections. Make sure you have enough memory to handle the connections and the concurrent report executions.

If you are using JNDI data sources, you can configure the number of connections in your application server. For more information, see the sections on JNDI in **A.10, "Working With Data Sources," on page 165**.

A.3   Custom URLs Not Loading in Dashboards

Dashboards allow you to specify frames that contain web pages loaded from custom URLs specified when designing the dashboard. These URLs can even include parameters from input controls. If the URL takes too long to load, JasperReports Server will display an error message instead of the content.

If you expect the custom URL to take longer than 10 seconds, you can change the default timeout as follows:

Configuring the Dashboard URL Loading Timeout	
Configuration File	
.../scripts/dashboard.designer.js	
Property	**Description**
`CUSTOM_URL_IFRAME_TIMEOUT`	Time in milliseconds that the server will allow a custom URL to load in a dashboard before displaying an error. The setting takes effect immediately when the file is saved, there is no need to restart the server.

A.4   Print View Not Displaying in Dashboards

If you cannot display the Print view for a dashboard, there might be an issue with the size of the input control values. Input control values are passed as URL parameters on this page, and the application server can limit the length of the URL that includes the parameters.

To avoid this limit and allow large numbers of input control values in dashboard print view, edit the following configuration file or the equivalent in your application server.

Configure Apache Tomcat to Accept Large Filter Values	
Configuration File	
<tomcat>/conf/server.xml	
Property	**Description**

`<Connector port="8080" protocol="HTTP/1.1"`
`        connectionTimeout="20000"`
`        redirectPort="8443"`
`        URIEncoding="UTF-8"`
`        maxPostSize="0"`
`        maxHttpHeaderSize=65535 />`
	Add the `maxHttpHeaderSize` parameter to set the number of bytes accepted in the URL by the app server; `"65535"` is equivalent to 64 KB. For more information, see the [Tomcat documentation]().

A.5   Scheduler Sending Multiple Emails

In cases where you have a cluster of JasperReports Server instances accessing the same repository, the schedulers in each instance can sometimes conflict and send multiple emails. The behavior depends on the run-time of the scheduled reports, because a long report may cause the scheduler on another node to attempt to run the same report before the first node finishes.

To change this behavior, set the following parameter in .../WEB-INF/js.quartz.base.properties:

```
org.quartz.jobStore.clusterCheckinInterval = 900000
```

In case a job fails on the first node, the check-in interval is meant to ensure that the job runs on a second node after this delay. Because the schedulers do not communicate directly, the second scheduler cannot distinguish between a node that had a failure and a node that is still running a job. The default value corresponds to 15 minutes.

This parameter can be adjusted as follows:
- If you have scheduled reports that take a long time to run, longer than 15 minutes, you may see multiple emails. Increase this parameter to an interval longer than your longest report's expected run-time.
- On the other hand, if you have small reports that finish quickly, the default value means that any scheduler or node problem isn't detected by the other scheduler before 15 minutes. If you have time-critical reports scheduled, you can lower this parameter, but the value should still exceed your longest expected report run-time.

Restart all of your server instances after changing this parameter.

A.6   Scheduler Not Sending STARTTLS Emails

If you use OpenJDK 7 as your JVM, the scheduler may not be able to send emails if you have configured it to use secure email with STARTTLS. The reports run as scheduled, but the emails are not sent, and an error is recorded in the server log.

This error occurs with the following configuration in the file .../WEB-INF/applicationContext-report-scheduling.xml:

```
<props>
    <prop key="mail.smtp.auth">true</prop>
    <prop key="mail.smtp.starttls.enable">true</prop>
    <prop key="mail.smtp.sendpartial">true</prop>
    ...
</props>
```

To fix this behavior, do one of the following:

- Upgrade to OpenJDK 8.
- Or upgrade to the latest `nss` library. The error no longer occurs with the following version:

```
Name        : nss
Arch        : x86_64
Version     : 3.19.1
Release     : 3.el6_6
```

A.7   Scheduler Running Deleted Jobs

In some cases, old versions of JasperReports Server did not delete the scheduled jobs when deleting a report. These jobs cause errors when the scheduler tries to run them, but you can't remove the jobs through the user interface. The server no longer creates such *orphan* jobs, but they may appear again when you upgrade or import a catalog that contains them.

If you accidentally imported orphan jobs, make the configuration change shown below and restart your server.

Automatically Deleting Orphan Jobs		
Configuration File		
.../WEB-INF/applicationContext-report-scheduling.xml		
Entry	**Bean**	**Description**
autoDeleteBroken UriReportJob	quartz Scheduler	Change this property from `false` (the default) to `true`. Orphan jobs are detected and deleted just before they run, so all orphan jobs will be deleted gradually over time.

A.8   Scheduler Timezones in Excel Output

When scheduling a report, you can specify the timezone that should be applied to the output instead of using the data source's timezone. This works for all outputs except Excel (XLS), both paginated and non-paginated. To overcome this limitation, you must explicitly enable timezone usage in XLS output.

Setting Timezones in Scheduled Excel (XLS) Reports		
Configuration File		
.../WEB-INF/classes/jasperreports.properties		
Property	**Value**	**Description**
`net.sf.jasperreports.export.xls.use.timezone`	`true`	Define this property as `true` to apply timezones in scheduled Excel (XLS) reports.

This property can also be defined on individual elements in the report's JRXML if only certain fields should apply the timezone.

A.9   Charts Not Appearing in Excel Export

When exporting a report to Excel, JasperReports Server usually removes images that decorate the report and that do not fit in the Excel data-centric layout. However, JasperReports Server also converts any charts to images and uses the special property `net.sf.jasperreports.export.xls.ignore.graphics` set to `false` to make the image appear. If your report does not set this property explicitly, the chart images do no appear in your reports when exported to Excel.

If you have a lot of reports with this issue, you can set the property on the server:

Charts Images in Excel Export	
Configuration File	
.../WEB-INF/classes/jasperreports.properties	
Property	**Description**
`net.sf.jasperreports.export.xls.ignore.graphics`	By default, this property is set to `true`; in this case, images and chart images from the report do not appear when exported to Excel. Set this property to `false` to make chart images appear in Excel exports.

A.10   Working With Data Sources

When adding a data source to JasperReports Server, several things can cause errors. Start by looking at the following general connectivity issues:

- Check that your database server is available and accepting TCP/IP connections from the host where JasperReports Server is installed.
- Check in your RDBMS that the username and password you're using are correct and have access to the selected database.
- Check for firewalls or network connectivity errors.

Many databases, including MySQL, also require the user grants to include the specific host from which connections are allowed. Otherwise, when testing the JDBC connection, a connection may not be allowed even though the username and password are correct. For more information, refer to the MySQL documentation for setting up users.

An easy way to test connectivity from the server to the database with a particular user is to use a tool such as SQuirreL or another DB query tool to connect to the database from the host of your JasperReports Server instance.

A.10.1 Logging JDBC Operations

You can enable additional logging to help you find the cause of the error. Set any or all of the following loggers in the server settings interface or in the .../WEB-INF/log4j.properties file:

- log4j.logger.com.jaspersoft.jasperserver.api.engine.jasperreports.service.impl.JdbcDataSourceService
- log4j.logger.com.jaspersoft.jasperserver.api.engine.jasperreports.service.impl. JndiJdbcDataSourceService
- log4j.logger.com.jaspersoft.jasperserver.war.action.ReportDataSourceAction
- log4j.logger.com.jaspersoft.commons.datarator.JdbcDataSet
- log4j.logger.com.jaspersoft.jasperserver.war.common.JasperServerUtil
- log4j.logger.com.jaspersoft.commons.semantic.dsimpl.JdbcDataSetFactory
- log4j.logger.com.jaspersoft.commons.semantic.metaapi.impl.jdbc.BaseJdbcMetaDataFactoryImpl
- log4j.logger.com.jaspersoft.jasperserver.war.validation.ReportDataSourceValidator

A.10.2 JDBC Drivers

As of JasperReports Server 5.1, JDBC driver management is automated and simplified. JasperReports Server ships with drivers for some databases, as listed in the dialog for creating data sources. If the JDBC driver for your database is not included, the administrator can easily upload the driver and use it immediately in a data source.

For instructions on updating JDBC drivers without restarting the server, see **4.3, "Managing JDBC Drivers," on page 52**. For instructions on adding JDBC drivers when using JBoss, see the next section.

A.10.3 JDBC Drivers on JBoss

For modularity reasons, the JBoss application server does not allow web applications such as JasperReports Server to change executable files, such as JDBC drivers, on the fly. In order for JDBC drivers to appear in the list of available drivers and be selectable, you must configure JBoss before you use the drivers to create a JDBC data source.

To add JDBC drivers on JBoss:

1. Download or copy the JDBC driver JAR file to the .../WEB-INF/lib/ directory.
2. Edit the jboss-deployment-structure.xml file as described in the following table:

Configuring JBoss for JDBC drivers	
Configuration File	
.../META-INF/jboss-deployment-structure.xml	
Property	**Description**
`resource-root path="<driver>.jar"`	Locate the resource root element for the JDBC driver you added and uncomment it. The name of the JAR file must match exactly the name of the JDBC driver that you upload. You can add a new resource root element if the JDBC driver of your choice is not given in the commented list.

3. Restart JBoss.

A.10.4 Database Permissions

When creating database users, you must ensure that they have the appropriate privileges to access data, as well as permission to connect from the server that JasperReports Server is running on.

*   The database user that you specify in your data source definition needs the privileges to run SELECT queries on the tables used in your reports. The server blocks any DROP, INSERT, UPDATE, and DELETE SQL commands through its SQL injection protection. In some cases, additional permissions may be required to execute stored procedures, depending on your configuration and needs.
*   If you accept the defaults during installation of JasperReports Server on Linux from an RPM using apt-get, rpm, or yum, the bundled PostgreSQL allows only the user who owns PostgreSQL to connect. Enter the following commands to connect:

```
su - postgres
psql -U postgres
```

*   Many databases, including MySQL, also require that the user permissions name the specific host from which connections are allowed. Otherwise, when testing the JDBC connection, a connection may not be allowed even though the user name and password are correct. For example, see the MySQL documentation for setting up users.

A fairly easy way to test permissions and connectivity is to use a tool such as SquirrelSQL or another DB query tool to connect to the database from the same host as JasperReports Server and to run typical queries against your database.

A.10.5 JDBC Database URLs

When you choose a JDBC driver, the data source creation wizard prompts you for the elements of the URL that are required for your database. In some cases, you may need to add certain arguments to the JDBC URL. Ensure that the database URL you entered when defining your JDBC data source is consistent with what is required for your specific database and database driver. The following table gives the default URLs and port numbers and examples of optional arguments supported by the most common databases:

Database	Default JDBC Database URL
PostgreSQL	jdbc:postgresql://<host>:5432/<db-name>
MySQL and MariaDB	jdbc:mysql://<host>:3306/<db-name>?tinylnt1isBit=false
Oracle	jdbc:tibcosoftware:oracle://<host>:1521;SID=ORCL
SQL Server	jdbc:tibcosoftware:sqlserver://<host>:1433;databaseName=<db-name>
IBM DB2	jdbc:tibcosoftware:db2://<host>:50000;databaseName=<db-name>
Cas-sandra	jdbc:cassandra://<host>:9042;DefaultKeyspace=
Hive	jdbc:tibcosoftware:hive://<host>:10001;TransactionMode=ignore
Impala	jdbc:impala://<host>:21050/default
Redshift	jdbc:tibcosoftware:redshift://<host>:5439;databaseName=<db-name>;TransactionMode=ignore
SparkSQL	jdbc:tibcosoftware:sparksql://<host>:10000;TransactionMode=ignore
Salesforce	jdbc:tibcosoftware:sforce://;SecurityToken=;TransactionMode=ignore;StmtCallLimit=1000;databaseName=[TMPDIR/]Salesforce

A.10.6 SQL Functions with TIBCO JDBC Drivers

As of JasperReports Server 5.6.1, TIBCO provides a set of JDBC drivers in the installed server. The TIBCO JDBC drivers are based on the Progress DataDirect Connect drivers and support a slightly different SQL syntax than the vendors' drivers. If you see errors like the one below when running reports or creating domains that use scalar functions, you'll need to modify your queries.

```
com.jaspersoft.commons.semantic.metaapi.MetaDataException: Cannot execute JDBC Query.
java.sql.SQLDataException: [TIBCO][SQLServer JDBC Driver][SQLServer]Conversion failed
when converting the varchar value 'Assets' to data type int.
```

For example, the following query does not work with the TIBCO JDBC drivers:

```
SELECT account_id+account_description AS account_concat from account
```

The correct syntax for the TIBCO JDBC drivers is as follows:

```
SELECT {fn CONCAT(account_id,account_description)} AS account_concat from account
```

For more information, see the Progress DataDirect page on scalar functions.

A.10.7 Enabling the TIBCO JDBC Drivers for Impala and Cassandra Data Sources

Starting in JasperReports Server 6.4, only one JDBC driver is enabled for each data source type by default. JasperReports Serveruses the Simba JDBC drivers for Impala and Cassandra data sources by default. If you want to use the TIBCO JDBC drivers for Impala (`tibcosoftware.jdbc.impala.ImpalaDriver`) or Cassandra (`tibcosoftware.jdbc.cassandra.CassandraDriver`) data sources, you can modify the .../WEB-INF/applicationContext-webapp.xml file to make the drivers active.

To enable the TIBCO JDBC drivers for Impala and Cassandra data sources:

1. Open the file .../WEB-INF/applicationContext-webapp.xml for editing.

2. Locate the `jdbcTibcoConnectionMap` bean and find the following lines for the inactive Impala and Cassandra JDBC drivers:

```
<!--  entry key="impala">
   ...
</entry -->
...
<!--  entry key="impala">
   ...
</entry -->
```

3. Modify the lines as follows to enable the JDBC drivers in JasperReports Server:

```
<entry key="impala">
   ...
</entry>
...
<entry key="impala">
   ...
</entry>
```

4. Save the file and restart JasperReports Server.

A.10.8 JNDI Services on Apache Tomcat

If you have trouble with a JNDI connection, you need to look at the JNDI definition for your database on your application server. This section gives common issues with JNDI definitions on Apache Tomcat connecting to MySQL. If you use a different application server or database server, refer to its documentation.

A JNDI connection on Tomcat is defined in two different files. Make sure both have the following information:

• <tomcat>/webapps/jasperserver[-pro]/META-INF/context.xml

```
<Resource name="jdbc/<db-name>" auth="Container" type="javax.sql.DataSource"
maxActive="100" maxIdle="30" maxWait="10000"
username="<db-user>" password="<db-user-password>"
driverClassName="org.postgresql.Driver"
validationQuery="SELECT 1" testOnBorrow="true"
url="jdbc:mysql://<host>:3306/<database>?autoReconnect=true&amp;autoReconnect
ForPools=true"/>
```

- <tomcat>/webapps/jasperserver-pro/WEB-INF/web.xml

```
<resource-ref>
<description>JNDI Example</description>
<res-ref-name>jdbc/<db-name></res-ref-name>
<res-type>javax.sql.DataSource</res-type>
<res-auth>Container</res-auth>
</resource-ref>
```

Also check the following points:

- Ensure the driver for your database connection is in the <tomcat>/lib folder.
- Ensure the database user has the privileges to run SELECT queries on the tables used in your reports. In some cases, additional permissions may be required to execute stored procedures, depending on your configuration and needs. For more information, see **A.10.4, "Database Permissions," on page 167**.
- If you installed JasperReports Server from a WAR file, Tomcat may have created a separate copy of context.xml in <tomcat>/conf/Catalina/Localhost/jasperserver-pro.xml. See the corresponding section in the troubleshooting appendix of the *JasperReports Server Community Project Installation Guide*.
- See the Apache Tomcat documentation for JNDI data sources.
- For Oracle databases, you may need to specify additional parameters in the context.xml file. For example, in order to support in Oracle, add the following line:

```
driverClassName="oracle.jdbc.OracleDriver"
validationQuery="SELECT 1 FROM dual"
accessToUnderlyingConnectionAllowed="true"
```

A.10.9 JNDI Services on JBoss

After defining JNDI Services on the JBoss application server, JasperReports Server does not automatically detect the new services. To use the new JNDI services as data sources in the server, follow these steps:

1. Define and deploy a JNDI data source in the JBoss administrator console.

   Ensure the database user in your JNDI definition has the privileges to run SELECT queries on the tables used in your reports. In some cases, additional permissions may be required to execute stored procedures, depending on your configuration and needs. For more information, see **A.10.4, "Database Permissions," on page 167**.

2. Modify the file <jboss>/webapps/jasperserver-pro/WEB-INF/web.xml to include a data source reference to this new JNDI service.

3. Modify jboss-web.xml to include a reference to this data source.

4. Because the deployment configuration files such as web.xml were modified, redeploy the JasperReports Server application.

Now you can define JNDI data source in the repository, as described in **4.4, "JNDI Data Sources," on page 55**.

A.10.10 JNDI Services on WebLogic

Follow these steps to configure JasperReports Server to use JNDI data sources with WebLogic:

1. Append the following definition to the `<reference-descriptor>` node of .../WEB-INF/weblogic.xml:

```
<resource-description>
```

```
    <res-ref-name>TestDatabase</res-ref-name>
    <jndi-name>jdbc/testDatabase</jndi-name>
</resource-description>
```

2. Append the following definition to .../WEB-INF/web.xml:

```
<resource-ref>
    <description>TestDatabase database</description>
    <res-ref-name>TestDatabase</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>
```

3. In the WebLogic Admin Console, create a JNDI data source, in this example its JNDI name would be
    `TestDatabase`.

    Ensure the database user in your JNDI definition has the privileges to run SELECT queries on the tables
    used in your reports. In some cases, additional permissions may be required to execute stored procedures,
    depending on your configuration and needs. For more information, see **A.10.4, "Database Permissions," on
    page 167**.

4. Restart the `jasperserver-pro` instance using the WebLogic Admin Console.

A.10.11 Creating a Data Source on SQL Server Using Windows Authentication

If your database is Microsoft SQL Server and you use Windows Authentication (also called Integrated Security),
use the following procedure to create a data source.

1. Go to the [download page for Microsoft SQL Server JDBC Driver 3.0](#).

    Do not use version 4.0 of this driver; it will not work.

2. Download and run the self-extracting executable: 1033\sqljdbc_3.0.1301.101_enu.exe

3. Open the extracted folder sqljdbc_3.0\enu\auth, and then either the x64 or the x86 subfolder, depending on
    whether your version of Windows is 64-bit or 32-bit, respectively.

4. Copy the file sqljdbc_auth.dll to the folder your app server automatically searches for DLLs.

    For Tomcat, this is the <tomcat>\bin folder.

5. Restart your app server.

6. Log into JasperReports Server as an administrator.

7. Right-click the Data Sources folder and select **Add Resource > Data Source** from the context menu.

8. In the Type field, select **JDBC Data Source**. The page refreshes to show the fields necessary for a JDBC
    data source.

9. Enter a name and optional description for your data source.

10. From the dropdown field, select com.microsoft.sqlserver.jdbc.SQLServerDriver.

11. Enter the database hostname and database name of your SQL Server instance.

12. In the URL field, add the following string to the end of the generated URL:

    `;integratedSecurity=true`

13. In the User Name field, enter any non-blank string you want, for example `none`.

14. In the Password field, enter any non-blank string you want, for example `none`.

15. Set the Time Zone and Save Location fields if necessary.

16. Click **Test Connection** and verify that the connection works.

17. Click **Save** to save the data source in the repository.

A.10.12 Upgrading Bean Data Sources

There was a change in the Spring configuration for JasperReports Server 6.0 which changes how some of the existing Spring beans are made accessible for use by other beans. This can break existing custom data sources. This change specifically affects beans which implement the interface `ReportDataSourceServiceFactory`.

Prior to release 6.0, if code in JasperReports Server accessed a bean of this type, it would get the actual instance of the Spring bean as configured in the Spring XML file, and it could be cast to the concrete class. In 6.0 and later, these beans were intercepted in order to implement the profile attributes feature, and instead of seeing the actual instance, other code would see a dynamic proxy instead of the actual bean.

Dynamic proxies are a Java feature which allows classes to be generated at run time that implement any interface that can be loaded on the classpath. The resulting object can be cast to any of those interfaces, but doesn't correspond to any concrete class. For more information:

http://www.javaworld.com/article/2076233/java-se/explore-the-dynamic-proxy-api.html

Since proxies can only represent interfaces, existing code that tries to cast the bean to a concrete class will break. Casting is usually done to get access to methods on a more specific class or interface. As long as the code is not casting the bean to a concrete class, it will work, so there are two ways to get around this problem:

• If the code needs to access methods on an existing interface, just do a cast to that interface, or inject the property using the existing interface, so no cast is needed.
• If the code needs to access methods that are not on an existing interface, simply create an interface with the methods needed, and have the target object implement that interface.

For example, let's say you have a bean with id `myBean` that needs to access the `jdbcDataSourceServiceFactory`, configured as follows:

```
<bean id="jdbcDataSourceServiceFactory" class="com.jaspersoft.jasperserver.
api.engine.jasperreports.service.impl.JdbcReportDataSourceServiceFactory">
...
</bean>
```

Where `myBean` has the following Spring configuration:

```
<bean id="myBean" class="example.MyBean">
    <property name="jdbcDSSF" ref="jdbcDataSourceServiceFactory"/>
```

The following code worked before but will now cause an error:

```
public class MyBean {
  private ReportDataSourceServiceFactory jdbcDSSF;

  public ReportDataSourceServiceFactory getJdbcDSSF() {
    return jdbcDSSF;
  }

  // before 6.0, was called by Spring with the actual bean
  // 6.0 and after, is called with a dynamic proxy
  public void setJdbcDSSF(ReportDataSourceServiceFactory jdbcDSSF) {
    this.jdbcDSSF = jdbcDSSF;
```

```
  }

  public void doSomething() {
    // this code used to work, but now it will break
    ((JdbcReportDataSourceServiceFactory) jdbcDSSF).createService();
    ((JdbcReportDataSourceServiceFactory) jdbcDSSF).doSomethingElse();
  }
}
```

Because the first call is a method which is part of the ReportDataSourceServiceFactory, the cast is unnecessary; to fix it, just leave it out:

```
    jdbcDSSF.createService();
```

In this example, JdbcReportDataSourceServiceFactory has a method called doSomethingElse(). This method is not part of any interface, but you can create an interface that includes it:

```
public interface MyDSSF extends ReportDataSourceServiceFactory {
    public void doSomethingElse();
}
```

JdbcReportDataSourceServiceFactory would need modification so that it implements this new interface:

```
public JdbcReportDataSourceServiceFactory implements MyDSSF {
....
}
```

You don't have to change the declaration in MyBean because Spring will generate a dynamic proxy implementing MyDSSF, but if you change the declaration, the code will be easier to understand because no casts will be necessary:

```
public class MyBean {
  private MyDSSF jdbcDSSF;

  public MyDSSF getJdbcDSSF() {
    return jdbcDSSF;
  }

  // called with a dynamic proxy which implements all needed interfaces
  public void setJdbcDSSF(MyDSSF jdbcDSSF) {
    this.jdbcDSSF = jdbcDSSF;
  }

  public void doSomething() {
    // no need to cast
    jdbcDSSF.createService();
    jdbcDSSF.doSomethingElse();
  }
}
```

A.10.13 SQLFeatureNotSupportedException Using TIBCO Drivers

When you use the TIBCO driver to connect to any database on WebSphere 7 or 8.5, you may see errors such as the following after a query, for example in the Ad Hoc editor. The query still performs correctly:

```
Stack Dump = java.sql.SQLFeatureNotSupportedException:
[DataDirect][Oracle JDBC Driver]Unsupported method:
Statement.setCursorName
```

This is a known error with the WebSphere Adapter(RRA) component. The error message appears in the log file, but it has no impact on functionality. The fix is currently targeted for inclusion in WAS fix pack 8.5.5.1. Please refer to IBM's Recommended Fixes for WebSphere Application Server page for delivery information.

A.11   Special Characters in Database Schemas

Some databases allow you to use spaces and special characters in their database schema. These spaces and symbols can thus appear in table names and column names. However, in order to define Domains, JasperReports Server relies on a restricted set of symbols. Databases having table names and column names with forbidden symbols will cause Domain errors and be unusable with Domains.

As of JasperReports Server 6.0, Domains allow the following symbols:

Symbols Allowed in Table Names	Symbols Allowed in Column Names
`<space> @ # $ % & _ /`	`<space> @ # $ % & ( ) _ [ ] : , + - /`

These symbols have been tested with the following databases:
- SQL Server
- DB2
- PostgreSQL

A.12   Cassandra Reports Not Running

The Cassandra architecture introduces some restrictions that other databases do no have. Certain filters such as is-one-of (IN), <, and > cannot be used with all fields. For example, you may see the following error:

```
Caused by: com.datastax.driver.core.exceptions.InvalidQueryException: Cannot use IN operator
on column not part of the partition key
```

Cassandra clusters use partition keys to split data between nodes, but this imposes a certain segmentation of the data. This segmentation is not compatible with all filtering, because it would be very inefficient. Therefore, the definition of the partition keys in your Cassandra schema limits you to certain comparison operations on certain fields.

In general you should avoid comparison operators other than strict equality in filters on Cassandra data sources. The Cassandra reference documentation recommends *not* using the IN comparison "under most conditions."

For more details, see the Cassandra documentation of the SELECT WHERE clause in CQL (Cassandra Query Language).

A.13   Reverting to the Old Home Page

JasperReports Server 5.5 introduced a new home page with more functionality and access to more features in the server. If your installation or customizations require the old home page with the large buttons, you can specify that the server use the old home page.

1. Open the .../WEB-INF/flows/homeFlow.xml file.

2. Locate the following line:

```
<view-state id="homeView" view="modules/home/home">
```

3. Replace the view value as shown in the following sample:

```
<view-state id="homeView" view="modules/old_home/home">
```

4. Restart the server or redeploy the JasperReports Server web app.

APPENDIX B   LOCALIZATION

By default, JasperReports Server is presented in the English language (US version), but it supports other languages with translations that include data formats and resource bundles. Supported languages are Brazilian Portuguese, Chinese (Simplified), French, German, Italian, Japanese, and Spanish. The translations are included in your JasperReports Server instance by default; to view the application in a specific locale, select it on the login page.

If you need to support a language other than the supported ones, you can localize JasperReports Server, including translating it into a different language by providing labels and messages in the preferred language. For other locales, you may also need to change the default locale and time zone. This chapter describes the procedures and gives a few examples.

For information about localizing Topics and reports, refer to the *JasperReports Server User Guide*.

This chapter contains the following sections:
- **Configuring JasperReports Server for Multibyte Fonts**
- **UTF-8 Configuration**
- **Changing Character Encoding**
- **Creating a Locale**
- **Configuring JasperReports Server to Offer a Locale**

B.1   Configuring JasperReports Server for Multibyte Fonts

Although translation packs for Chinese and Japanese ship with JasperReports Server, the fonts that it uses by default do not support those languages. Therefore, if your organization requires those fonts, you need to configure JasperReports Server for them.

While the fonts JasperReports Server uses are generally dictated by the JRXML files that define your reports, some font configuration is required for special circumstances. For example, you can configure Jaspersoft OLAP to offer different options in the **Chart Default Font** field in the Chart Options dialog. But note that, in order to use a font, the font must be available to the host's operating system. This section describes steps you may need to take, depending on the functionality you use and the locales you support.

The tasks in this section require you to edit these files:

File Name	Location	Purpose of Edits
jpivot_internal_messages.properties	.../WEB-INF/internal	Specifying chart fonts for Jaspersoft OLAP Community
Ja_pro_internal_messages.properties	.../WEB-INF/internal	Specifying chart fonts for Jaspersoft OLAP Professional and Enterprise
userConfig.xml	.../WEB-INF/jpivot/print	Embedding fonts in PDF

B.1.1   Enabling East Asian Fonts

The default configuration of the Java Runtime Environment (JRE) does not support East Asian fonts. If your locale requires such a font, you need to configure your users' computers for the fonts and update their JRE.

To configure a Microsoft Windows computer (XP and later) for East Asian fonts:

Details of this procedure vary, depending on your version of Windows.

1. In the Control Panel, click **Region and Language**.
2. In the Region and Language dialog, select the **Keyboards and Languages** tab.
3. On the tab, install the language(s) that you need.
4. If necessary, install the related keyboard modifications.
5. Close the control panel.
6. Locate the fontconfig.properties.src file in the C:\Program Files\<JRE_directory>\lib folder.
7. In the file, locate the following line:

   ```
   sequence.allfonts=alphabetic/default,dingbats,symbol
   ```
8. Change the line to include the East Asian fonts that you need, such as the following:

   ```
   sequence.allfonts=alphabetic/default,dingbats,symbol,korean,japanese,chinese-
   ms936,chinese-ms950
   ```
9. At the end of the file, check to be sure that the fonts you selected are listed, as in the following:

   ```
   filename.Gulim=gulim.TTC
   ```

   If the fonts are not listed, add them.
10. Save and close the file.
11. Rename the file to fontconfig.properties in the file system.

B.1.2   Configuring Ad Hoc Charts for Asian Fonts

The default font for the legend of an Ad Hoc chart does not support some Asian characters, such as Japanese kana glyphs. Default fonts for Ad Hoc reports are defined in the following file:

Changing Ad Hoc Chart Legend Fonts		
Configuration File		
.../WEB-INF/adhoc/themes/default.new.jrxml		
Property	**Value**	**Description**

`style name=`
`    "ChartLegend"`
`fontName=`
	"DejaVu Sans"
	<default>
	
	"SansSerif"
		The default value, DejaVu Sans, does not include kana glyphs, and thus the legend contains blank symbols.
		
		If the server machine has system Japanese fonts available, setting the value to SansSerif should work in most Java font configurations.

> After making changes to the default.new.jrxml file, you must rerun all Ad Hoc reports that contain Japanese characters for them to appear.

If Japanese fonts are not installed in the server machine, you can add it in the following manner:

1. Package a Japanese font as a font extension JAR.
2. Add the new JAR to the .../WEB-INF/lib folder of the JasperReports Server WAR file.
3. Edit the default.new.jrxml file as described above to specify the new font name.
4. Redeploy JasperReports Server or restart its app server.
5. Rerun the affected reports. Reports saved from Ad Hoc should now display correctly in the Report Viewer, but the corresponding view may not display chart legends correctly within the Ad Hoc Editor.

B.1.3   Configuring OLAP Options for Chart Default Fonts

If you implement Jaspersoft OLAP and support a locale with special font requirements, you can configure Jaspersoft OLAP to offer different options in the **Chart Default Font** field in the Chart Options dialog of the OLAP view. This may be necessary if you implement locales that Latin 1 doesn't support.

An OLAP view's Chart Options dialog includes the **Chart Default Font** field, which allows users to select the font to use in charts. The default options are SansSerif, Serif, and MonoSpaced. JasperReports Server reads these values from a properties file and attempts to map them to fonts available in the server host's operating system. You can configure the server to offer different fonts if these fonts don't support the locales you implement.

To change the Chart Default Font field's options:

1. Save the jpivot_internal_messages.properties file with a new name that reflects the new locale. For example, for Japanese, the new file would be called jpivot_internal_messages_ja.properties.

2. Open the new file and locate the following keys:

```
JAJ_000_jsp.jpivot.chartpropertiesform.sansSerif=SansSerif
JAJ_000_jsp.jpivot.chartpropertiesform.serif=Serif
JAJ_000_jsp.jpivot.chartpropertiesform.monospaced=Monospaced
```

> If you are using Jaspersoft OLAP Community Edition, the name of the file and the keys that you edit are different. For the Community Edition, open the jpivot_internal_message.properties file and edit these keys:
>
> ```
> jsp.wcf.chart.sansserif=SansSerif
> jsp.wcf.chart.serif=Serif
> jsp.wcf.chart.monospaced=Monospaced
> ```

3. Change one or more of the strings to the name of a font available in the host's operating system. For example, if you wanted to change the SansSerif font to the SimHei font, edit the value specified by `jsp.wcf.chart.sansserif`. For example:

```
jsp.wcf.chart.sansserif=SimHei
```

4. Save the file.
5. Restart JasperReports Server.

B.1.4 Embedding Fonts in PDF Output

> By default, JasperReports Server can create PDF (Portable Document Format) files with many different fonts. However, if you experience font problems in the PDF output, you may need to take the steps described in this section to make the fonts available to JasperReports Server's XSL Formatting Object (XSL-FO) processor.
>
> You must have distribution rights to a font in order to embed it in a PDF file.

When users save reports in PDF format, JasperReports Server generates the PDF output using Apache FOP (Formatting Objects Processor). In order for FOP to render fonts properly, you must install the font itself (for example, a TTF file) on the server host, create a font metrics file (using Apache's `org.apache.fop.fonts.apps.TTFReader` utility), and update the userConfig.xml file to associate the font with its metrics. For more information, refer to the Apache FOP documentation.

You can embed any Unicode font using this procedure, though larger font files may have significantly larger memory footprints. To keep memory requirements small, we recommend you use the smallest font file you can, such as SimHei to support Chinese, Japanese, and Korean.

B.2 UTF-8 Configuration

JasperReports Server uses UTF-8 (8-bit Unicode Transformation Format) character encoding. If your database server or application server uses a different character encoding form, you may have to configure them to support UTF-8. This section provides information for configuring the character encoding for several application servers and database servers. If you use a different application server or database, and its default character encoding isn't UTF-8, you may need to make similar updates to support certain locales. For more information, refer to the documentation for your application server or database.

B.2.1   Tomcat

By default, Tomcat uses ISO-8859-1 (ISO Latin 1) character encoding for URIs, which is sufficient for Western European locales, but does not support many locales in other parts of the world.

If you plan to support locales that Latin 1 does not support, you must change Tomcat's URI encoding format.

> If you chose the instance of Tomcat that is bundled with the installer, you don't need to make this change. The bundled Tomcat is preconfigured to support UTF-8. If you installed the WAR file distribution with your own instance of Tomcat and want to support UTF-8, follow this procedure.

To configure Tomcat to support UTF-8:

1. Open the conf/server.xml file and locate the following code:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector>
    port="8080" maxHttpHeaderSize="8192"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" redirectPort="8443" acceptCount="100"
    connectionTimeout="20000" disableUploadTimeout="true"
</Connector>
```

2. At the end of this section, insert the following line before the closing tag:

```
URIEncoding="UTF-8"
```

3. For example, after your changes, the section might read:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
    <Connector>
    port="8080" maxHttpHeaderSize="8192"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" redirectPort="8443" acceptCount="100"
    connectionTimeout="20000" disableUploadTimeout="true"
    URIEncoding="UTF-8"
</Connector>
```

4. Save the file.
5. Restart Tomcat.

B.2.2   JBoss

Because JBoss uses Tomcat as its web connector, the configuration changes in **B.2.1, "Tomcat," on page 181**) also have to be made for JBoss. The only difference is that the server.xml file is located in the Tomcat deployment directory, typically server/default/deploy/jbossweb-tomcat55.sar. Make the same configuration changes, then restart JBoss.

B.2.3   PostgreSQL

JasperReports Server requires PostgreSQL to use UTF-8 character encoding for the database that stores its repository as well as for data sources. A simple way to meet the requirement is to create the database with a

UTF-8 character set. For example, enter the following command:

```
create database jasperserver encoding='utf8';
```

B.2.4   MySQL

By default, MySQL uses ISO-8859-1 (ISO Latin 1) character encoding. However, JasperReports Server requires MySQL to use UTF-8 character encoding for the database that stores its repository as well as for data sources. The simplest way to meet the requirement is to create the database with a UTF-8 character set. For example, enter the following command:

```
create database jasperserver character set utf8;
```

To support UTF-8, the MySQL JDBC driver also requires that the `useUnicode` and `characterEncoding` parameters be set as in this startup URL:

```
url="jdbc:mysql://localhost:3306/jasperserver?useUnicode=true&characterEncoding=UTF-8"
```

If the MySQL database is a JNDI data source managed by Tomcat, such as the JasperReports Server repository database, the parameters can be added to the JDBC URL in .../WEB-INF/context.xml. The following is a sample resource definition from that file:

```
<Resource name="jdbc/jasperserver" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    username="root" password="password" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/jasperserver?useUnicode=true&amp;characterEncoding=UTF-8" />
```

JBoss ignores the context.xml file, instead requiring an XML file to define JNDI data sources in the deployment directory, which is typically server/default/deploy. The following is an example of a resource definition in one of those XML files:

```
<local-tx-datasource>
    <jndi-name jdbc/jasperserver />
    <connection-url>
        jdbc:mysql://localhost/jasperserver?useUnicode=true&amp;characterEncoding=UTF-8
    </connection-url>
    <driver-class com.mysql.jdbc.Driver />
    <user-name jasperadmin />
    <password jasperadmin />
    <min-pool-size 5 />
    <max-pool-size 20 />
    <idle-timeout-minutes 0 />
    <metadata>
        <type-mapping mySQL />
    </metadata>
</local-tx-datasource>
```

If the database is a JDBC data source configured in the repository, change the JDBC URL by editing the data source in the JasperReports Server repository. The following is an example of the JDBC URL (note that the ampersand isn't escaped):

```
jdbc:mysql://localhost:3306/foodmart_ja?useUnicode=true&characterEncoding=UTF-8
```

B.2.5   Oracle

Oracle databases have both a default character set and a national character set that supports Unicode characters. Text types beginning with N (NCHAR, NVARCHAR2, and NCLOB) use the national character set. All the text data used by the JasperReports Server repository (when stored in Oracle) is stored in NVARCHAR2 columns, so that JasperReports Server metadata can use the full Unicode character set. For more information about Unicode text support, refer to the Oracle white paper (PDF).

To work properly with Unicode data, the Oracle JDBC driver requires you to set a Java system property by passing the following argument to the JVM:

```
-Doracle.jdbc.defaultNChar=true
```

In Tomcat, add the variable to JAVA_OPTS in bin/setclasspath.sh (Linux) or bin/setclasspath.bat (Windows):

1. Locate the following line in the script:

| Linux | # Set the default -Djava.endorsed.dirs argument |
| Windows | rem Set the default -Djava.endorsed.dirs argument |

2. Add the following line before it:

| Linux | JAVA_OPTS="$JAVA_OPTS "-Doracle.jdbc.defaultNChar=true |
| Windows | set JAVA_OPTS=%JAVA_OPTS% -Doracle.jdbc.defaultNChar=true |

Because JBoss also uses JAVA_OPTS to pass options to the JVM, you can add the same line to bin/run.sh (Linux) and bin/run.bat (Windows). Add it before this line:

| Linux | # Setup the java endorsed dirs |
| Windows | rem Setup the java endorsed dirs |

B.3   Changing Character Encoding

Depending on the third-party software you use and the locales you support, you may also have to configure JasperReports Server and its host. The steps described in this section are necessary only under certain circumstances, such as if you plan to use a character encoding form that UTF-8 cannot handle.

To use a character encoding form other than UTF-8, you must configure JasperReports Server, your application server, and your database server.

B.3.1   Configuring JasperReports Server

To specify a different character encoding:

1. Open the .../WEB-INF/applicationContext.xml file and locate the following bean. It's configured for UTF-8:

```
<bean id="encodingProvider"
      class="com.jaspersoft.jasperserver.api.common.util.
      StaticCharacterEncodingProvider">
   <constructor-arg value="UTF-8" />
</bean>
```

2. Change "UTF-8" to the encoding type your database server and application server use. For example:

```
<bean id="encodingProvider"
      class="com.jaspersoft.jasperserver.api.common.util.
      StaticCharacterEncodingProvider">
  <constructor-arg value="UTF-16" />
</bean>
```

3. Save the file.
4. Restart JasperReports Server.

B.3.2   Configuring the Application Server and Database Server

If you want to use a character encoding other than UTF-8, you may need to configure the third party software that JasperReports Server relies on. For more information, refer to the documentation for your application server and database server. For Tomcat, you can specify a different character encoding by following steps similar to those described in **B.2.1, "Tomcat," on page 181** and **B.2, "UTF-8 Configuration," on page 180**.

B.3.3   Configuration for Localized Analysis Schemas

If you plan to use localized OLAP views, you must take additional steps to configure JasperReports Server.

To configure JasperReports Server for localized OLAP views:

1. Every Unicode database that JasperReports Server interacts with (whether it's the repository database or a database accessed through a data source defined in JasperReports Server) must be created to support UTF-8. For example, to create the Foodmart database in PostgreSQL, you might give a command similar to the following:

```
create database foodmart_ja encoding='utf8';
```

2. The URL of any OLAP data source that JasperReports Server accesses must be properly configured in the .../META-INF/context.xml file. For example, the URL definition for the Foodmart sample database might be similar to the following:

```
<Resource name="jdbc/MondrianFoodMart_ja"
    auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    username="postgres" password="postgres" driverClassName="org.postgresql.Driver"
    url="jdbc:postgresql://localhost:5432/foodmart_ja" />
```

3. Encoding options must be added to the JDBC connection string for any data source that points to an OLAP database. For example, when creating a data source in JasperReports Server that points to an OLAP database, use the following connection string:

```
jdbc:postgresql://localhost:5432/foodmart_ja
```

B.4   Creating a Locale

When you want to create other locales for JasperReports Server, translation is only one aspect of localization. Creating a locale includes these tasks:

• Translating labels and messages

The tasks in this section require you to edit these files:

File	Location	Purpose of Edits
*.properties files	.../WEB-INF/bundles <js-install>/buildomatic/conf_ source/iePro/bundles	Translating labels and messages
jasperserver_ config.properties	.../WEB-INF/bundles	Changing date formats

B.4.1   About Properties Files

Resource bundles for JasperReports Server and Jaspersoft OLAP are Java properties files found in the .../WEB-INF/bundles directory. The properties files contain all the labels and messages used in JasperReports Server and Jaspersoft OLAP.

A bundle includes a default locale (for example, jasperserver_messages.properties), which is written in U.S. English. Then it consists of all the properties files with the same base name, but different locale (such as jasperserver_messages_fr.properties). Each file translates all of the strings of the default file into the language given by the locale. The Java programming language has rules for specifying locales and alternate locales and determining which locale in the bundle to use.

Default Resource Bundles in JasperReports Server	
File in .../WEB-INF/bundles	**Description**
accessibility_messages.properties	Text spoken in accessibility software such as screen readers.
AttributeBundle.properties	Messages for attribute validation.
AttributeBundles.properties	Messages for attribute dialogs.
calendar.properties	Labels and messages used by the pop-up calendar dialog.
.properties	.
CommonBundle.properties	Bundle for tests.
HomeBundle.properties	Labels for the new home page introduced in version 5.5.
image_descriptions_messages .properties	Labels and messages for the AWS (Amazon Web Services) machine images.
jasperreports_highcharts_ messages.properties	Messages for charts in the report viewer.

Default Resource Bundles in JasperReports Server	
File in .../WEB-INF/bundles	**Description**
jasperreports_messages .properties	Labels and messages for the report viewer.
jasperserver_config.properties	Configuration properties for dates and date-times.
jasperserver_messages.properties	Labels and messages used in the main JasperReports Server user interface.
jsexceptions_messages.properties	Messages used in errors and exceptions, both in the UI and in the log messages.
LicenseMessages.properties	Labels and messages used when validating licenses.
logger_descriptions.properties	Internal log messages.
pro_nav_messages.properties	Labels and messages for the menu bar and old Home page.
ScalableInputControlsBundle .properties	Labels for lists of values in input controls.
scheduling_ws.properties	Validation messages for the report scheduler.
security.properties	Messages used by input validation.
ValidationBundle.properties	Messages for dashboard dialogs.

Default Resource Bundles in Jaspersoft OLAP	
File in .../WEB-INF/bundles	**Description**
ja_mondrian.properties	Labels and messages in the OLAP settings UI.
ja-pro_messages.properties	Labels and messages in the commercial edition OLAP viewer.
jpivot_messages.properties	Labels and messages in the community edition OLAP viewer.
mondrian_exception_messages .properties	MDX validation error messages specific to the internal analysis engine.

The standalone import and export tools have their own bundles located outside of the web app. The following bundles are located in the WAR file distribution package, under the buildomatic folder.

Default Resource Bundles for js-export.sh and js-import.sh	
File in buildomatic/conf_ source/ieCe/bundles	**Description**
ji-export-messages.properties	Labels and messages for `js-export --help`.
jsexceptions_messages.properties	Messages used in errors and exceptions, both in the UI and in the log messages.

If you use the JasperReports Server portlet to display JasperReports Server content in a portal (such as Liferay), the deployed portlet includes properties files, as well:

File	**Location**	**Description**
jaspersoft_portlet_ message.properties	.../WEB-INF/bundles under the location where the portlet is deployed. For example: C:\liferay\webapps\<portlet_context_ name>\WEB-INF\bundles (where <portlet_ context_name> is the name specified when the JasperReports Server portlet was deployed).	Labels and messages that appear in the portlet, including the help text. Note that this does not include text in specific reports; to localize the report's content, you must upload its resource bundle to the repository.

B.4.2   Creating a Resource Bundle

Create a resource bundle by making a copy of each *.properties file, using the following syntax for the copy's file name:

```
<default_file_name>_<locale>.properties
```

where

`<default_file_name>` is the name of the default version of the properties file, and

`<locale>` is a Java-compliant locale identifier.

For example, consider the core JasperReports Server resource bundle. For various locales, it might be named as follows:

Language	**File Name**
Default (English)	jasperserver_messages.properties
French	jasperserver_messages_fr.properties
French in Switzerland	jasperserver_messages_fr_CH.properties

For a list of Java-compliant locales, please refer to the Java documentation.

The resource bundles described in this document consist of locale-specific Java properties files. Java properties files use the ISO-8859-1 (Latin-1) encoding that is the same as ASCII for all English non-accented characters. For international characters that are not in ISO-8859-1, use Unicode escape sequences (for example `\u00e9` is é).

To create a new JasperReports Server resource bundle:

1. Copy each of the properties files (keeping them in the same directory as the originals) and rename them according to your locale.

2. Translate each *.properties file's labels and messages into the new language.

   Some of the strings in the properties files are date formats and format masks that may need to be edited for the new locale. For more information, refer to **B.4.3, "Setting Date and Datetime Formats," on page 188**.

3. Save the files.

4. If the new locale requires specific character encoding or fonts, ensure that JasperReports Server and the third party software it relies on are configured to support them. For more information, refer to **B.1, "Configuring JasperReports Server for Multibyte Fonts," on page 177**.

The new locale is not available in JasperReports Server until you follow the steps described in **B.5.1, "Specifying Additional Locales," on page 189**.

B.4.3   Setting Date and Datetime Formats

Each locale may have its own rules for displaying dates and datetime values. System date and datetime formatting is controlled by four patterns that are specified in the jasperserver_config_<locale>.properties file associated with a particular locale.

For example in the English resource bundle, the four entries are:

```
date.format=dd-MM-yyyy
datetime.format=dd-MM-yyyy HH:mm
calendar.date.format=%d-%m-%Y
calendar.datetime.format=%d-%m-%Y %H:%M
```

The first two keys are used to parse and format dates and datetime values using an internal `java.util.DateFormat` object across the whole application. These patterns should be non-localized date patterns, in accordance with the Java Development Kit (JDK) syntax.

The other two keys are used by the calendar control, which formats the user-selected date and datetime values in accordance with its own pattern syntax.

To change the system date and datetime formatting for a new locale, edit the strings specified by these keys.

B.5   Configuring JasperReports Server to Offer a Locale

After creating a locale, you must configure JasperReports Server to offer it to your users, along with any new time zones.

The tasks in this section require you to edit these files:

File Name	Location	Purpose of Edits
applicationContext-security.xml	WEB-INF	Specifying additional locales
jasperserver-servlet.xml	WEB-INF	Specifying additional time zones

B.5.1   Specifying Additional Locales

By default, JasperReports Server appears in the locale selected in the end user's browser. The Login page allows users to specify the locale they want to use. The list of available locales is defined in applicationContext-security.xml. Edit this file to add a new locale.

To add a new locale:

1.   Edit the applicationContext-security.xml file and locate the bean named `userLocalesList`. For example:

```
<bean id="userLocalesList"
        class="com.jaspersoft.jasperserver.war.common.LocalesListImpl">
    <property name="locales">
        <list>
            <value type="java.util.Locale">en</value>
            <value type="java.util.Locale">fr</value>
            <value type="java.util.Locale">it</value>
            <value type="java.util.Locale">de</value>
            <value type="java.util.Locale">ro</value>
            <value type="java.util.Locale">ja</value>
            <value type="java.util.Locale">zh_TW</value>
        </list>
    </property>
</bean>
```

2.   Add the new locale to the end of the list. For example, add the following line for Dutch (Java's nl_NL locale):

```
            <value type="java.util.Locale">nl_NL/value>
```

3.   Save the file.
4.   Restart JasperReports Server, and log into the web application to test your translation. Reviewing the translated strings in context can help you improve your word choices.

For a list of Java-compliant locales, please refer to the Java documentation.

B.5.2   Specifying Additional Time Zones

By default, JasperReports Server assumes the user's time zone is that of the JasperReports Server host. The Login page allows users to choose a different time zone. The available list is defined in applicationContext.xml file.

To add a time zone:

1.   Open the applicationContext.xml file and locate the `userTimeZonesList` bean. For example:

```
<bean id="userTimeZonesList"
      class="com.jaspersoft.jasperserver.war.common.JdkTimeZonesList">
   <property name="timeZonesIds">
      <list>
         <value>America/Los_Angeles</value>
         <value>America/Denver</value>
         <value>America/Chicago</value>
         <value>America/New_York</value>
         <value>Europe/London</value>
         <value>Europe/Berlin</value>
         <value>Europe/Bucharest</value>
      </list>
   </property>
</bean>
```

2. Add the new time zone to the bottom of the list. Specify each time zone as the standard Java time zone values so that JasperReports Server adjusts for daylight savings time when appropriate. For example, add the following line for Tokyo:

```
<value>Asia/Tokyo</value>
```

3. Save the file.
4. Restart JasperReports Server.

For more information about Java-complaint time zones, please refer to the Java documentation.

B.5.3   Setting a Default Time Zone

If you want JasperReports Server to use a time zone other than the host computer's, you can set a specific time zone in Java. It becomes the default time zone for all users, but they can still select a different time zone when they log in.

To set a default time zone, set the `user.timezone` property in the JVM as shown in the tables below. Locate the file containing JVM settings for your platform and application server. The value for the property must be a Java-compliant time zone, for example, `Europe/Bucharest`.

You must restart your application server for this setting to take effect. The time zone is set for all applications in your application server, including JasperReports Server.

JVM Settings for Default Time Zone			
Operating System	**App Server**	**File**	**Setting**

Windows	Tomcat	\<apache-tomcat\>\bin\setenv.bat	Add this line of code:
			`set JAVA_OPTS=%JAVA_OPTS%`
			
			`-`
			`Duser.timezone=<timezon e>`

	JBoss	\<jboss\>\bin\run.bat	

JVM Settings for Default Time Zone			
Operating System	**App Server**	**File**	**Setting**

Linux	Tomcat	<apache-tomcat>/bin/setenv.sh	Add this line of code:
			`export JAVA_OPTS="$JAVA_`
			`OPTS`
			`-`
			`Duser.timezone=<timezon`
			`e>"`

	JBoss	<jboss>/bin/run.sh	

Both	GlassFish	<glassfish>/domains/domain1/config/domain.xml	Add this line of code to the <jvm-options> section:
			`-`
			`Duser.timezone=<timezon`
			`e>`

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

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.

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.

Frame can also refer to an element in a legacy dashboard; it's the equivalent of a dashlet.

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.

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.

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 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

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.

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.

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.

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