# JasperReports® Server Data Management Using Domains

*Software Release 8.2*

JASPERSOFT™

**Important Information**

# TABLE OF CONTENTS

# CHAPTER 1    INTRODUCTION TO JASPERREPORTS SERVER

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

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

The heart of the 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. See the *JasperReports Server User Guide* and the *Jaspersoft Studio User Guide* for more information.

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. See the *JasperReports Server User Guide* and the *Jaspersoft Studio User Guide* for more information.
- If you have a subscription to our professional support offerings, please contact our Technical Support team when you have questions or run into difficulties. They're available on the web at and through email at http://support.tibco.com and js-support@tibco.com.

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

# CHAPTER 2     UNDERSTANDING DOMAINS

Domains model and present data that report creators may not understand in raw form. Production databases typically contain data in tables optimized for storage and retrieval. A Domain is a virtual view, created and stored in JasperReports Server without modifying the data source, that models and presents the data in a way that is more accessible to report creators and readers. Columns with data relevant to users can be joined across several tables, filtered by business need, and augmented with calculated fields.

A Domain is a metadata layer that models and presents data accessed through a data source. You can use Domains to structure the data and present it in business terms appropriate to your audience. You can also limit access to data based on the security permissions of the person running the report and provide translations for static report text. Domains in JasperReports Server are used to create Ad Hoc views and reports.

By extension, a Domain is also the dataset that it selects and makes available in a more usable form. When creating a Domain, you define the view and the metadata that selects and presents the data that you want. When using a Domain, users see the data you've made available and create Ad Hoc reports with it.

## 2.1     Who Uses Domains?

Domains help make data accessible to different types of users:

- Database administrators or business analysts create Domains, choosing and modeling the data to expose, and creating user-friendly labels where appropriate. These users have an understanding of the raw data as it appears in the data source.

  > In the default server installation, Domain creators must have organization-level admin privileges. See *JasperReports Server Administrator Guide* for more information.

- Report designers use Domains to create Ad Hoc views in JasperReports Server or reports in Jaspersoft Studio. These users understand the data in the context of their specific business case, but might not understand the raw data in the original data source.
- Report readers see only the data they need and understand.

## 2.2     What Does a Domain Do?

A Domain has multiple levels that allow you to model and present data:

- Select schemas and tables from your data source on the Data Management tab, as well as create derived tables using SQL queries. See **3.3, "The Data Management Tab," on page 19** and **4.2, "Derived Tables," on page 42** for more information.
- Define joins between any included tables and/or derived tables on the Joins tab. See **3.4, "The Joins Tab," on page 23** for more information.
- Set conditions on field values to limit the data accessed through the Domain on the Pre-filters tab. See **3.5, "The Pre-filters Tab," on page 28** for more information. You can also filter data using calculated fields; see **4.3, "Calculated Fields," on page 43** for more information.

> A Domain design often includes data filtering, but report creators can filter data even further or create input controls for report readers to use.

- Choose which tables and columns to expose to users on the Data Presentation tab, and optionally define or change display properties. See **3.6, "The Data Presentation Tab," on page 30** for more information.
- Create or upload a security file that defines access permissions on the Security tab. See **3.7, "The Security Tab," on page 37** for more information.
- Upload locale bundles for the display names on the Locales tab. See **3.8, "The Locales Tab," on page 39** for more information.

Each Domain is stored in the repository as an XML design file along with the files the Domain references, such as a data source, locale bundles for localization, or a security file. You can create Domains in the Domain Designer, or you can upload a Domain design file in XML format. Security files can be created similarly in the Domain Designer or uploaded as an XML file. Localization bundles must be created in text format and uploaded. See **Chapter 8, "Securing Data in a Domain," on page 111**, and **Chapter 9, "Localizing Domains," on page 129** for more information.

## 2.3    Planning a Domain

Plan Domain features before you begin the creation process:
- Decide on the data source and define it in the server repository.
- If you need to combine several data sources into a single data source, choose your data virtualization strategy. Either define each data source in the repository and then create a virtual data source that combines them, or use Jaspersoft Advanced Data Services and define a data source based on that. See **4.1, "Domains and Data Virtualization," on page 41** for more information.
- If you want to use attributes, make sure they are defined and that you understand how they will work. This is especially important if you are using attributes for schema names. See **4.5, "Using Attributes in the Domain Designer," on page 53** for more information.
- Know the elements of your design: tables and columns to choose, joins to perform, pre-filters to define, and items you want to present to users.
- Determine whether the users of the Domain need localized resources, such as translated labels and local formats such as dates.
- Determine a security policy for the data retrieved through the Domain. Is there any data that should be restricted to only certain users or roles in your server?

## 2.4     Data Security

Domains may optionally define permissions that control access to data based on user names, roles, and attributes existing in the server. This allows you to create one Domain for all users, with access to sensitive data restricted by these permissions. Users may run the same report based on the Domain, but they will see only the data that they are allowed to access.

Permissions can be set separately on the data's columns and rows. In Domains, columns display the items in the Domain; rows display the values of each item. A user can see results only where they have both column- and row-level access. When a user is designing a report in the Ad Hoc Editor, they see only the columns to which they have access. When the report runs, portions to which the user has no access are blank.

Data security for a Domain is defined in a single security file. The file is attached to the Domain as a resource, as described in **3.7, "The Security Tab," on page 37**. The security file references the `ids` of tables, columns, sets, and items in the Domain design. For more information, see **Securing Data in a Domain**.

# CHAPTER 3    WORKING WITH THE DOMAIN DESIGNER

This chapter covers the process of creating a Domain and defining its contents through the Domain Designer. It covers the basic functionality on the various tabs of the user interface. For complex joins, derived tables, and calcuated fields, see **Chapter 4, "Advanced Domain Topics," on page 41**.

For instructions about creating views based on Domains in the Ad Hoc Editor, see the *JasperReports Server User Guide*. Domains defined in the server can be accessed through Jaspersoft Studio as well.

This chapter contains the following sections:

- **Opening the Domain Designer**
- **Overview of the Domain Designer**
- **The Data Management Tab**
- **The Joins Tab**
- **The Pre-filters Tab**
- **The Data Presentation Tab**
- **The Security Tab**
- **The Locales Tab**

## 3.1    Opening the Domain Designer

**To open the Domain Designer and create a new Domain:**

1. Log in as an administrative user.
2. Create a Domain in one of the following ways.
   - Select **Create > Domain** from the main menu.
   - Click **Create** in the Domains section of the home page.
   - Right-click a repository folder and choose **Add Resource > Domain**.
   
   The Choose Data dialog appears.
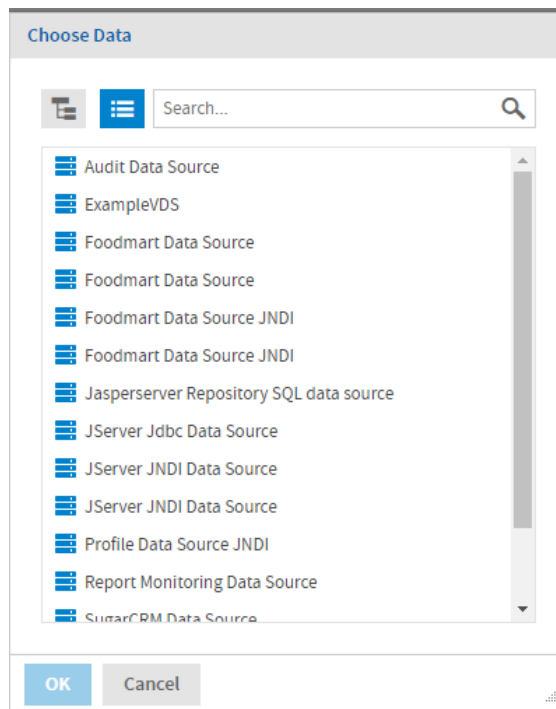
**Figure 3-1  The Choose a Data Source dialog**

3.  Select a data source in the Choose Data dialog and click **OK**.
    - Use only data sources in the repository for which the intended user has at least execute permission.
    - If you installed the sample data, you can use the sample JDBC, JNDI, and virtual data sources in **Organization > Data Sources** or **Organization > Analysis Components > Analysis Data Sources**.
    - You can use the icons at the top of the Choose Data dialog to switch between a repository tree view and a flat list of data sources:

      ⊟ – Display data sources as a repository view.

      ☰ – Display data sources as a list.
    - For performance reasons, the Choose Data dialog has an upper limit on the number of data sources it displays. If the data source you want does not appear in the list, use the **Search...** bar 🔍 to locate it.
4.  Click **OK** to open the Domain Designer with the selected data source.

**To edit an existing Domain:**
1.  Log in as an administrative user.
2.  Display a list of Domains in one of the following ways:
    - Navigate to a folder in the Repository that contains Domains.
    - Click the large icon in the Domains block on the home page.
3.  Right-click the Domain you want and select **Edit** from the menu.

    The Domain Designer displays the Data Management tab for the Domain you selected.

If you close the domain without saving the changes, you are prompted to save or cancel the changes to the existing domain.

## 3.2    Overview of the Domain Designer

The Domain Designer is a tool for defining the components of a Domain. Tabs let you choose schemas and tables in your data source, join tables, pre-filter data, and choose the data to present to report creators and users. You can also create calculated fields and derived tables, and upload text files to translate static text or apply security to the Domain.
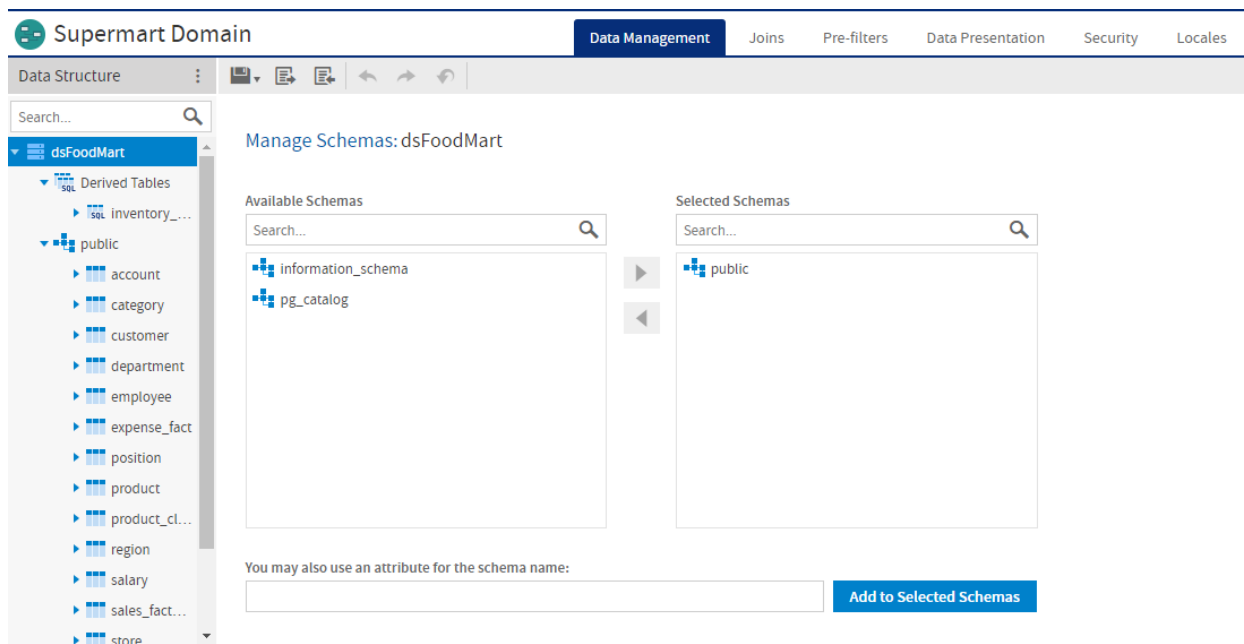


**Figure 3-2**  **User Interface of the Domain Designer**

### 3.2.1    Domain Designer Tabs

Use the tabs at the top of the Domain Designer to view and edit various aspects of the design. To navigate among tabs, click a tab name at the top of the Domain Designer:

- Data Management tab – Select schemas and tables you want to use in the Domain, including tables you refer to but might not want to expose. See **3.3, "The Data Management Tab," on page 19** for more information.
- Joins tab – Define joins between any included tables and/or derived tables. See **3.4, "The Joins Tab," on page 23** for more information.
- Pre-filters tab – Set conditions on field values to limit the data accessed through the Domain. See **3.5, "The Pre-filters Tab," on page 28** for more information.
- Data Presentation tab – Create sets that specify tables, columns, and items to expose to users. Optionally define or change display properties, such as names and descriptions. See **3.6, "The Data Presentation Tab," on page 30** for more information.
- Security tab – Create or upload files a security file. See **3.7, "The Security Tab," on page 37** for more information.

- Locales tab – Upload locale bundles for translating elements of the Domain. See **3.8, "The Locales Tab," on page 39** for more information.

All tabs except the Security and Locales have at least two panels, from left to right:

- **Data Structure** panel, which displays the schemas, tables, columns, and joins available on the current tab. Use the search bar at the top of the Data Structure tab to locate a Domain element. Not all elements are available on all tabs.
- A design panel on the right, with a working area specific to the current task. In some cases, the design panel is divided into sub-panels.

### 3.2.2    Domain Designer Tool Bar

Use the icons on the tool bar to create derived tables and calculated fields, change the data source, save the Domain, and import and export Domain design files.

**Table 3-1  Domain Designer Tool Bar Icons**

| Icon | Name | Description |
|------|------|-------------|
| | More | Click to choose one of the following: <br> • Create a global constant field <br> • Create a derived table <br> • Choose a different data source or reload data from your existing data source <br> • Clear all data in the Domain |
| | Save | Place the cursor over this icon to display a menu of options for saving the Domain. |
| | Export | Click this icon to export a Domain design file. |
| | Import | Click this icon to import a Domain design file. |
| | Undo | Click this icon to undo the most recent action. |
| | Redo | Click this icon to redo the most recently undone action. |
| | Undo All | Click this icon to revert the Domain to its state when you last saved. |
| | Close Domain | Click this icon to close the Domain and return to the previous screen. |

Before you can save the design, you must add at least one table to the Domain. The ![save icon] button on the tool bar validates the design and saves it in the location you specify. For more information, see **"Domain Validation" on page 60**. After saving a Domain, you can continue modifying it using the Domain Designer.

### 3.2.3    The Data Structure Panel

The Data Structure panel contains a hierarchical view of the available schemas, tables, columns, and joins. The available elements and the layout depend on the current tab. A search bar lets you search for items in the panel by

name.
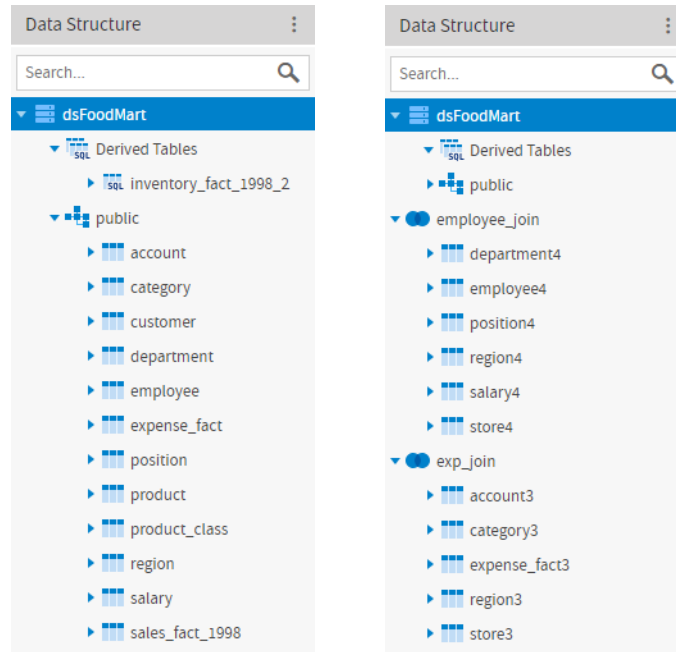


**Figure 3-3  Examples of Data Structure panel on Data Management and Joins tabs**

The following icons to indicate element type can appear on the Data Structure panel or elsewhere in the Domain Designer:

| Icon | Description |
|---|---|
|  | The data source for the Domain. There is only one data source for a Domain. |
|  | A database schema that has been added to the Domain. |
|  | A database schema that has been added using an attribute for the schema name. |
|  | A table that has been added to the Domain. |
|  | A column (field) in the Domain. |
|  | Constant Fields. Only appears if you have created one or more constant calculated fields. |
|  | A calculated field or constant field that has been added to the Domain. Constant fields appear under the Constant Fields node; other calculated fields appear under the join or table where they were created. |

| Icon | Description |
|---|---|
| | Derived Tables. Only appears if you have created one or more derived tables. |
| | A derived table that has been added to the Domain. |
| | A join tree. A Domain may contain many join trees, but when a user creates an Ad Hoc view from a Domain, they can only select one join tree to use in the view. |
| | A data island. A data island contains the tables and columns from a specific join tree that you have chosen to expose to users. Data Islands are only visible on the Data Presentation tab. |
| | A set, that is, a named collection of items grouped together for ease of use in the Ad Hoc Editor. Sets are visible only on the Data Presentation tab. |
| | An item, that is, the representation of a database field or a calculated field along with its display name and formatting properties. Items are visible only on the Data Presentation tab. Items can be grouped in sets and are used in the creation of Ad Hoc views. |

The following icons represent items on the Data Presentation tab. An item is the representation of a database field or a calculated field along with its display name and formatting properties. Items can be grouped in sets and are used in the creation of Ad Hoc views.

| Icon | Description |
|---|---|
| | Boolean field |
| | Boolean measure |
| | calculated field |
| | calculated measure |
| | date field |
| | date measure |
| | numeric field |
| | numeric measure |
| | string field |
| | string measure |

## 3.3      The Data Management Tab

The Data Management tab is where you manage the schemas and tables from your data source and choose which ones to include in your Domain.
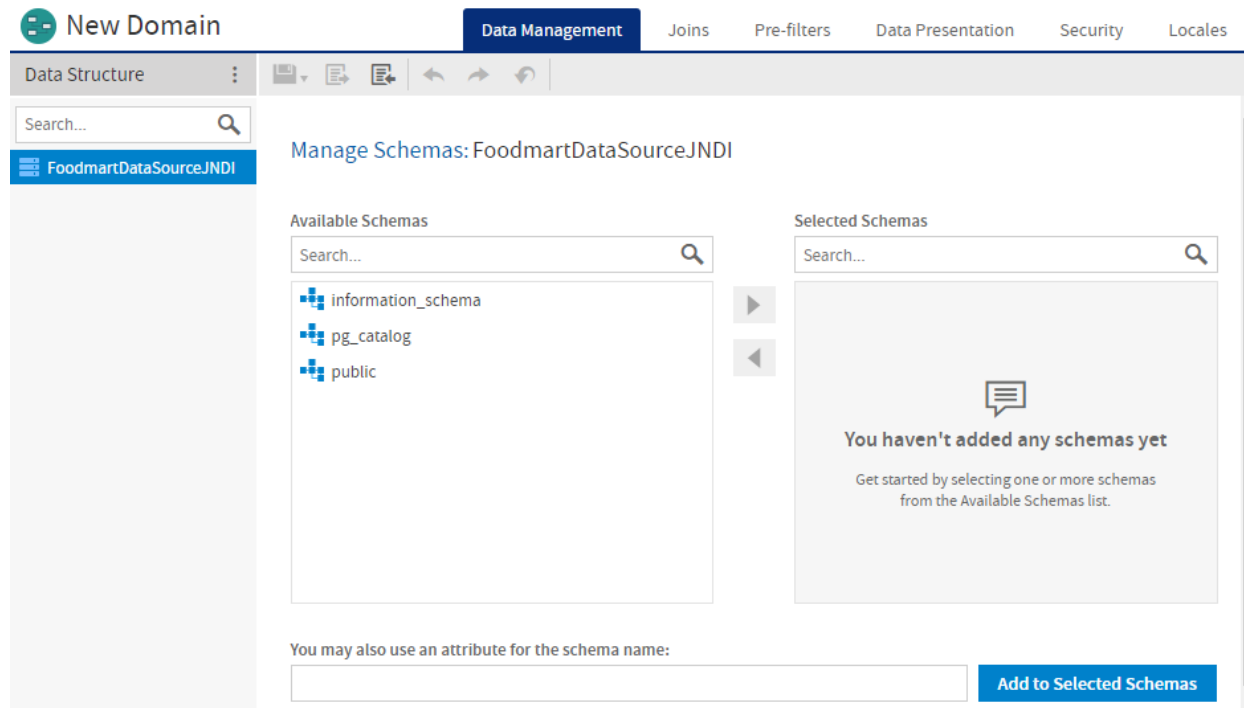


**Figure 3-4  The Data Management tab before a schema has been added to the Domain**

### 3.3.1     The Data Structure Panel on the Data Management Tab

On the Data Management tab, the Data Structure panel displays the data source and the schemas, tables, and columns from the data source that you have added to the Domain. It also displays any constant fields and derived tables you have created.

> The Data Structure panel on the Data Management tab does not display internally-defined Domain elements, such as joins, calculated fields, or copied tables. It does display derived tables.

> The Data Structure panel shows columns that have a supported type listed in **Supported Types**. If the data source has special datatypes like CLOB or NVARCHAR2, or if you access synonyms on an Oracle database, you need to configure the server to recognize them. See the configuration chapter in the *JasperReports Server Administrator Guide*.

- Expand the data source to see the schemas you have added to your Domain design. If you have added any derived tables, they appear under a separate node.
- Select the data source to display the Manage Schemas panel and add or remove schemas from your design.
- Right-click the data source to display a context menu with the following choices:
    - **Create Derived Table…** – Select this to open the New Derived Table dialog and create a derived table. See **4.2, "Derived Tables," on page 42** for more information.

- Expand a schema to see which of its tables you have added to your design.
- Select a schema to display the Manage Tables panel for that schema and add or remove tables from your design.
- Expand the Derived Tables node to view the derived tables you have created.
- Right-click a derived table to copy, edit, or remove it.
- Expand a table or derived table to view its columns.

> You cannot select tables or columns in the Data Structure panel on the Data Management tab. You can select them on the Joins, Pre-filters, and Data Presentation tabs.

## 3.3.2 Managing Schemas

If your data source supports database schemas, like Oracle RDBMS, you need to choose one or more schemas to use in the Domain. You also need to select schemas when using a virtual data source.
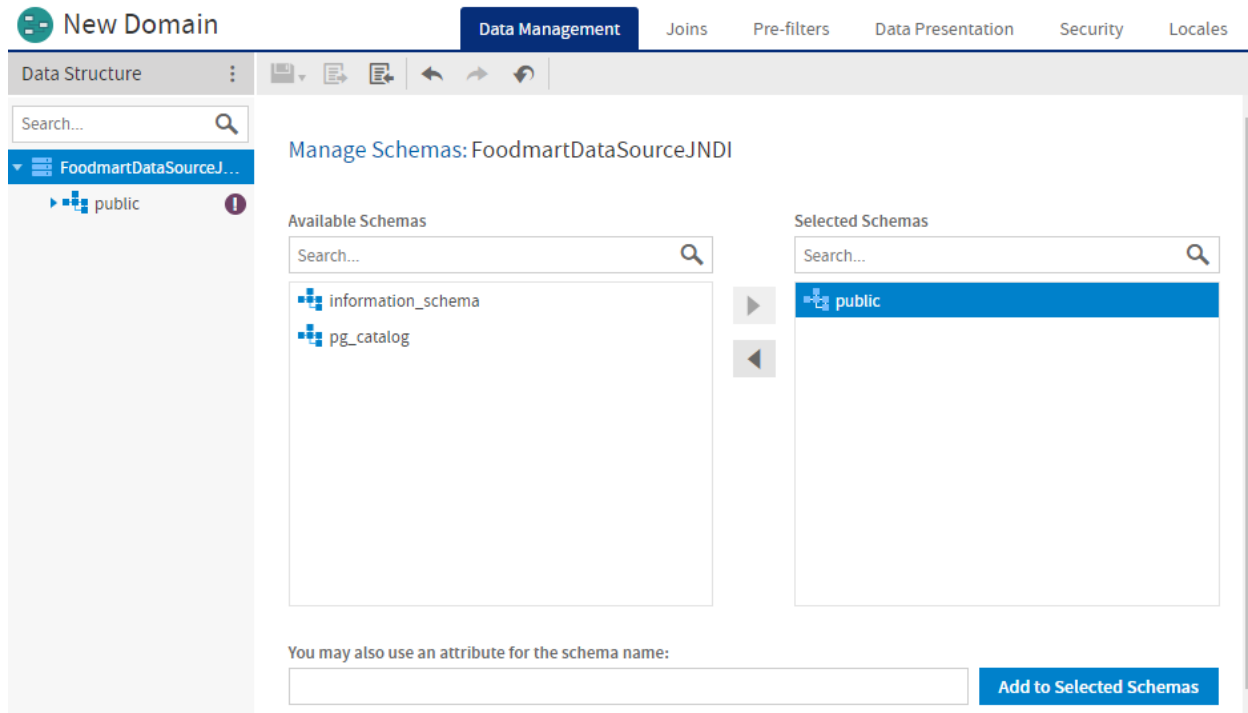


**Figure 3-5**  **Select Database Schemas**

To add schemas, first select the data source in the Data Structure panel on the Data Management tab. The Manage Schemas panel appears in the Data Design panel.

> The Manage Schemas panel appears automatically when you create a new domain.

- The **Available Schemas** list displays the available schemas in your data source.
- The **Selected Schemas** list shows the schemas you selected.

- Selected schemas appear in the Data Structure panel. If you have not yet added any tables from the schema to the Domain design, 🛈 appears shown to the right of the schema name. See **3.3.3, "Managing Tables," on page 22** for more information.
- You can move a schema back and forth between the lists by dragging, double-clicking, or selecting the item and clicking an arrow button, such as ▶ .

### 3.3.2.1  Using Attributes

You can use an attribute for a schema name. When you use an attribute for the schema name, the Domain Designer displays current name of the attribute as the schema and retrieves the tables and columns from that schema. See **4.5, "Using Attributes in the Domain Designer," on page 53** for more information.

> 📝 When you use an attribute for a schema name, the Domain will fail if the schema is missing. You can optionally set a default schema in the Domain design file. The default schema is used whenever the schema is missing or undefined. See **6.2.5.1, "Default Schema," on page 71** for more information.

**To use an attribute for a schema:**

1.  Make sure you have defined the attribute you want. See the *JasperReports Server Administrator Guide* for information about creating attributes.

2.  Enter the correct syntax for the attribute in the **You may also use an attribute for the schema name:** text box, depending on the attribute level:

    - `{attribute('attributeName'), 'server'}` for a server-level attribute.
    - `{attribute('attributeName', 'organization')}` for an organization-level attribute.
    - `{attribute('attributeName', 'user')}` for a user-level attribute.

    If no level is specified, the server will search for the attribute hierarchically, starting at the `'user'` level:

    - `{attribute('attributeName')}`



**Figure 3-6**  Using a server-level attribute for a schema name

> 📝 If you forget the attribute syntax, enter any string in the **You may also use an attribute for the schema name:** text box and click **Add to Selected Schemas** to see a hint with the correct syntax.

3.  Click **Add to Selected Schemas** to add the attribute to the Selected Schemas list.

    The attribute is resolved to its current value and the corresponding schema is shown in the Selected Schemas list. A special icon 🔲 appears.

You can view the attribute information for a schema, if any, by going to the Data Management tab, clicking on the data source node, and then selecting the schema in the Selected Schemas list. The attribute information is show in the **You may also use an attribute for the schema name:** text box.

### 3.3.3 Managing Tables

You need to choose one or more tables to use in the Domain. Typically, you select the following tables for use in the Domain:

- Tables that you want to expose to report designers and users.
- Tables that you want to join to other tables.
- Tables that you want to reference in the Domain design, even if their columns do not appear directly in the Domain. For example, make sure to select the tables containing columns that you want to use in a derived table or calculated field.

> An understanding of the logical design of tables in the data source is critical to selecting the tables to be joined. Once you have added a table, you can expand it in the Data Structure tab to verify it contains the columns you want.
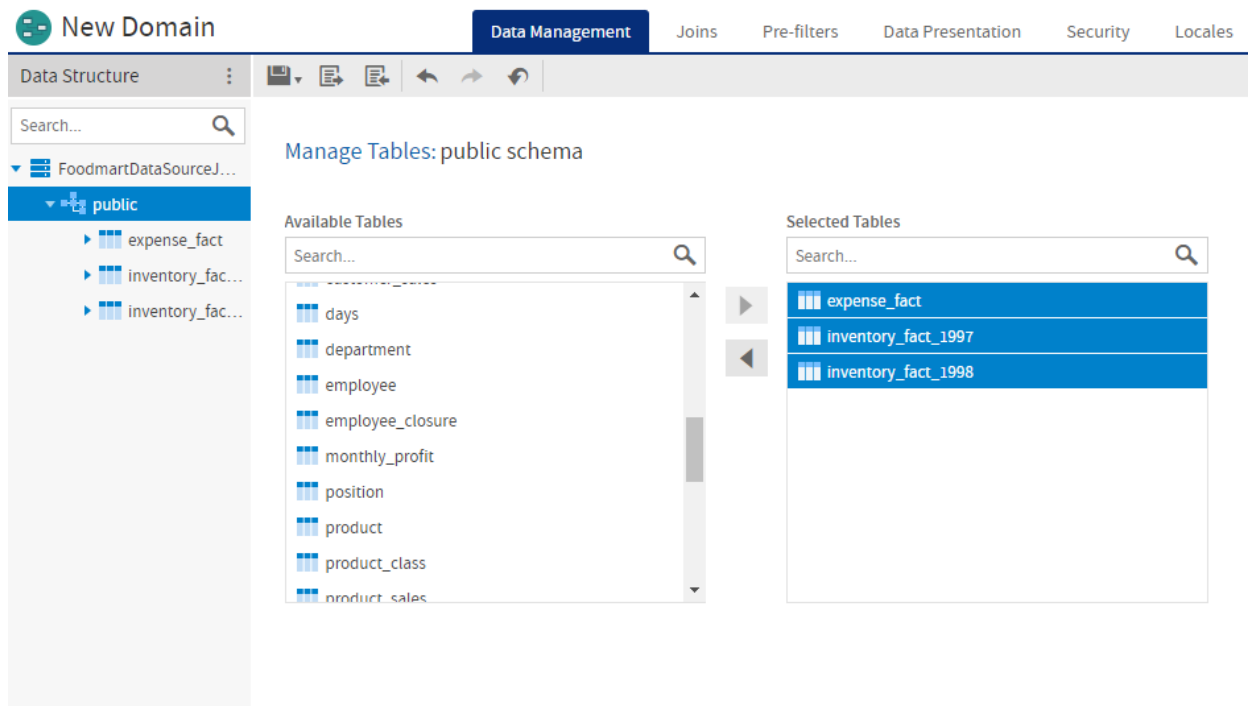


**Figure 3-7** **Select Tables**

To work with tables, first select a schema in the Data Structure panel on the Data Management tab. The Manage Tables panel appears on the right and presents two lists:

- The **Available Tables** list displays the available tables in the selected schema.
- The **Selected Tables** list shows the tables in the selected schema that you have added to the Domain. Initially, this list is blank. When you add a table to this list, the table name also appears in the Data Structure panel.
- Selected tables appear in the Data Structure panel under the correct schema.

To add a table to the Domain:

1. Select a schema in the Data Structure panel on the Data Management tab.
2. Select the table you want in **Available Tables**. Ctrl-click to select multiple tables

3.  Use ▶ to move the highlighted tables to the **Selected Tables** panel. Alternatively, double-click or drag the table name from the **Available Tables** panel to the **Selected Tables** panel.
4.  To remove tables, use ◀, double-click a table, or drag a table from **Selected Tables** to **Available Tables**.

> On the Data Management tab, you can select only entire tables. On the Joins, Pre-filters, and Data Presentation tabs, you can make table- and column-level selections.

## 3.4    The Joins Tab

The Joins tab is where you create associations between tables so that you can present them together in the same report. Multiple joins associate columns across many tables to create powerful data visualizations when used in reports. The number and complexity of joins in the Domain depends on your business needs.
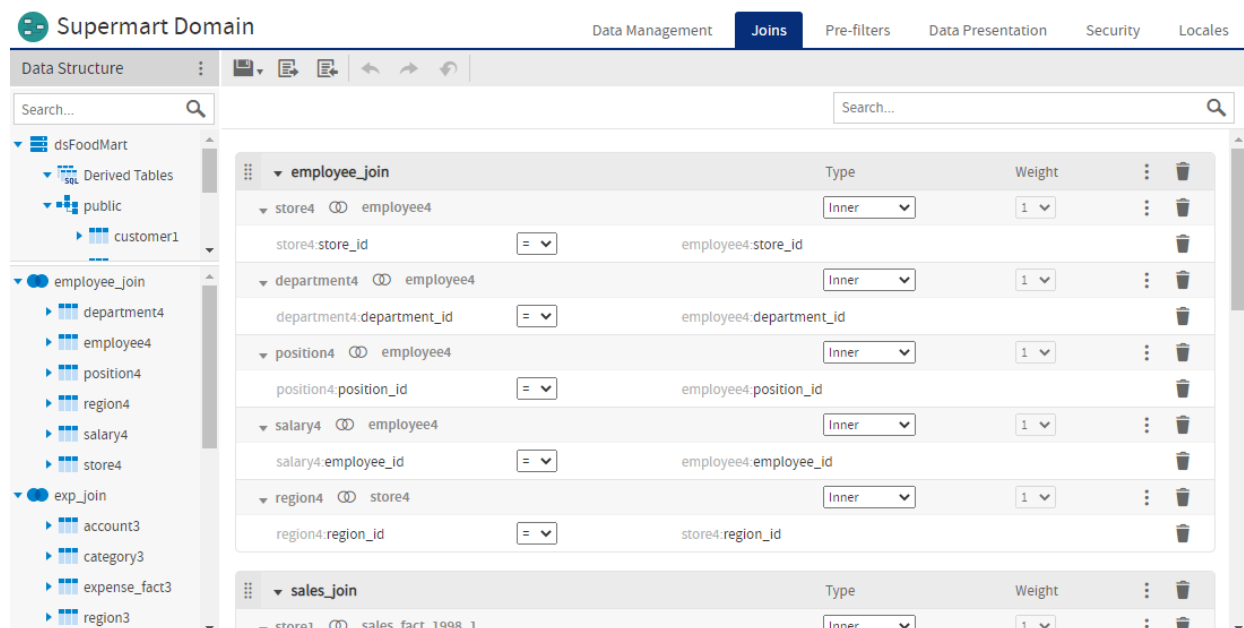


**Figure 3-8**  **Layout of the Joins Tab**

Joins in the Domain Designer are grouped in join trees. A join tree is a group of tables that are all connected directly or indirectly through joins. Different join trees have no connections between them. The Joins tab shows tables organized by join tree in the Data Structure panel, and joins organized by join tree in the Design panel. You create joins by dragging fields from the Data Structure panel to the Design panel.

## 3.4.1    The Data Structure Panel

On the Joins tab, the Data Structure panel displays the tables in your Domains, organized into join trees. In this view, a single join tree contains a group of tables that are all connected directly or indirectly through joins. Unjoined tables and columns appear at the top (underneath the data source node), and joined tables and their columns appear at the bottom. The following actions are available:

•   Expand the data source node to see the unjoined tables and columns in your Domain.

- Drag the divider between the data source tables and the join trees to adjust the viewable area.
- Expand a join tree to see the tables it contains.
- Expand a table or derived table to see the columns it contains.
- Drag a column to the Joins design panel to use it in a join.
- Right-click the data source node to display a context menu with the following choices:
  - **Generate Joins** – Automatically creates joins based on foreign keys. If the database does not have foreign keys, you see an error and no joins are generated.
- Right-click a join tree to display a context menu with the following choices:
  - **Create Calculated Field** – Opens the Calculated Field dialog box to create a calculated field using columns from the tables in the join. See **4.3, "Calculated Fields," on page 43** for more information.
- Right-click a table to display a context-sensitive menu that includes some of the following:
  - **Copy Table** – Copies the selected table and gives it a name with sequential numbering.
  - **Rename Table** – Changes the name of the selected table. The new name becomes the ID of the table throughout the Domain, and is updated everywhere it appears in the Domain Designer.
  - **Remove Table** – (Copy of table only.) Removes the selected table copy from the Domain. To remove the original table from the Domain, use the Data Management tab.
  - **Edit Derived Table…** – (Derived table only.) Opens the Edit Derived Table dialog, where you can change the table name, query, and selected fields. If you change the name, the new name becomes the ID of the table throughout the Domain and is updated everywhere it appears in the Domain Designer.
  - **Remove Derived Table…** – (Derived table only.) Removes the selected derived table from the Domain.
  - **Always Include Table** – (Table in a join tree only; not available for unjoined tables.) Includes this table in all Ad Hoc views that use this join tree, even if the table is not explicitly added by the Ad Hoc user. Enable **Use Minimum Path Joins** when you select this option. See **4.4.8, "Prioritizing Joins," on page 50** for more information.
  - **Create Calculated Field** – Opens the Calculated Field dialog box to create a calculated field using the columns in the table. See **4.3, "Calculated Fields," on page 43** for more information.
- Hover over a table or table copy to see the name of the original table in the data source.

## 3.4.2    The Joins Design Panel

The Joins design panel shows the join trees and joins you have added to your Domain.

### 3.4.2.1  Working With Join Trees

In the Joins panel, the join trees in your Domain appear as containers with individual joins inside them. The selected join options are displayed for each join. For more information about the many possible join operations, see **4.4, "Advanced Joins," on page 46**.

**Figure 3-9**  A join tree with multiple joins

The join tree title bar at the top of each join tree lets you perform the following actions:

- ⠿ – Drag to move the join tree container to a new location and change the order of the join trees. A thick blue line appears in locations where you can place the join tree.
- ▶▼ – Use the arrows to expand and collapse the join tree.
- **Type**, **Weight** – Read-only headings that refer to properties of the individual joins in the join tree.
- ⋮ – **More**. Click to perform the following actions:
  - **Rename Join Tree** – Displays the Rename Join Tree dialog, where you can give your join a more useful name.
  - **Use Minimum Path Joins** – Helps you choose how joins are chosen in Ad Hoc views. Deselected by default; it is good practice to select this. See **4.4.8, "Prioritizing Joins," on page 50** for more information.
  - **Use all joins** – Forces an Ad Hoc view to include all joins in the join tree. Included for backwards compatibility. Unless you have specific need for it, it is good practice to leave this option unselected. See **4.4.8, "Prioritizing Joins," on page 50** for more information.
- 🗑 – Click to remove the join tree and all the joins it contains. If there are other Domain items that depend on the join tree, such as pre-filters or sets based on the join, you are prompted to remove them. The tables and columns remain in your Domain.

### 3.4.2.2  Working with Joins

Individual joins appear in their join tree container. A join includes a title bar and entries for the joined columns.



**Figure 3-10**  A Single Join Between Two Tables

If you create multiple joins between the same two tables, the new join component is added to the existing join. See **4.4.5, "Composite Joins," on page 48** for more information.
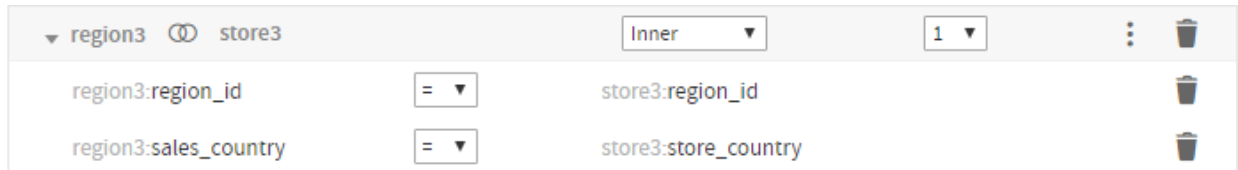


**Figure 3-11** **Multiple Joins For the Same Two Tables**

The join title bar at the top of each join includes the following:

- ▶ ▼ – Use these arrows to expand and collapse the join.
- Left column name (read-only) – Displays the table used in the left side of the join.
- Join icon
- Right column name (read-only) – Displays the table used in the right side of the join.
- **Type** drop-down – Sets the type of your join: **Inner**, **Left Outer**, **Right Outer**, or **Full Outer**. See **4.4.3, "Join Type," on page 47**.

> **Full Outer** is not supported for MySQL databases.

- **Weight** drop-down (default = 1) – Influences which joins are used in an Ad Hoc view. Higher weights indicate less desirable joins. To change this value, you must enable **Use minimum path joins** for this join tree. See **4.4.8.3, "Join Weights," on page 52** for more information.
- ⋮ – **More**. Selecting **Create Custom Joins...** opens the **New Custom Join** dialog, which allows you to set a constant condition on a single column. See **4.4.6, "Custom Joins," on page 48** for more information.
- 🗑 – Deletes the join. If there are other Domain items that depend on the join, s pre-filters or sets based on the join, you are prompted to remove them. The tables and columns remain in your Domain.

The entry for each component displays the following:

- Left column name (read-only) – Displays the table and column for the left side of the join.
- Join comparison menu – Drop-down that lets you select a comparison operator, based on the type of the join. Operators include =, ≠, >, <, >=, and <=.

> Comparison operators other than = (such as ≠, >, <, >=, <= ) generate a large amount of rows (similar to a Cartesian product) when used on their own. Best practice is to use them as constraints in conjunction with a = join on the same pair of tables. See **4.4.4, "Comparison Operators," on page 47** for more information.

- Right column name (read-only) – Displays the table and column for the right side of the join. For a custom join, displays the constant.
- ⋮ (custom joins only) – **More**. Lets you select **Edit Custom Join...**, which opens the Edit Custom Join dialog box.
- 🗑 – Deletes the join component. If there is only one component in the join, deletes the entire join.

### 3.4.2.3  Creating a Join

1.  In the Data Structure panel, find the column you want to use for the left-hand side of your join and drag it into the Design panel in the location you want:

    •   To create a new join on an empty canvas, drag the column anywhere on the Design panel. A new join is created, along with the join tree that contains it.

    •   To add a new join tree when join trees are already present, drag the column to a blank space above, between, or below the existing join trees. The target area is highlighted with a thick blue line.

    •   To add a join to an existing join tree, drag the column to the join tree. The target join tree is outlined with a dashed line to show it is selected.

        A join is added to the Design panel. The column you dragged is added on the left, and the **Drag a field here** is displayed on the right.
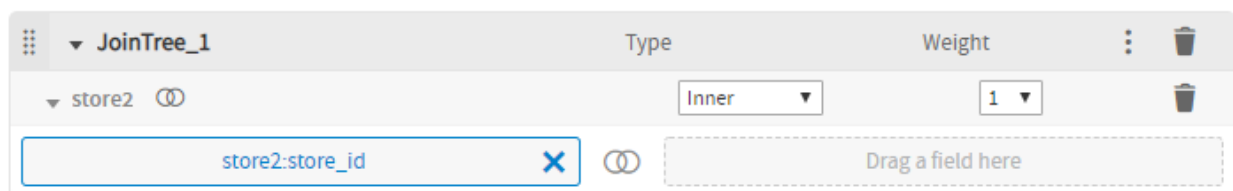


**Figure 3-12**  Creating a New Join

2.  Select the join type (**Inner**, **Left Outer**, **Right Outer**, **Full Outer**) you want from the **Type** menu. This can be changed later. See **4.4.3, "Join Type," on page 47** for more information.

3.  Drag the column for the right table to the **Drag a field here** box. Make sure to choose a column that's compatible with the column you already chose. The column is added to the join.



**Figure 3-13**  Adding the Right Table to a Join

4.  If you want to change the comparison operator for the join, select a different operator from the list. The list of available operators depends on the column type.

    > Comparison operators other than = (such as ≠, >, <, >=, <= ) generate a large amount of rows (similar to a Cartesian product) when used on their own. Best practice is to use them in conjunction with a = join on the same pair of tables. See **4.4.4, "Comparison Operators," on page 47** for more information.

5.  If you want to add a weight to the join, first make sure that **Use minimum path joins** is selected in the join tree settings. Then select the weight you want from the **Weight** menu. Remember that a higher weight means a less desirable join. See **4.4.8.3, "Join Weights," on page 52** for more information.

## 3.5    The Pre-filters Tab

A pre-filter on one or more columns restricts the data that the Domain retrieves from the data source. Pre-filtering irrelevant data reduces the size of query results and processing time. In addition, building frequently-used pre-filters into the Domain design avoids the need for each report designer to define filters independently and reduces the chance for errors.

You can define a pre-filter on a column you don't plan to expose in the Domain. The pre-filter remains active and restricts the data that appears to report users. For example, you can pre-filter data to select a single country, in which case it doesn't make sense for the column to appear in the Domain's presentation data. However, you should clearly document such data restrictions in the description of the Domain, so users understand what data is accessible through the Domain.

> Ad Hoc views and reports based on the Domain can define their own filters. When you use a pre-filter, the data is never retrieved from the database. When you filter on the Ad Hoc level, the data is retrieved, but may not be displayed to the user.

You have two options for defining parameters for a pre-filter:
- Define the parameters statically, so they are the same for every user. For example, you could create a static filter that filters out data for every country except the USA.
- Have the server derive the parameter of the filter at run time based on attributes you provide. For example, you can create a filter that only shows data for the logged-in user's country, based on a Country attribute set in your users' profiles. If the specified attribute has not been defined anywhere on the server, the filter will fail. If the attributes is NULL, the filter will not retrieve any data.

> You can also filter the data using calculated fields. The primary difference between pre-filters and calculated fields is that calculated fields are visible to report designers and users, and can be used as the basis for additional filters in the Ad Hoc Designer. See **4.3, "Calculated Fields," on page 43**

For more information on attributes, see **4.5, "Using Attributes in the Domain Designer," on page 53**.

**To define a pre-filter:**
1. Go to the Pre-filters tab in your Domain.
2. Drag a column from the Data Structure panel to the Pre-filters panel.

    The column appears in the Field column of the Pre-filters design panel with a list of comparison operators you can apply to that column.
3. Choose the comparison operator from the drop-down in the Operator column.

    In the Pre-filters panel, the choice of comparison operators depends on the column's datatype. For example, strings offer a choice of search operators and dates offer time-comparison operators.
4. Select **Field to Value Comparison** or **Field to Field Comparison** from the ⋮ menu.

    If you select **Field to Field Comparison**, the Value column displays a box labeled **Drag a field here**.
5. Select or enter a value for your filter. For a filter that compares fields, drag field(s) of the same type to the **Drag a field here** box.

    The format of the filter value changes depending on your previous selections. For example, if you select a date column with the `is between` operator, the Filters panel displays two calendar widgets for specifying a date range:
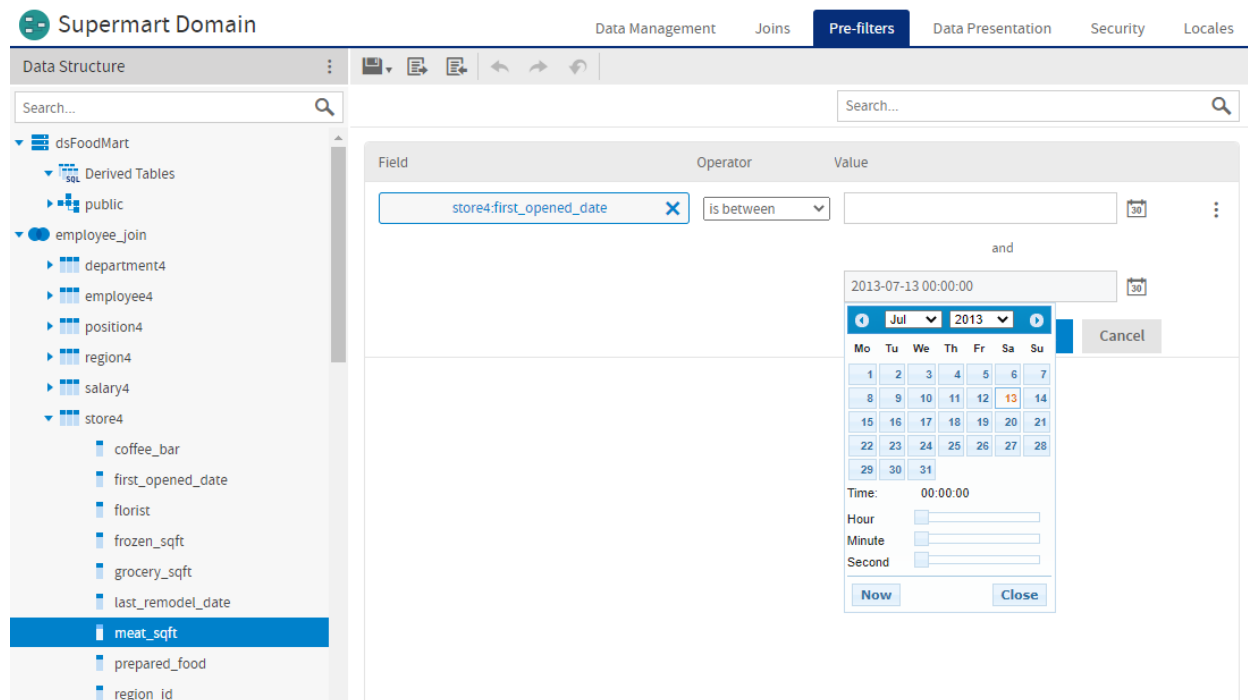
**Figure 3-14**  **Filters Panel of the Domain Designer**

Text columns have both substring comparison operators such as `starts with` or `contains` and whole string matching such as `equals` or `is one of`. When you select a whole string matching operator, the panel displays a list of all existing values for the chosen column, retrieved in real-time from the database. If more than 50 values are available, use search 🔍 to narrow the list. For multiple value matching, double-click the

available values to select them. You may perform multiple searches and select values from each list of results.

If you want to use an attribute for a parameter, enter the attribute in the field using the syntax `attribute ('<attrName>')`. For example, if you want to use an attribute called Cities in the filter, enter `attribute ('Cities')` into the field. If you want to specify that this is a user attribute, and not an organizational or server one, enter `attribute('Cities', 'user')`.

6.  Click **OK** to define the filter.

    The Pre-filters panel shows all the filters you have defined. The overall filter applied to the data is the logical AND of all conditions you defined.

To modify an existing filter, click 🖉 . To remove a filter, click 🗑 .

> ⚠️ Pre-filters defined in the Domain Designer are limited to conditions on one column or comparisons of two columns, with more complex filters created by the conjunction (logical AND) of several conditions. Other filter expressions are not supported. You can create more complex filters in the Domain Design file, but these filters are read-only and are not editable in the Domain Designer.

## 3.6 The Data Presentation Tab

The Data Presentation tab is where you specify the columns and calculated fields you want visible to users. You can also create user-friendly names and descriptions and set other column display properties. Typically, you expose only columns that are useful in building or filtering reports. Columns that you don't display are still part of the Domain and can affect the data retrieved.
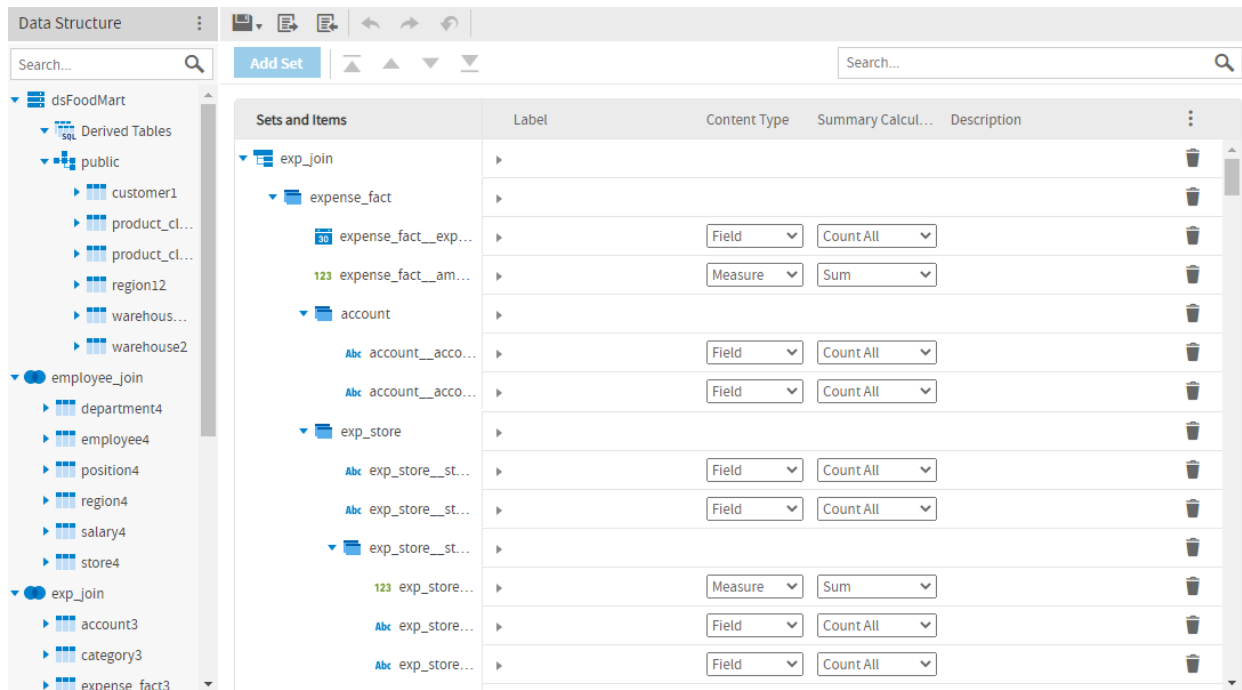


**Figure 3-15** The Data Presentation Tab

The Data Presentation tab contains the following:
- **Data Structure** panel – Displays available tables and fields for this Domain, including table copies, derived tables, and calculated fields. Joined tables appear as join trees. Dragging a join tree, table, or column from Data Structure to Sets and Items does not remove it from the Data Structure panel. This reflects the fact that you can add a resource to more than one set.

> You cannot delete tables from the Domain on the Data Presentation tab. You must do this on the Data Management or Joins tab.

- **Sets and Items** list – Lists resources that will appear to report creators who use the Domain.
- **Properties** pane – Displays the properties of the associated set or item. This area can be expanded or collapsed.

To specify which columns and calculated fields are exposed to users of the Domain, drag them from the **Data Structure** panel to the **Sets and Items** panel. The following icons show the different resources:

-  – A data island. Corresponds to a join tree or unjoined table, but does not necessarily have the same structure. For example, a data island does not need to contain all the tables or columns from its join tree. In addition, it can contain sets that do not correspond to any table, and it can contain the same field or table several times across different sets. However, a data island can only contain resources from a single join tree or

unjoined table, and a join tree or unjoined table can correspond to at most one data island. Can contain sets and items.

When a user creates an Ad Hoc view, they can only choose sets from the same data island.

- – A set. A group of resources; can contain items and other sets. Sets can be created by dragging a table, in which case they can only contain items from that table; or they can be created by clicking **Add Set**, in which case they can contain items from different tables in the same join tree.
- Item icons. An item is a column or calculated field that you want to appear in the Domain. The icon shows the data type of the item:

| Icon | Description |
|------|-------------|
|  | Boolean field |
|  | Boolean measure |
| *fx* | calculated field |
| *fx* | calculated measure |
| 30 | date field |
| 30 | date measure |
| 123 | numeric field |
| 123 | numeric measure |
| Abc | string field |
| Abc | string measure |

## 3.6.1  The Data Structure Panel

On the Data Presentation tab, the Data Structure panel displays the tables in your Domains organized into join trees. In this view, a single join tree contains a group of tables that are all connected directly or indirectly through joins. Unjoined tables and columns appear at the top (underneath the data source node), and joined tables and their columns appear at the bottom. The following actions are available:

- Expand the data source node to see the unjoined tables and columns in your Domain.
- Expand a join tree to see the tables it contains.
- Expand a table or derived table to see the columns it contains.
- Use **Ctrl-click** or **Command-click** to select multiple items; use **Shift-click** to select a range of items.
- Drag selected items to the Sets and Items design panel to display it to the users.
- Hover over an item to see its ID.

In previous versions, you were not required to create data islands. Now, adding an item automatically adds it to the correct data island or creates a new data island if it does not yet exist. Opening and saving a Domain with the previous model updates it the current model. It is good practice to update older Domains.

## 3.6.2    The Sets and Items List

The Sets and Items list shows the resources you want the user to see, organized into data islands, sets, and items in a nested hierarchy. If there are sets and items at the same level, items appear first. The menu bar at the top of the Sets and Items list shows the following:

- **Add Set** – Creates a new subset of a selected set. Not available when no sets have been added.
- ▲ ▲ ▼ ▼ – Move the current selection as follows: to the top, up, down, or to the bottom. Data islands are moved relative to other data islands; items and sets are moved within their containing set or data island. You can also move a set or item by dragging. You can reorder sets and items by dragging them to another location in the same set; however, when you close and reopen a Domain, sets always appear below items. Data islands can be dragged above or below other data islands. You can move items and lower-level sets to any level in the same data island. If you drag a set or item between sets, it appears at the top of the set that it was moved to.

⋮ – Controls the display of sets and items and their properties:

- The following selections control the properties you view when a set or item is collapsed. When a resource is expanded, all available properties are visible.

| Menu Item | Description | Visible Properties |
|---|---|---|
| **Default Properties** | Displays an overview of the resource. | Label, Content Type, Summary Calculation, Description |
| **Identification Properties** | Lets you view the ID and edit the label and description. | Label, ID, Description |
| **Bundle Keys Properties** | Lets you view and edit properties related to bundle keys for localization. | Label Key, Descriptions Key |
| **Data Properties** | Lets you select whether an item is a dimension or measure; lets you view and edit properties specific to measures. | Source, Content Type, Summary Calculation, Data Format |

- The following selections let you expand and collapse the list of sets and items and their properties.
  - **Expand All Properties** – Expands all sets and items and displays expanded properties for all.
  - **Collapse All Properties** – Collapses all sets and items and displays collapsed properties for all. The properties shown depend on the selection you made.

The following selections are available for resources:

- ▼, ▶ – Expands or collapses a node in the Sets and Items list.▼
- ▼, ▶ – Expands or collapses properties. Click in any property to start editing it.
- 🗑 – Removes the associated resource.

- – Searches items and sets and shows matches if any of the following contain the search string: label, ID, description, label key, descriptions key. Can be used to quickly find similar items and compare and edit their properties.

### 3.6.3    Properties



**Figure 3-16**  Item Properties on the Data Presentation Tab

The Properties pane of the Data Presentation tab lets you refine the Domain's appearance by renaming and providing descriptions for data islands, sets, and items. The following table describes the properties that may appear.

| Property | Appears On | Description |
|---|---|---|
| Label | Data Island, Set, Item | User-friendly name displayed in the Data Chooser and the Ad Hoc Editor. |
| ID | Data Island, Set, Item | An identifier used within the Domain. Default table and field IDs are based on the names in the data source, but you can change the ID of a table as long as it remains unique. Presentation IDs are a separate namespace in which each ID must be unique, although based on table and field IDs by default. The ID property value must be alphanumeric. You should not change IDs for a Domain that has already been used to create Ad Hoc views or Topics. |
| Description | Data Island, Set, Item | User-friendly description displayed as tooltip on the label in the Ad Hoc Editor. The description helps the report creator understand the data represented by this set or item. |
| Content Type | Item | Label that designates the item as either a qualitative value (field) or quantitative value (measure). By default, all numeric types are assumed to be measures, and all non-numeric items are plain fields. Use this setting to override the default. |

| Property | Appears On | Description |
|---|---|---|
| Summary Calculation | Item | Default summary calculation for the item when used in a report. The available functions depend on the field's data type (Boolean, date, numeric, or text). See the *JasperReports Server User Guide* for more information. |
| Data Format | Item | Default numerical format (such as number of decimal places) for the item when used in a report. Numeric and dates only. |
| Label Key | Data Island, Set, Item | Internationalization key for the label property; locale bundles associate this key with the localized text for the label. |
| Descriptions Key | Data Island, Set, Item | Internationalization key for the description property; locale bundles associate this key with the localized text for the description. |
| Source | Item | References the data source, schema, table, and field associated with this item; the syntax is `datasource.schema.table.field`. Not editable. |

Labels and descriptions are visible to users of the Domain. Descriptions of sets and items appear as tooltips in the Ad Hoc Editor to help report creators understand their purpose.

The internationalization keys must match the property names of strings in locale bundles. Keys may use only characters from the ISO Latin-1 set, digits, and underscores (_). If you create internationalization keys on the Data Presentation tab, you can download the keys as a template file for locale bundles by clicking **Download Template** in the Locale Bundles section on the Options tab.

## 3.6.4    Working With Sets and Items

**Adding a set or item:**

To move a resource to Sets and Items, drag it from the Data Structure panel and drop it in Sets and Items. A blue bar appears in the target location. If you can't see the blue bar, then you can't drop the resource in that location.
- To drop a resource in an existing data island or set, drag it anywhere in the data island or set. It appears at the bottom.
- To add a resource that does not belong to an existing data island, drag the resource to the top or bottom of the Sets and Items list, or in the area between any two data islands. The resource is added, along with the data island that contains it. If you drag a join tree, it creates the corresponding data island.

**To move most of the tables and columns in a join tree or a table to Sets and Items:**
1. Drag-and-drop a join tree or table from the Data Structure panel to Sets and Items.
    - If you drag a join tree, it is added as a data island, all tables are added as sets, and columns are added as items within the sets.
    - If you drag a table, the table is added as a set, and all columns in the table are added as items with the set.
2. Remove any unwanted sets or items individually from Sets and Items using 🗑.

**To move a few columns from a join tree or a table to Sets and Items:**
1. Drag a single column from the Data Structure panel to Sets and Items.
    - If the column is part of a join tree, the join tree is added as a data island, and the column is added as an item directly under that data island.

- If the column is part of an unjoined table, the table is added as a data island and the column is added as an item.

2. Continue to drag columns to the data island that was created in the first step.

   Columns are added directly to the data island.

**To create an empty set:**

You can create empty sets. If the parent is a data island or a set , the set can contain items from different tables.

1. Click on the set or data island you want as a parent.

2. Click **Add Set**. You must have a parent selected to use **Add Set**. This button is not active if you do not have any data islands.

   The new set appears at the bottom of the parent set you chose. Drag resources from the same join tree or unjoined table to add them to the set.

**To change the order of items within a set in Sets and Items:**

1. Select the item.

2. Reposition the item using one of the following buttons:

   ▲ Move to top.

   ▲ Move up.

   ▼ Move down.

   ▼ Move to bottom.

You can also reposition an item or set by dragging.

> Individual items inside a set always appear above subsets. You can't move individual items below sets.

### 3.6.4.1  Understanding Data Islands

Elements in the Data Presentation design panel are grouped into data islands. Each data island comes from a join tree or unjoined table, but data islands differ from join trees in a number of important ways. Join trees represent the logical model for data in the Domain and data islands represent the business model presented to the users.

| Join Tree (or Unjoined Table) | Data Island |
|---|---|
| Each join tree is unique and does not connect to any other join tree. | Each data island is unique and does not connect to any other data island. |
| Structure is organized in tables and fields. | Structure is organized in sets and items. |
| Directly reflects the structure of the tables and columns in the database, along with derived tables, calculated fields, and table and column copies (aliases). | Lets you group elements in sets to create a structure that better reflects your business model. You can exclude tables and columns you do not want to expose to the user, without removing them from the original join tree. |

| Join Tree (or Unjoined Table) | Data Island |
|---|---|
| Table and column names come from database or internal Domain Designer naming (for calculated fields, derived tables, and table copies). | Sets and items can be renamed. Names that appear to users can be defined in localization bundles based on strings assigned in the data island. |

## 3.6.5 Properties



**Figure 3-17** **Item Properties on the Data Presentation Tab**

The Properties pane of the Data Presentation tab lets you refine the Domain's appearance by renaming and providing descriptions for sets, subsets, and items. The following table describes the available properties:

| Property | Appears On | Description |
|---|---|---|
| Label | Set, Item | User-friendly name displayed in the Data Chooser and the Ad Hoc Editor. |
| ID | Set, Item | An identifier used within the Domain. Default table and field IDs are based on the names in the data source, but you can change the ID of a table as long as it remains unique. Set and item IDs are a separate namespace in which each ID must be unique, although based on table and field IDs by default. The ID property value must be alphanumeric.<br><br>You should not change IDs for a Domain that has been used to create Ad Hoc views or Topics. |
| Description | Set, Item | User-friendly description displayed as tooltip on the label in the Ad Hoc Editor. The description helps the report creator understand the data represented by this set or item. |
| Content Type | Item | Designates the item as either a qualitative value (field) or quantitative value (measure). By default, all numeric types are assumed to be measures, and all non-numeric items are plain fields. Use this setting to override the default. |

| Property | Appears On | Description |
|---|---|---|
| Summary Calculation | Item | Default summary calculation of the item when used in a report. The available functions depend on the field's data type (Boolean, date, numeric, or text). See the *JasperReports Server User Guide* for more information. |
| Data Format | Item | Default numerical format (such as number of decimal places) for the item when used in a report. Numeric and dates only. |
| Label Key | Set, Item | Internationalization key for the label property; locale bundles associate this key with the localized text for the label. |
| Descriptions Key | Set, Item | Internationalization key for the description property; locale bundles associate this key with the localized text for the description. |
| Source | Item | References the data source, schema, table, and field associated with this item; the syntax is `datasource.schema.table.field`. Not editable. |

Labels and descriptions are visible to users of the Domain. Descriptions of sets and items appear as tooltips in the Ad Hoc Editor to help report creators understand their purpose.

The internationalization keys must match the property names of strings in locale bundles. Keys may use only characters from the ISO Latin-1 set, digits, and underscores (_).

## 3.7 The Security Tab

The Security tab is where you enter and view rules that restrict access to the data in a Domain, based on a user's roles and attributes. Rules are stored in an XML file that works together with the Domain's XML design file to identify what data you want to expose to each user. When creating or running a report based on a Domain, the user name and roles are checked against the permissions in the security file. When a user is designing a report in the Ad Hoc Editor, they see only the columns to which they have access. When the report runs, portions to which the user has no access are blank.

The Security tab in the Domain designer provides a basic editor that lets you upload, view, and edit the Domain's security file. The contents of the editor is the current security file defined for the Domain.

To create security rules, you need to understand the syntax for Domain files and Domain security files. Make sure to read and understand the material in **Chapter 8, "Securing Data in a Domain," on page 111** before you create a security file.
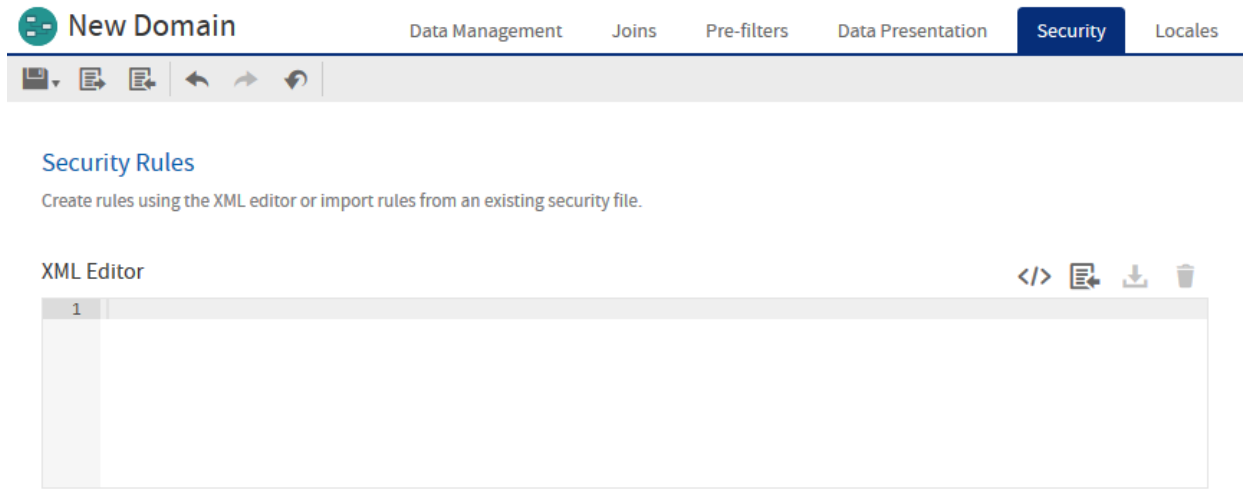
**Figure 3-18  Security Tab with no Security Rules Created**

The security file references the `ids` of tables, columns, sets, and items in the Domain design file. When creating a security file, be sure to use the `ids` of items and groups as they are defined in the Domain design file exported from the Domain Designer.

The Security tab has the following icons to work with the security file editor.

**Table 3-2  Domain Designer Security File Icons**

| Icon | Name | Description |
|------|------|-------------|
| `</>` | Create Template | Generates a generic security file template in the file editor. If the file editor contains any text, you are prompted to replace it. <br><br> This template contains a pattern of access grants that you can use to start your security file. The template does not contain any IDs from the Domain. You can then modify this template to create your file or download it to work in an external editor. For more information, see **8.6.1, "Downloading the Security File Template," on page 118**. |
| | Import | Uploads a security file from your computer or from one stored in the repository. If the file editor contains any text, you are prompted to replace it. For more information, see **8.6.6, "Uploading the Security File," on page 123**. |
| | Download | Prompts you to save the contents of the security file editor as an XML file on your computer. |
| | Clear | Removes the security file from the Domain and erases the content of the editor. If the file editor contains any text, you are prompted before deleting it. |

The editor checks the XML syntax and Domain IDs of your file and lets you fix any errors. Upon saving the Domain, the contents of the editor are validated for join references and principal expressions, then becomes the Domain's security file.

If you modify the Domain, you should edit the security file on this tab with any IDs that have changed. Or if the changes are extensive, download the file, edit it in an external editor, and upload it again.

For a comprehensive example of designing, writing, and uploading a security file, see **Chapter 8, "Securing Data in a Domain," on page 111**.

## 3.8    The Locales Tab

The Locales tab is where you manage locale bundles, used to internationalize your Domain. A locale bundle is a file that provides translated labels for the sets and items of the Domain design. When a user specifies a locale at login and they use the Domain in the Ad Hoc editor or generate reports from the Ad Hoc report, they automatically see the labels that have been translated in the corresponding locale bundle.

You create a separate bundle for each language or locale you want to support. A Domain can have any number of locale bundles. The Locales tab lets you upload, attach, and remove locale bundles.
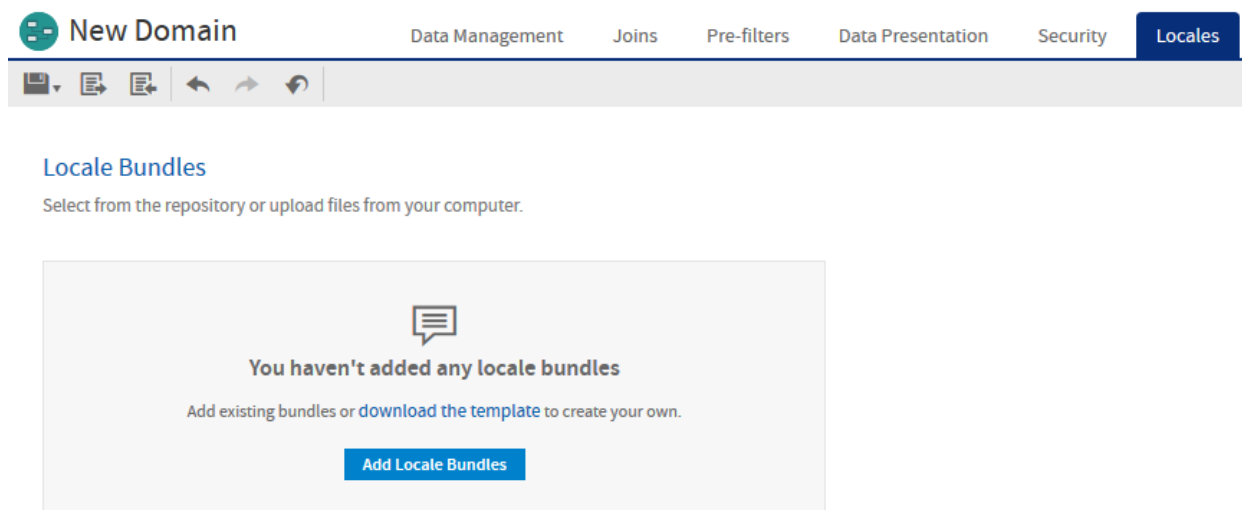


 **Figure 3-19**  **Locales Tab with no Locale Bundles**

For further information about creating locale bundles and uploading them to the Domain, see **Chapter 9, "Localizing Domains," on page 129**.

# CHAPTER 4    ADVANCED DOMAIN TOPICS

This chapter describes how to create more advanced Domains through the interface of the Domain Designer. These topics let you refine your Domain so that the data it access and makes available is better tailored to your users.

This chapter contains the following sections:

- **Domains and Data Virtualization**
- **Derived Tables**
- **Calculated Fields**
- **Advanced Joins**
- **Using Attributes in the Domain Designer**
- **Modifying a Domain**
- **Importing and Exporting Domain Design Files**

## 4.1    Domains and Data Virtualization

There are two ways to integrate data virtually from multiple sources using Domains in JasperReports Server:

- For simple data virtualization with low volumes of data and limited join complexity, create a JasperReports Server virtual data source and build your Domain on top of it. In this case, you define the relationships that connect the tables of the different data sources using the Domain Designer. See the *JasperReports Server Administrator Guide* for more information about virtual data sources in JasperReports Server. This is a lightweight solution that doesn't require any additional server or installation.
- For more complex virtualizations, or where performance and scale are essential, use Jaspersoft Advanced Data Services to access and combine your different data sources. Once you have designed and created a integrated data source in Jaspersoft Advanced Data Services, you create a JasperReports Server JDBC data source to access it and then use the Domain Designer to choose the tables you want and create user-friendly names. Using Jaspersoft Advanced Data Services offers better performance and scaling, with management tools designed specifically for combining multiple data sources. Contact your sales representative for information about licensing for Jaspersoft Advanced Data Services. This requires a separate installation for Jaspersoft Advanced Data Services.

### 4.1.1    Notes on Using Virtual Data Sources

Normally, you combine data sources that have relationships you want to explore. To create a useful Domain using a virtual data source, you need to define relationships that connect the tables of the different data sources. You do

this by adding at least one schema from each data source and then creating a join between tables in the different schemas. If the data sources don't have a column that matches exactly, you can create derived tables that allow you to implement the join you want. For example, if you use the sample SugarFoodMartVDS virtual data source, which combines the Foodmart and SugarCRM JDBC data sources, you can create two derived tables, one that aggregates `sales` by `city` for the Foodmart data source and another that aggregates `amount` by `city` for the SugarCRM data source. Then you can join the two data sources on `city` to combine them.

> Domains built from virtual data sources may have additional syntax restrictions. For example, Domains built from a virtual data source do not work with table names that contain \.

## 4.2    Derived Tables

You can add tables to your Domain based on SQL queries of the chosen data source. These are called derived tables, and once created, they can be used like other tables to create joins and then displayed as sets and items. They can also be referenced in the Domain's security file and locale bundles.

You create a derived table in a Domain using a custom query. Because Domains are based on JDBC and JNDI data sources, you write the query in SQL. You run the query, and then select columns from the query result for use in the Domain design.

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.

> The clauses in a query determine the structure and content of the table returned by the query. For example, the WHERE clause may contain conditions that determine the rows of the derived table.

**To define a derived table:**

1.  Right-click the data source node on the Data Management tab and select **Create Derived Table…** OR click ⋮ and select **Create Derived Table…**.
    The New Derived Table dialog appears.
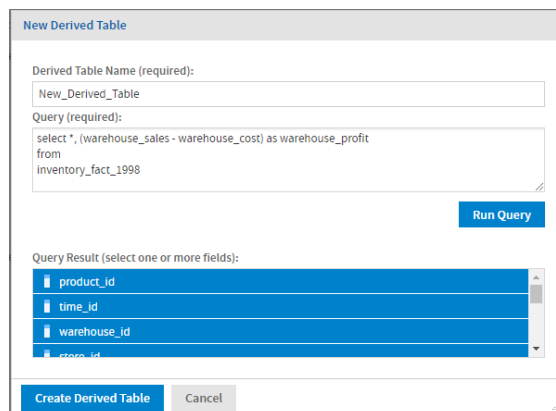


**Figure 4-1**  **New Derived Table dialog**

2. Type a name for the table in the **Derived Table Name** field.

3. Enter a valid SQL query in **Query**. Only queries that begin with the `SELECT` statement are allowed. Stored procedures and functions are supported. Do *not* include a closing semi-colon (`;`).

   To use an attribute, enter `{attribute('AttributeName')}` or `{attribute('AttributeName', 'Level')}`. This must be a single-valued attribute; collections can't be used. See **4.5, "Using Attributes in the Domain Designer," on page 53** for more information.

4. When the query is complete, click **Run Query** to test it. If the query is successful, the resulting fields are displayed in the Query Result list. By default, all columns in the result are selected.

5. Ctrl-click fields in the Query Result list to change the selection. If you want only a few columns out of many, it may be easier to specify the column names in the `SELECT` clause of the query.

6. Click **Create Derived Table**.

   The derived tabled is added under the Derived Tables node in the Data Structure panel. A distinctive icon  identifies it as a derived table.

**To copy a derived table:**

1. Right-click the derived table in the Data Structure panel, and select **Copy Table**.

**To edit a derived table:**

1. Right-click the derived table in the Data Structure panel, and select **Edit Derived Table…**. The Edit Derived Table dialog appears.

2. Edit the Derived Table Name, Query, and selected tables in the Query Result list.

3. Click **Update Derived Table** to save changes.

**To remove a derived table:**

1. Right-click the derived table in the Data Structure panel, and select **Remove Derived Table…**.

> You can't copy, edit, or remove derived tables on the Data Presentation tab.

## 4.3    Calculated Fields

You can add new colums to a Domain by performing calculations on other columns. These are called calculated fields, and once created they can be used like any other fields, for example to create additional calculated fields or pre-filters. They can also be referenced in the Domain's security file and locale bundles.

You create a calculated field for a Domain in two ways:

* By writing an expression that computes a value based on the data in one or more columns in a single table or join tree. All columns in the expression for a calculated field must be from the same join tree.
* By creating a constant expression. For example, you might create an integer field named Count that has the value 1 and later has a default summary function to count all occurrences.

> Calculated field expressions use the Domain Expression Language, fully described in **Chapter 7, "Domain Expression Language (DomEL)," on page 105**.
>
> Calculated fields may also be used in the expression to compute another calculated field.

Calculated fields appear in the Data Structure panel on the Joins, Pre-filters, and Data Presentation tabs. Calculated fields are shown in the table or join tree to which they belong in the Data Structure panel. Global constant fields appear above the data source. Calculated fields and constants are identified by a distinctive icon  , and the field name is shown in bold. In addition, if a calculated field is used by another calculated field, its name is also in italics.

## 4.3.1    Creating a Calculated Field

1.  Navigate to the Join tab or Pre-filters tab and locate the table or join tree containing the fields that you want to perform a calculation on. If the fields are in separate tables, they must first be joined.
2.  Right-click on the table or join tree and select **Create Calculated Field**. The New Calculated Field dialog appears.
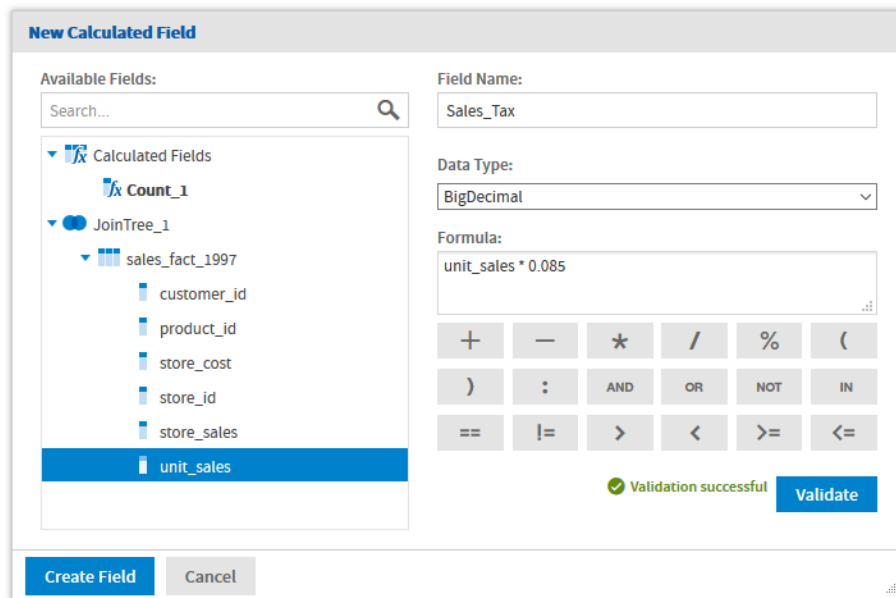


**Figure 4-2**  New Calculated Field

3.  In **Field Name**, enter the name you want to use for the calculated field. This name becomes the field's ID in the Domain.

    After it has been created, you can give the calculated field a more meaningful label and full description on the Data Presentation tab.
4.  In **Data Type**, select a datatype for the calculated field. The expression you write must return a value of this type.

    Generally, this datatype matches the datatype of the columns in the expression. Therefore, you need to be familiar with the datatypes of columns in the data source.
5.  Enter an expression for the calculated field in the Formula text box:

    •    To insert a reference to the value of another column, find it in the **Available Fields** list and double-click the column name. The column name appears in the expression at the cursor, qualified by its table name. You can also type a column name directly in the Formula box, in the format `tablename.fieldname`.

    •    To add a supported operator, click the operator icons below the Formula box, or type the operator directly.

- To use an attribute, enter `attribute('AttributeName')` or `attribute('AttributeName', 'Level')`. This must be a single-valued attribute; collections can't be used. See **4.5, "Using Attributes in the Domain Designer," on page 53** for more information.

> Calculated field expressions use the Domain Expression Language, fully described in **Chapter 7, "Domain Expression Language (DomEL)," on page 105**.

6. Click **Validate** to verify the calculated field datatype and syntax. You must fix any errors before you can save; the validation error message can help you with this:

> ⚠ Expression result type [integer] doesn't match to expected [bigDecimal]   **Validate**

7. Click **Create Field** to save the new calculated field.

If the calculated field is valid, it appears in the Data Selection panel under the table or join tree you chose. A distinctive icon *fx* identifies it as a calculated field.

## 4.3.2  Creating a Constant Field

An expression that does not reference any fields has a constant value. For example, you might create an integer field named Count_1 that has the value 1 and later has a default summary function to count all occurrences. Constant fields are independent of join trees and appear above the data source in the Data Structure panel.

1. Navigate to any tab where the Data Structure panel appears: Data Management, Joins, Pre-filters, or Presentation.

2. At the top of the Data Structure panel, click ⋮ and select **Create Constant Field…**. The New Calculated Field dialog appears. The Available Fields list shows any other constant fields you have created.



**Figure 4-3**  Creating a Constant Field

3. In **Field Name**, enter the name you want to use for the calculated field. This becomes the ID of the field in the Domain, and you can later give it a label and description on the Presentations tab.

4. In **Data Type**, select a datatype for the calculated field. The datatype must match the result of the expression.

5. Enter a constant expression for the calculated field in the Formula text box. Make sure the value matches selected datatype, for example 1 for integer.

6. Click **Validate** to verify the calculated field is valid. You must fix any errors before you can save; the error message can help you with this.

7. Click **Create Field** to save the new calculated field.

If the constant field is valid, it appears at the top of the Data Selection panel under a Calculated Fields node:
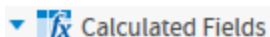


### 4.3.3 Modifying a Calculated Field

1. Locate the calculated field in the Data Structure panel on the Joins tab or Pre-filters tab. Constant fields can be edited from any tab on which they appear.

2. Right-click the calculated field or constant field and select **Edit Calculated Field**. The Edit Calculated Field dialog appears.

3. Modify its name, its type, or its expression.

4. Click **Validate** to verify the calculated field is still valid.

5. Click **Update Field** to save your changes.

### 4.3.4 Deleting a Calculated Field

1. Locate the calculated field in the Data Structure panel on the Joins tab or Pre-filters tab. Constant fields can be removed from any tab on which they appear.

2. Right-click the calculated field or constant field and select **Remove Calculated Field**.

## 4.4 Advanced Joins

Joins are associations you make between tables so that their rows can be presented together in the same report. Multiple joins associate columns across many tables to create powerful data visualizations when used in reports. The number and complexity of joins in the Domain depends on your business needs.

### 4.4.1 Understanding Join Trees

A group of tables that are all connected directly or indirectly through joins is called a join tree. A Domain may contain several join trees, but when a user creates an Ad Hoc view from a Domain, they can only select one of them to be available in the view. Different join trees have no connections between them.

Join trees appear as top-level nodes in the Data Structure panel on the **Joins**, **Pre-Filters** and **Data Presentation** tabs. Tables that are not joined appear as children of the data source node at the top of the Data Structure panel.

The options on the ⋮ menu in the join tree title bar help you prioritize which join paths are chosen when multiple options are possible in an Ad Hoc view. These options are primarily used to avoid circular joins. See **4.4.8, "Prioritizing Joins," on page 50** for more information.

## 4.4.2  Understanding Join Options

If you want to create a join between two tables, each table must have a column that's compatible with a column in the other table. For example, the `store_id` column in the `store` table can be joined with the `store_id` column in the `employee` table.

In some cases, you may need to duplicate a table in order to join it several times without creating a circular join, or to join it to itself. You can also duplicate a table so it may be joined with different tables for different uses.

> The XML design file supports additional join features not supported in the Domain Designer, including:
> * NOT or OR operator between joins inside a composite join.
> * Joins within the same table.
>
> If you have a Domain design file in XML format that uses these features and you open it in the Domain Designer, these joins will be displayed as read-only joins. See **4.4.7, "Read-Only Joins," on page 50** for more information.

## 4.4.3  Join Type

The Domain Designer supports the four most common join types:
* **Inner** – The result contains only rows where the values in the chosen columns are equal.
* **Left Outer** – The result contains all the rows of the left table, paired with a row of the right table when the values in the chosen columns are equal or contain blanks.
* **Right Outer** – The result contains all the rows of the right table, paired with a row of the left table when the values in the chosen columns are equal or contain blanks.
* **Full Outer** – The result contains all rows from both tables, paired when the joined columns are equal, and filled with blanks when the columns are not equal. Not available in MySQL.

## 4.4.4  Comparison Operators

The Domain Designer supports the following comparison operators between fields. You can't compare fields of different data types:

    =, ≠, >, <, >=, <= .

Comparison operators other than = often generate a large amount of rows (similar to a Cartesian product) when used on their own. For best results, use them in a composite join in conjunction with an = join. See **4.4.5, "Composite Joins," on page 48** for more information.
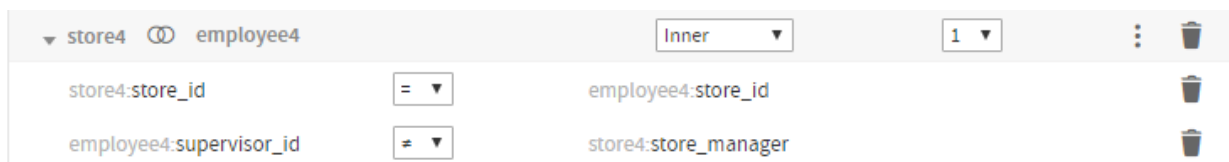
**Figure 4-4** **Example of a Join that Uses an Inequality**

## 4.4.5 Composite Joins

Composite joins implement multiple join conditions for the same pair of tables. You can create composite joins in the following ways:

- Add a join to a join tree that already contains a join between those tables.
- Select **Create Custom Join...** from the ⋮ menu of the join.
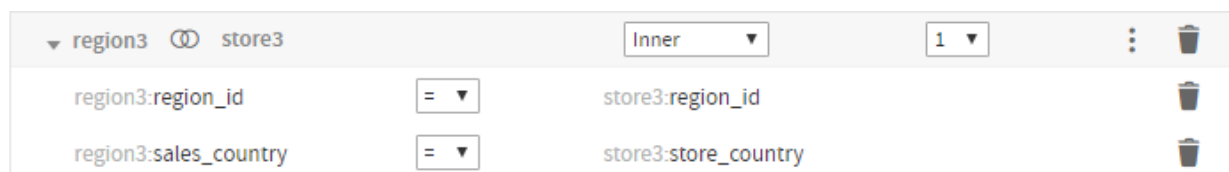


**Figure 4-5** **Example of Composite Join**

The join options you select on the join title bar, such as join type and weight, apply to the entire composite join. In addition, when the Domain is used in an Ad Hoc view or report, the composite join is treated as a single join with multiple components. See **4.4.8, "Prioritizing Joins," on page 50** for more information. When the Domain is exported as XML, composite joins are represented as a single join expression with multiple components.

## 4.4.6 Custom Joins

You can add custom joins to an existing join. Custom joins let you place constant conditions on a single column from the joined tables.

**To add a custom join to an existing join:**

1. Click ⋮ at the upper right of the join and select **Create Custom Join...**. The New Custom Join dialog appears.

**Figure 4-6**  **New Custom Join dialog**

2.  Select a column from the Field list. You can select a column from either table in the join.

3.  Select an operator. The available operators depend on the column type.

4.  Enter the range, set, or value you want for the field, based on the operator you chose.

- • = or ≠ (available for all column types)

  >, <, >=, or <= (available for numeric and date columns only)

  Enter a constant value or an attribute that takes a single value. Strings are enclosed in single quotes. For example:

  - • `5000`
  - • `'Mexico'`
  - • `attribute('CountryAttribute')`

  > For more information about using attributes in Domains, see **4.5, "Using Attributes in the Domain Designer," on page 53**.

- • IN or NOT IN – Enter one of the following:
  - • A set of strings or values, enclosed in parentheses and separated by commas. Strings are enclosed in single quotes. For example:
    - • `(1,2,3,4,5)`
    - • `('San Francisco','Portland', 'Seattle')`
    - • `('true')`
  - • A range of values, separated by a colon (numeric and date columns only). For example:
    - • `(7000 : 8000)`

5.  To verify that your syntax is correct, click **Validate**. Fix errors if necessary.

6.  Click **Create Custom Join**. The join is added below the existing join as part of a composite join.
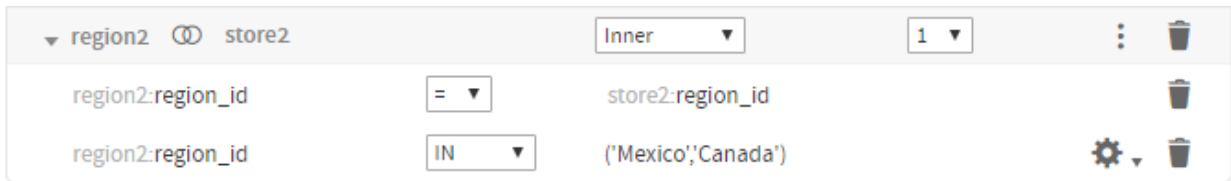
**Figure 4-7** **A custom join in the design panel**

7.   To edit or delete a custom join:
  - Click ⋮ and select **Edit Custom Join…** to open the Edit Custom Join dialog and modify the join.
  - Click 🗑 to delete a custom join.

## 4.4.7    Read-Only Joins

The design file format (XML) supports additional join features not supported in the Domain Designer, including:
- NOT or OR operator in a single join expression.
- Joins between different columns in the same table.

If you have a Domain design file in XML that uses these features and you open it in the Domain Designer, these joins are displayed as read-only joins. The join expression is shown as a string. You can still set the join type and weight.
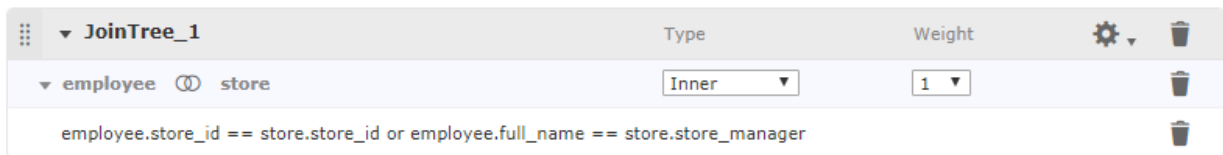


**Figure 4-8** **Example of Read-Only Join**

To create or modify joins outside of the Domain Designer, you must export the Domain in XML format and modify it in a text editor. See **4.7, "Importing and Exporting Domain Design Files," on page 61** and **6.6, "Representing Joins in XML," on page 79** for more information.

## 4.4.8    Prioritizing Joins

The Domain Designer and Domain Design file do not impose any limits on the number of joins you can specify in a join tree. However, when you define joins among N tables, it is good design to use N-1 or fewer joins to avoid circular joins (also known as loops). Circular joins occur, for example, when table A is joined to table B and table B is joined to table C which is in turn joined to table A. However, if your design requires circular joins, JasperReports Server includes features that help you influence how joins are prioritized when they are used in an Ad Hoc view, Topic, or report.
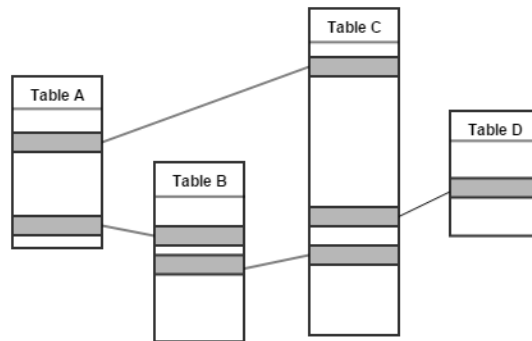
**Figure 4-9**  Circular Joins in a Domain

### 4.4.8.1  Best Practices for Join Trees

Wherever possible, follow these practices to avoid loops or to minimize their impact on Ad Hoc views:

- For a join tree with N tables, use N-1 joins. A composite join is considered a single join. See **4.4.5, "Composite Joins," on page 48** for more information.
- Create copies of tables you need to use more than once in the join tree. You can copy a table by right-clicking it in the Data Structure panel on the Joins tab and selecting **Copy Table** from the context menu.
- Enable **Minimum Path Joins** for each join tree. (This option is not the default.)
- For better join performance, prioritize joins that involve indexed columns. To do this, assign low join weights to the preferred joins and high join weights to less desirable joins.

The options on the ⋮ menu in the join tree title bar, along with specific table and join options, give you influence over which join paths are chosen in an Ad Hoc view when multiple options are possible.

### 4.4.8.2  Join Tree Options

Often, an Ad Hoc view created from a join tree in your Domain does not need to use all the tables and joins in the join tree. The ⋮ menu on the join tree includes options you can use to change how Ad Hoc views select joins. Only one of these options can be enabled at a time.

Suppose you create an Ad Hoc view using two tables from a Domain with a loop. JasperReports Server attempts to avoid circular joins in the SQL by choosing a path between the tables. However, by default, you cannot be sure which join path the Ad Hoc view will use. If you are using tables A and B, you do not know whether the server will use the direct join from A–B or the indirect join A–C–B. The following options influence which joins are used by the Ad Hoc view:

- **Use minimum path joins** (not selected by default) – When this option is selected, an Ad Hoc view created from the join tree uses a minimum join length. When **Use minimum path joins** is selected, you can optionally assign weights to the joins in the join set. Higher weights indicate less desirable joins. For most situations, the best practice is to enable this option to avoid circular joins. Deselect it only if you need backwards compatibility with legacy Domains.
- **Use all joins** (not selected by default) – Available for backwards compatibility. When this attribute is selected, an Ad Hoc view created from the data island includes all the joins in the join tree. For most situations, the best practice is to leave this option unselected.

### 4.4.8.3  Join Weights

When **Use minimum path joins** is enabled, each join in the join tree is given a default weight of 1. You can optionally change the weights of some or all of the joins in the join tree. Higher weights indicate lower-priority, less-desirable joins. When **Use minimum path joins** is used with weights, JasperReports Server calculates the total weight of a join path by summing the weights of its individual joins. It uses join path of the possible lowest total weight to generate the data for the Ad Hoc view, Topic, or report.

> Increasing the weight increases the cost of a particular join and lowers its priority. To give preferential treatment to joins that involve indexed columns for better join performance, assign lower join weights to the preferred joins and higher join weights to less desirable joins.

In this situation, you can enable **Use minimum path joins**. This automatically assigns a default weight of 1 to each join, as shown in the following simplified diagram.

> The best practice is to avoid circular joins in your Domain by creating aliases for one or more tables in the join. You create an alias, or copy, by right-clicking a table in the Data Structure panel on the Joins tab and selecting **Copy Table** from the context menu.
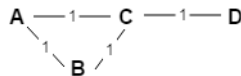


**Figure 4-10**  Circular Join with Default Weights

Now, when you create an Ad Hoc view using table A and table B, the Ad Hoc Designer calculates the weight of the possible join paths from A to B by summing the weights of their subpaths. The direct path A–B has total weight=1, while the indirect path A–C–B has weight 1+1=2. The path with the smallest total weight is chosen, in this case, the direct path from A to B.

Note, however, that this does not cover all cases. For example, suppose you have four joins as shown in the following table, where table A is joined to tables C and D, and table B is also joined to tables C and D. If you create an Ad Hoc view with tables A and B, both possible paths A–C–B and A–D–B have weight 1+1=2, so again, you do not know which join is being used.
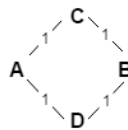


**Figure 4-11**  Circular Joins with Equal Weights Between A and B

To exert even more control over the joins used by your Ad Hoc view, you can add an optional weight to one or more of the joins in your join tree. Adding a weight increases the cost of a particular join, and lowers its priority. The higher you set a weight the more you dislike the join. For example, you can add a weight of 2 to the join between table B and table D. Then you can ensure that an Ad Hoc view with tables A and B will use the A–C–B join path, which only has a total weight of 2, and not the A–D–B join path, which now has a total weight of 3. Note that, in this situation, an Ad Hoc view with tables B and D will still use the direct path with weight 2, instead of the path B–C–A–D, which has a total weight of 3.
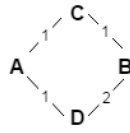
**Figure 4-12**  **Circular Joins with Weights**

### 4.4.8.4 Always Include Table

When **Use minimum path joins** is selected, you can also select **Always Include Table** to the Data Structure panel to specify that any Ad Hoc view created from this join tree must include the indicated table in its join path. This option takes effect even when no columns from the table are used in the Ad Hoc view. You can enable this option on multiple tables in the same join tree.

Technically, you can select this property whether or not you have set **Use minimum path joins**. However, it is most useful when **Use minimum path joins** is selected.

## 4.5 Using Attributes in the Domain Designer

You can use attributes in the Domain Designer to have the server derive values at run time, based on the user, organization, or server. The Domain Designer supports attributes in the following places:

- Schema names
- Derived table queries
- Calculated fields
- Custom joins
- Pre-filters

In addition, you can use attributes in a data source definition. This does not appear directly in the Domain Designer, but allows you to change the data source based on the attribute value.

### 4.5.1 Attribute Syntax

When referring to an attribute, 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. You can specify these attribute categories:
  - User – In the attributes defined on the logged-in user.
  - Tenant – In the attributes defined on the organization of 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 filter searches attributes hierarchically and uses the value for the first attribute it finds with the given name. This search starts with the logged-in user, then proceeds to the user's organization and parent organization, and finally to the server level.

For join expressions, calculated fields, and pre-filters, the following syntax is used:

- `attribute('attributeName', 'server')` for a server-level attribute.
- `attribute('attributeName', 'tenant')` for an organization-level attribute.
- `attribute('attributeName', 'user')` for a user-level attribute.

If no level is specified, the server will search for the attribute hierarchically, starting at the `'user'` level:

- `attribute('attributeName')`

> The simplified syntax is used in locations that support the internal Domain syntax, DomEL. See **Chapter 7, "Domain Expression Language (DomEL)," on page 105**.

For schema names and derived tables, the attribute must be escaped with curly brackets ({}):

- `{attribute('attributeName'), 'server'}` for a server-level attribute.
- `{attribute('attributeName', 'tenant')}` for an organization-level attribute.
- `{attribute('attributeName', 'user')}` for a user-level attribute.
- `{attribute('attributeName')}` to search for an attribute hierarchically.

For more information on attributes, see the section "Managing Attributes" in the *JasperReports Server Administrator Guide*.

### 4.5.2 Things to Remember When Using Attributes

Note the following when using attributes:

- The attribute value must match the expected data type. For example, if you are creating a calculated field of type Integer, the attribute value must be an Integer.
- Attribute collections are not supported in the main Domain design. The attribute must be a single value. Attribute collections can be used in Domain security, however.
- If you are using attributes for the schema, make sure that the tables and columns referenced in the Domain are available for every possible value of the attribute. See **4.6.2, "Changing the Data Source," on page 56** for more information.

> If you use an attribute in your data source definition, you must ensure that referenced schemas, tables, and columns will be available. Data source attributes are defined outside the Domain Designer. See the section on defining attributes in the *JasperReports Server Administrator Guide* for more information.

- If a specified attribute does not exist, any item that uses it will fail. For example, if you specify the attribute `Country` for a schema name, and a `Country` attribute does not exist anywhere on the server, you will be unable to load the schema. This is similar to referring to column that does not exist.
- If an attribute exists but is not defined (empty) for this particular user, the behavior depends on where the attribute is used. An empty attribute can occur, for example, if you have set up a `Country` attribute, but the current user has no value defined for `Country`.
  - For pre-filters, an empty attribute is interpreted based on the field's data type as described below.
    - For a field of type String, an empty attribute is interpreted as an empty string: `""`
    - For a Boolean field, and empty attribute is interpreted as `FALSE`.
    - For numeric and date fields, an error will occur.
  - For derived table queries and custom joins, the attribute will fail in most cases. However, in some cases, JasperReports Server will consider the resulting expression to be valid, with potentially unexpected results, so be cautious when using attributes in this part of your design.
  - For schema names, the default schema is used. The default schema must be defined in the XML Domain design file. See **6.2.5.1, "Default Schema," on page 71** for more information.

### 4.5.3 Using Attributes for Pre-Filters

For example, you can use attributes to create a pre-filter for a country. Instead of a static filter that filters out data for every country except the USA, you could create a filter that only shows data for the logged-in user's country. To do this, you would create a Country attribute for your users in their profiles and enter the name of their country,

which must match one of the country names used in the Domain's data source. Then, when creating the pre-filter, you can use the parameter `attribute('Country')` to filter out data for all countries except the user's country.

For pre-filters, the data type for the attribute is automatically based on the filtered column's type. For example, if you are creating a filter based on a Date column, the attribute parameter will return a Date value.

### 4.5.4     Using Attributes for Schema Names

You can use attributes to choose a schema in your database. For example, suppose you have schemas named dev and qa in your database. You could create a server-level attribute and set it to `dev` on one server and `qa` on another. You can then use the attribute for the database schema in your Domain. Similarly, in a multi-organization deployment, you could have different schemas for different organizations, and set an organization-level attribute to choose which schema to access.

The Domain Designer needs a concrete schema to read and display tables. When you use an attribute for a schema name, the Domain Designer automatically substitutes the current value of the attribute. For the example above, if you use the Domain Designer on a server where the attribute is set to `dev`, the Domain Designer displays the `dev` schema in the Selected Schemas list and in the Select Data panel. You will see the tables and columns in the `dev` schema in the Domain Designer UI. Any joins, pre-filters, or presentation sets you create are based on that `dev` schema.

- To find the underlying attribute for a schema, go to the Data Management tab, select the database in the Data Structure panel, and then click on the schema name in the Selected Schemas list. The attribute details appear in the **You may also use an attribute for the schema name:** text box.

> When you use an attribute for a schema name, the Domain will fail if the schema is missing. You can optionally set a default schema in the Domain design file. The default schema is used whenever the schema is missing or undefined. See **6.2.5.1, "Default Schema," on page 71** for more information.

## 4.6     Modifying a Domain

You can add, change, and delete domain components.

> Use caution when editing Domains that have been used for Ad Hoc views or Domain Topics. By default, if you remove or change Domain elements that are used in an Ad Hoc view or Topic, any dependencies are removed the next time the Ad Hoc view or topic is opens. In some cases, you might want to create a placeholder field to preserve the structure of the Ad Hoc view.

**To edit a Domain:**

1. Log into the server as an administrator and navigate to the Domain location in the repository.
2. Right-click the Domain and select **Edit** from the context menu.

   The Domain Designer opens on the Data Presentation tab. Click the tab with the information you want to edit and make your changes.

3. If you want to change the data source, select **Replace Data…** from the ⋮ menu, select the data source you want in the Choose Data dialog, and click **OK**.

   If you change to a data source with a different database, the definitions in the Domain design may become invalid and you can't save the Domain. See **4.6.2, "Changing the Data Source," on page 56** for more information.

> Before you switch the data source for a Domain, back up the Domain by exporting a Domain design file.

4.   Save the Domain.

After modifying a Domain, you must clear the Ad Hoc cache of all queries based on the Domain. This removes any data that was based on the old instance of the Domain and avoids inconsistencies in new reports. For instructions, see the *JasperReports Server Administrator Guide*.

## 4.6.1    Removing Schemas, Tables, and Joins

If you attempt to remove a schema, table, or join that is used in the Domain Designer, a warning dialog lists the affected components of your design. Click **Delete Items** to continue and delete all the items listed, or click **Cancel** to keep the original data source.

If a schema or table is used in a join tree, the individual joins that reference the table are deleted, but other joins in the join tree are preserved. This may affect your design. You should examine the updated join tree to see how it has been affected.
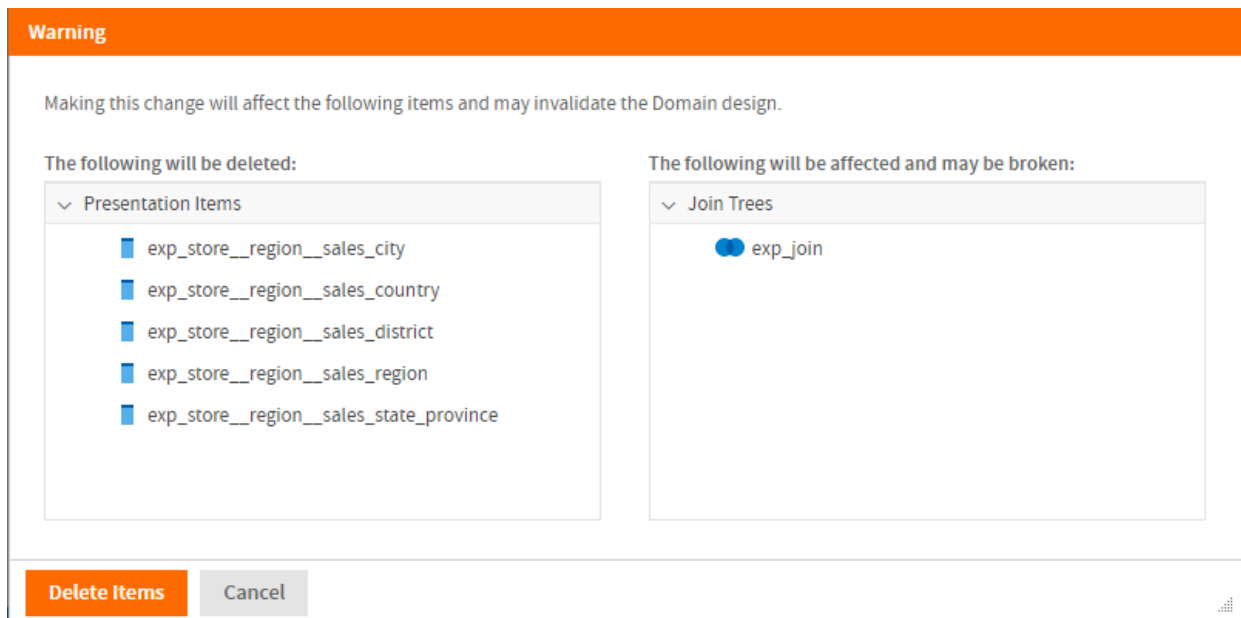


**Figure 4-13**  Deleted Table Warning

## 4.6.2    Changing the Data Source

When you open an existing Domain with the Domain Designer, the Domain Designer reads the current state of the database and uploads it to the design. The Domain Designer does not detect changes to the database schemas, tables, and columns in real-time, but it will detect them the next time you open the Domain in the Domain Designer. The data source or structure can change for any of the following reasons:

•   You manually change the JasperReports Server data source by selecting Replace Data from the menu.

•   The selected data source or a schema uses an attribute and the attribute changes or you log in with a different attribute value.

- The structure of the underlying database or other data source has changed. For example, if a table or column is added to or deleted from the database, it is detected the next time you open the Domain Designer.

Missing schemas, tables, or columns in the new or updated data source are automatically deleted from the design along with any elements, such as calculated fields, joins, or sets, that depend on them.

Because the Domain design relies extensively on the data source, changing the data source makes sense only in certain cases. Examples include:

- Switching to a data source with a different schema but with the same tables, for example, when moving from staging to production.
- Switching from a regular data source to a virtual data source that contains the original data source. In this case, JasperReports Server attempts to locate the original data source inside the virtual data source and create the correct prefix. Derived tables and calculated fields are not preserved.
- Switching from a virtual data source to one of the underlying data sources. In this case, any resources that are not in the selected data source are deleted along with any dependent information, such as derived tables, joins, calculated fields, or labels that depend on the deleted resources.

Before you select a new data source manually or use attributes to specify data sources and/or schemas, do your best to ensure that the schema structures are identical. You should also export your Domain design before changing the data source manually.

### 4.6.2.1  Selecting a New Data Source

> If you have modified your database tables, fields, or schema, use **Replace Data** and select your current database to upload the changes.

To select a new data source:

1. Select **Replace Data…** from the ⋮ menu.
2. Select the data source you want in the Choose Data dialog, and click **OK**.

   The Select Schemas to Map dialog appears. This dialog shows the schemas used by your Domain on the left, and the schemas available in the data source on the right.
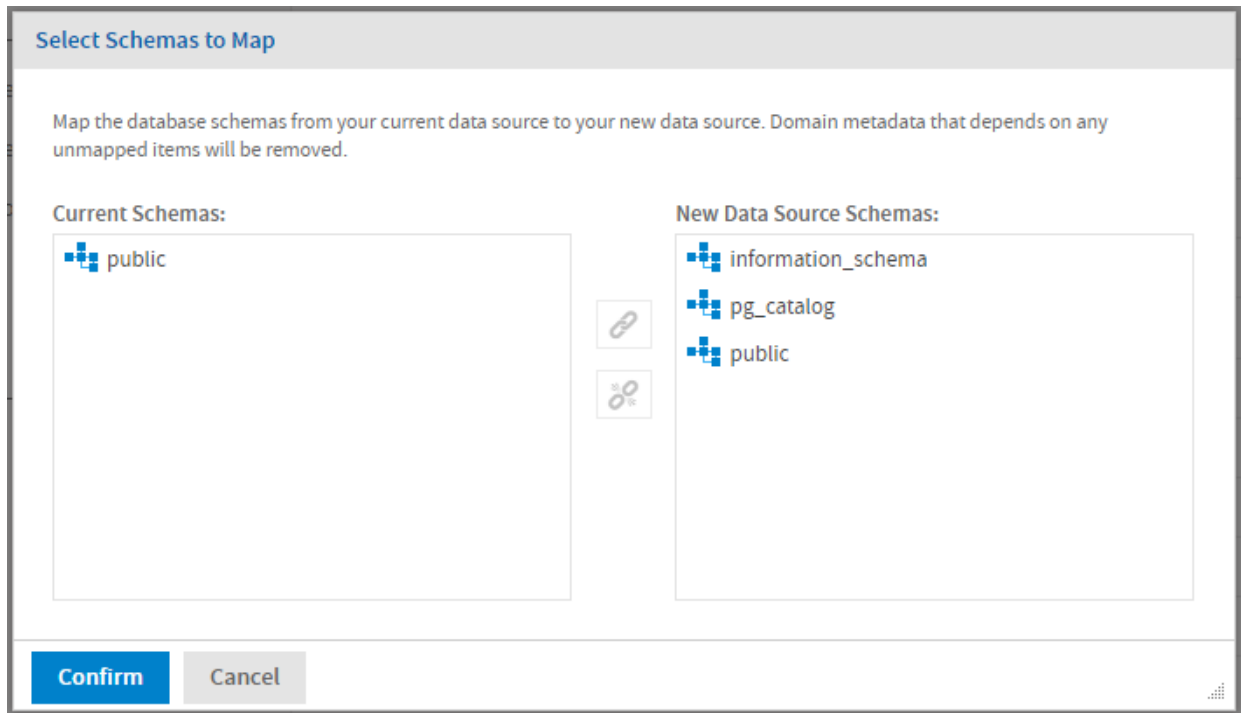
**Figure 4-14  Selecting schemas to map**

> If you are switching between a schemaless data source (such as MySQL) and a schema-based data source (such as Oracle), the dialog is different, as follows:
>
> • If the original data source is schemaless and the new data source supports schemas, the Current Schemas list is omitted. Select a new schema from the available schemas.
> • If the original data source has schemas and the new data source is schemaless, the New Data Source Schemas list is omitted. Select one of the schemas in your Domain to use for the new data source. In addition, if your Domain has only one schema and you are selecting a schemaless data source, the schema is mapped automatically and you do not see this dialog.

3. To associate an existing schema in your Domain with a schema in the data source:

   a. Select the schema in the Domain in the Current Schemas list.

   b. Select the schema you want to replace it with in the New Data Source Schemas list.

   c. Click 🔗 .

      The schemas are associated. A number is added after the schema name to make it easier to manage multiple schemas.

4. Repeat the previous step for each schema you want to map. If you want to unlink two schemas, click ✂ .

5. Click **Confirm** to map the schemas.

   JasperReports Server retrieves the data structure from the data source and validates the Domain. If all the tables and columns in your Domain are found, the Domain Designer opens at the Data Management tab.

6. If there are missing tables or columns, a warning dialog lists the affected components of your design. Click **Delete Items** to continue and delete all the items listed, or click **Cancel** to keep the original data source.
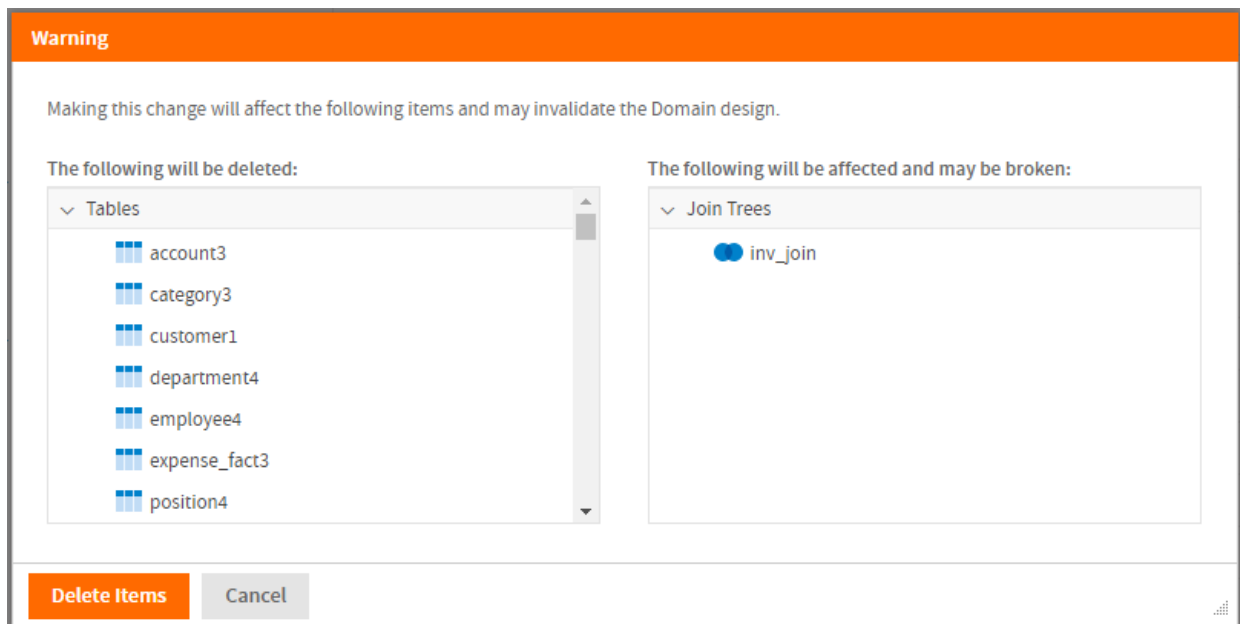
**Figure 4-15** **Warning dialog for changing data source**

If you click **Delete Items**, JasperReports Server deletes all the displayed items. There may still be some errors you need to resolve before the Domain is valid.

### 4.6.2.2 Changes Due to Attributes

If you use attributes to define a database schema, a calculated field, or another Domain element, you may see the following when you log in as a different user:

- If a schema is missing, the Select Schemas to Map dialog appears. See **4.6.2.1, "Selecting a New Data Source," on page 57** for more information. This can happen because the attribute is no longer defined or because the attribute is defined but resolves to a non-existent schema.
- If you use an attribute for the schema name and you later select a different schema in the Select Schemas to Map dialog, the design will always use that schema; the schema will no longer be attribute-based.
- If any tables or fields are missing in the new data source or schema, JasperReports Server displays a warning dialog. Clicking **Delete Items** deletes the missing tables and columns and any Domain design elements that depend on them.
- If an attribute is not defined or does not resolve to a usable value, JasperReports Server displays a warning dialog. Clicking **Delete Items** deletes any Domain design elements that depend on the undefined or invalid attribute(s).

> If you use an attribute in your data source definition, you may see similar problems. Data source attributes are defined outside the Domain Designer. See the section on defining attributes in the *JasperReports Server Administrator Guide* for more information.

### 4.6.2.3 Updating an Existing Data Source

To update a Domain after changes to the underlying data structure, do one of the following:

- Refresh the browser.
- Close the Domain Designer then open the Domain for editing again.

---

- Select **Replace Data…** from the ⋮ menu and select your current data source.

Keep the following points in mind regarding data source updates:
- You will be prompted to remap any schemas no longer found in the database.
- If your design uses tables and columns that are no longer found in the data source, JasperReports Server displays a warning dialog. Clicking **Delete Items** deletes the missing tables and columns and any Domain design elements that depend on them.
- New schemas, tables, and columns appear in the Data Structure panel on the Data Management tab. JasperReports Server has no way to automatically map schemas, tables, or columns that have been added since the last time you edited the Domain.
- To add a new column to the Domain, move its table to Selected Tables. If the table is already selected, the column is automatically available. Table selection works only with entire tables.

## 4.6.3   Domain Validation

Validating a Domain ensures the consistency of all its components. The Domain Designer checks the syntax of files when they're uploaded, and overall consistency is also checked when saving a new or edited Domain.

During validation, the Domain Designer does the following:
1. Verifies that the tables and columns of the Domain design exist in the data source (validation against data source).

> In special cases where you need to create a design before the data source is available, this step can be omitted by setting a parameter in the server configuration file. See the *JasperReports Server Administrator Guide*.

2. Verifies that all items in each defined set originate in the same join tree.
3. Verifies that all items reference existing columns.
4. Verifies that derived tables have valid SQL queries.
5. If a security file has been uploaded, verifies that all items and sets in the security file exist in the Domain design.

If validation fails, you will not be able to save the Domain. Make the necessary changes to the settings and save again. If the settings are in the uploaded files, edit the files and upload them again.

Validation occurs at the following times:
- When you open a Domain for editing: JasperReports Server validates the Domain syntax and validates the Domain against the data source.
- When you replace the data source: JasperReports Server validates the Domain syntax and validates the Domain against the data source.
- When you upload a schema from an XML file: JasperReports Server validates the Domain syntax and validates the Domain against the data source.
- When you save the Domain: JasperReports Server validates the Domain syntax and verifies permissions.

When validation fails a message appears to help you correct the error.

## 4.7        Importing and Exporting Domain Design Files

You can export and import Domain design files in XML format. This lets you copy or back up Domains, even between different servers. Domain design files can be also created or edited manually in a text editor or programmatically.

The results of editing or using a Domain based on an inconsistent or incorrect design file are unpredictable. Before you modify Domain design files you need to be familiar with the Domain syntax and structure. For more information, see **Chapter 5, "Working with Domain Files," on page 63** and **Chapter 6, "XML Design File Reference," on page 67**.

### 4.7.1      Exporting a Domain Design File

1.  Log into the server as an administrator and edit an existing Domain or create a new one.

2.  Save any changes you have made to the Domain.

3.  Click on the menu bar to export a Domain schema file, schema.xml, in XML format.

### 4.7.2      Uploading a Design File to a Domain

After you have modified an XML design file, you can upload it through the Domain Designer. Alternatively, you can create a new Domain based on a modified file or even on a design file created from scratch.

Uploading a design file overwrites the existing Domain and may break Ad Hoc views, Topics, or reports that depend on the Domain.

**To upload a Domain design file:**

1.  Log into the server as an administrator and edit an existing Domain or create a new one.

2.  Click on the menu bar.

3.  **Browse** to find the design file you want to upload, select it, and click **Open**.
    A warning dialog appears.

4.  Click **Update Domain Design** to continue.

> The design file overwrites any existing design without prompting. If you make a mistake or upload the wrong file, exit the Domain Designer without saving and start over.

The server validates the syntax of the uploaded file. If there are syntax or semantic errors, the current design is not replaced. You may also see warnings if you have changed the data source, schema names, or deleted schemas, tables, or columns from the Design file. If there are any errors or inconsistencies, you should make changes to the design file, upload it again, and verify it again.

The results of editing a design in the Domain Designer based on an inconsistent XML file are unpredictable. If you can't resolve problems, or are unsure that the result will be what you want, exit the Domain Designer without saving.

5.  After the design appears correctly in the Domain Designer, make any further modifications on any of the tabs.

6.  Select **Save Domain** from the menu to update the Domain in the repository.

7.  If you modified an existing Domain, you must clear the Ad Hoc cache of all queries based on the Domain. This removes any data that was based on the old instance of the Domain and avoids inconsistencies in new reports. For instructions, see the *JasperReports Server Administrator Guide*.

## 4.7.3    Exporting a Domain and Its Resources

The export option on the Domain Designer menu exports the Domain design file, which describes all the features visible in the Domain Designer. This is a good option if you want to modify the Domain in a text editor. However, a Domain in JasperReports Server needs additional information, including: permissions, the data source in the repository, the security file (if it exists), and any localization bundles.

For example, to duplicate a Domain on another instance of JasperReports Server, export it from the repository to create an export catalog file that can be moved to another server. All necessary files are assembled into a zip that can be easily transmitted and uploaded. You can then modify the Domain in the Domain Designer on the destination server to account for any small differences.

**To export a Domain with all its associated resources::**

1. Log into JasperReports Server as an administrator.
2. Browse in the repository to the Domain you want to export.
3. Right-click on the Domain and select **Export** from the context menu. The Export Resources dialog appears.
4. Enter a name for the exported file, select the options you want, and click Export. The file is exported in .zip format to the Downloads folder set by your browser.

For more information about exporting and importing repository files, see the *JasperReports Server Administrator Guide*.

# CHAPTER 5    WORKING WITH DOMAIN FILES

The Domain Designer offers a graphical interface for creating and editing Domains, but you can also export a Domain in XML format and edit it outside of JasperReports Server. Some features, such as default schemas and joins between two columns in the same table, can be configured only in the design file. In other cases, it may be easier to work directly with the XML to accomplish certain tasks. Design files can also simplify the sharing of Domains between systems.

> The results of using a Domain based on an inconsistent or incorrect design file are unpredictable and can be difficult to resolve. Make sure you thoroughly understand Domains and Domain syntax before editing any Domain design files.

Domain security and translation of Domain labels and descriptions are also configured via optional text-based files that you upload to the Domain. Security definitions and locale translations take effect in the Ad Hoc Editor when creating the report and in the final output when running the report.

This chapter provides an overview of XML Domain design files and best practices for working with them:

*   **Understanding Domain Design Files**
*   **Requirements of XML Design Files**

Details about XML design files, security files, and locale bundles are given in the following chapters:

*   **XML Design File Reference** - The syntax of all elements in the XML design file, such as the data source, tables, joins, sets, and items.
*   **Securing Data in a Domain** - A sample scenario that documents how to create a security file.
*   **Localizing Domains** - How to create and upload locale bundles to translate labels and descriptions appearing in Domain-based reports.

## 5.1    Understanding Domain Design Files

When you design a Domain, you specify the tables in the data source, derived tables, joins, calculated fields, and filters, as well as how those elements appear to users. You can design a Domain in JasperReports Server interactively through the Domain Designer, or you can use the file-based format to create or modify Domains. A text file containing a Domain design in XML is called a Domain design file.

The XML in a design file is a hierarchy of elements and attributes that specifies all the settings in the Domain. The elements and attributes are defined by an XML schema provided in an XSD file. In addition, there are constraints on a Domain design that are not expressed in the XML schema. A design file can be modified or written from

scratch in an editor and uploaded to the server, as long as it conforms to the XML schema and the design constraints.

For information about exporting and uploading design files, see **4.7, "Importing and Exporting Domain Design Files," on page 61**.

> XML is not the native format of the Domain design. The XML file is only a representation from which the design can be inferred. A design has additional constraints that are not mapped in the XML format.

## 5.1.1 Design File Use Cases

Due to the complexity of a valid design file, it is usually much easier to begin with a basic design file exported from the Domain Designer or to modify an existing design file. Common use cases for working with design files include:

- **Enhancing a design** – You can use the design file to add features that can't be defined in the Domain Designer. To do this, use the Domain Designer to define as much of the design as possible, then export the design file and modify the exported file directly. Some features can only be defined in the design file. You can still upload these files to the Domain Designer and you can modify the other features of the Domain.
- **Defining presentation for large Domains** – The Domain Designer makes it easy to select all tables and columns and expose them as sets and items, but editing the labels and descriptions of dozens of items may be faster when they appear in a single design file.
- **Making repetitive changes** – If the database changes or you want to move a design file to a different system, you may need to edit many elements in the same way. Using search-and-replace in an external editor can speed up this process.
- **Programatically creating Domain designs** – If you are creating many similar Domains, you may be able to start with a valid design file and generate similar design files that meet the constraints of JasperReports Server.
- **Creating files for security and translations** – These optional files refer to elements of the Domain design, and it is often more convenient to copy-paste them between external files. For translation files, you can export a template from the Domain Designer as a basis for the translation files. See **4.7, "Importing and Exporting Domain Design Files," on page 61**.

After editing, a design file can be uploaded, validated, and used to define the design of a new or existing Domain. With a few exceptions, when you open the design in the Domain Designer again, the modifications are editable in the Designer. For example, a description you added in the XML design file appears on the Data Presentation tab and you can edit it in the Designer. Other elements of the XML file appear on some or all of the Domain Designer tabs.

## 5.1.2 Features That Cannot Be Defined in the Domain Designer

The following elements can be edited only in the design file. Design files with these elements can still be opened in the Domain Designer, and other elements can be edited:

- Joins that use `or` or `not` between two join expressions. These appear in the Domain Designer as read-only joins. See **4.4.7, "Read-Only Joins," on page 50** for more information.
- Joins between two columns in the same table. These appear in the Domain Designer as read-only joins. See **4.4.7, "Read-Only Joins," on page 50** for more information.
- The `defaultSchema` property when using JasperReports Server attributes for a schema name. This does not appear in the Domain Designer directly, but is used when the attribute exists but has null value. See **6.2.3, "schemaMap," on page 70** for more information.

- Individually-selected columns of a table. You can only select whole tables on the Data Management tab of the Domain Designer.

## 5.2 Requirements of XML Design Files

The relationship of item definitions, column definitions, and actual database columns is the essence of the Domain itself and must be maintained when editing the design file. In order to be uploaded to a Domain, a design file must meet the following conditions:

- It must be well-formed XML. This means that all syntax, spelling, and punctuation is correct so that the file contains a hierarchy of elements, attributes, and content.
- The following symbols can't be entered directly; they must be escaped using their HTML encoding:
    & (&amp;), " (&quot;), < (&lt;), > (&gt;);
- It must be valid with regard to the XML schema. The XML schema defines allowed XML element and attribute names and how they're nested in a hierarchical structure. This ensures that the elements and attributes are those used by JasperReports Server. The allowed versions of the XML schema of a Domain design are given in the XSD files located in <install-dir>/samples/domain-xsd/. This document describes the functionality supported by the following version:
    <install-dir>/samples/domain-xsd/schema_1_3.xsd
- The design file must be internally consistent. You can upload a partial file, but to use the Domain, you must define all the necessary elements of a Domain design. These constraints cannot be expressed in the XSD file because they are outside the scope of an XML schema.

> It is possible to create or upload partial Domain files, but to be usable, a Domain file must include certain elements, such as: a data source, one or more tables, and one or more presentation items.

- The tables and columns in the design must be consistent with their external definition in the data source of the Domain. The design must reference valid table and column names in the data source; in particular, table names for a design based on an Oracle RDBMS must include the database schema name. The data source also defines the datatype of a column, and the design must use that column accordingly. As a result, a design file is specific to a given data source and may fail when used in a Domain with a different data source.

The more complex elements of a design file have further constraints:

- SQL queries for a derived table must be valid with respect to the JDBC driver for the data source. The tables and columns in the query must exist in the data source, and the columns in the results must match those declared in the design.
- For a virtual data source that combines data from data sources with different JDBC drivers, SQL queries are validated against Teiid SQL; for more information, see the Teiid Reference Guide under the Documentation link on the Jaspersoft Support Portal.
- Expressions for filters and calculated fields must be valid programmatic expressions in Jaspersoft's Domain Expression Language (DomEL). This format is documented in **"Domain Expression Language (DomEL)" on page 105**.

> The design of a Domain is stored internally in the repository. The XML is only a representation from which the design can be inferred, and it may have some validity errors that cannot be detected. As a result, the Domain resource and the XML may not remain totally synchronized through several cycles of exporting to XML, editing, and uploading. For example, the Domain Designer sometimes renames the result of a join (JoinTree_ 1), which affects the security file.
>
> However, the Domain Designer also has limitations and cannot create some valid designs. For example, in a design file, you may select the columns of a table whereas you can only select whole tables on the Tables tab.

A design file is plain text and can be edited in any text editor. The server exports well-formatted XML, and if you want to make only a few changes or simple additions, a text editor is sufficient. For editing the content of the design file, a specialized XML editor ensures that the design file is well-formed, so you don't introduce other errors. If you want to make structural changes or write a design file from scratch, use an XML editor that understands the XML schema in the schema_1_3.xsd file. By loading both the XML and XSD files, this type of XML editor lets you insert elements and attributes only where they're allowed to ensure that the design file is valid.

However, no editor can enforce the consistency requirements in a design file. The next chapter gives a complete XML reference for the elements and attributes of a design file and the various constraints that apply to each of them.

# CHAPTER 6     XML DESIGN FILE REFERENCE

This section explains each of the XML elements and attributes in a design file and how they relate to the settings in the Domain Designer.

> Because certain XML elements correspond to objects in the Domain design, this section refers to XML elements that *contain* Domain objects such as sets. This is a short-hand description that means the XML elements contain other XML elements that represent the Domain object.
>
> Technically, the only requirement to save or upload a Domain design file is a data source reference. However, for the file to be usable, many more elements are needed. As a short-hand, elements you need to create a usable Domain are marked "required".
>
> The following symbols can't be entered directly; they must be escaped using their HTML encoding:
>
> **&** = `&amp;`      **"** = `&quot;`      **<** = `&lt;`      **>** = `&gt;`

## 6.1     The schema Element

The `schema` element is the outermost container element of an XML Domain design file. The `schema` element is required.

> The `schema` element refers to the XML schema. It has no relationship to database schemas in the data source.

### schema Hierarchy

The following hierarchy is used for the `schema` element:

```
<schema xmlns="http://www.jaspersoft.com/2007/SL/XMLSchema" version="1.3"
       schemaLocation="schema_1_3.xsd">
       <dataIslands> (0..1)
       <dataSources> (0..1)
       <itemGroups> (0..n)
       <items> (0..n)
       <resources> (1)
```

When you export a file from the Domain Designer, these elements appear alphabetically in the file. In this reference, they are presented in the following order, based on their function in the Domain:

- Data sources and database schemas (`dataSources` element).
- Domain tables and rows (`resources` element); includes tables, derived tables, calculated fields, joins, and pre-filters.
- Domain presentation (`dataIslands`, `itemGroups`, and `items` elements).

## Child Elements

| Element Name | Description |
|---|---|
| `<dataIslands>` | Container for a list of data islands in the Domain design. |
| `<dataSources>` | Container for the Domain data source and schemas. |
| `<itemGroups>` | Containers for sets; each data island listed in the `dataIslands` element must have a corresponding `itemGroups` element. |
| `<items>` | Containers for items that aren't in any set. Items in different `dataIslands` must be in different `items` elements. |
| `<resources>` | (Required) Container for all elements that define tables and rows in the Domain, including:<br><br>• Tables<br>• Derived tables<br>• Joins<br>• Calculated fields<br>• Pre-filters<br><br>Because the elements under `resources` refer to database objects, they must be externally consistent with the data source for the Domain. |

## XML Attributes

| Attribute | Type | Description |
|---|---|---|
| schemaLocation | String | Sometimes added by XML editors to locate the XSD file. |
| version | String | Version of the XSD used to create this design file. Included in design files exported from Domain Designer. |
| xmlns | String | XML namespace URI. This string is unique to Jaspersoft, and it does not correspond to a valid URL. For more information, see http://www.w3.org/TR/REC-xml-names. Included in design files exported from Domain Designer. |

## 6.2 Representing Data Sources and Database Schemas in XML

Data sources and database schemas are declared under the `dataSources` element. A data source for a Domain is declared using `jdbcDataSource` and a data source for a Topic is declared using `jrQueryDataset`.

## 6.2.1    dataSources

The `dataSources` element is a container for the `jdbcDataSource` element. A Domain can only have at most one data source. `dataSources` is a child of the top-level `schema` element.

`dataSources` is not required for schemaless JDBC databases such as MySQL unless you are using an attribute for the database schema name.

### Child Elements

One of the following elements is required for `dataSources`. Only one of these can be declared.

| Element Name | Description |
| --- | --- |
| <**jdbcDataSource**> | Declares a data source for a Domain and the id used to reference it. |
| <**jrQueryDataset**> | Declares a data source for a Topic and the id used to reference it. |

## 6.2.2    jdbcDataSource

The `jdbcDataSource` element declares a data source for a Domain and is a container for the `schemaMap` elements used in the Domain. `jdbcDataSources` does not directly reference a data source in the repository; instead, the actual link to the repository data source is attached as metadata to the Domain design in the repository.

For Topics, use **6.2.7, "jrQueryDataset," on page 72**. There can only be one `jrQueryDataset` or `jdbcDataSource` element in a Domain design file.

### jdbcDataSource Hierarchy

The following hierarchy is used to represent JDBC data sources and their database schemas. This hierarchy is under the `schema` element.

```
<dataSources> (1)
  <jdbcDataSource> (1)
    <schemaMap> (1...n)
      <entry> (1)
        <string> (1)
```

### Child Elements

| Element Name | Description |
| --- | --- |
| <**schemaMap**> | (Required) Declares a database schema used in the Domain. |

### XML Attributes

| Attribute | Type | Description |
| --- | --- | --- |
| id | String | (Required) Unique identifier for the data source in the Domain design file. |

## 6.2.3 schemaMap

The `schemaMap` element is a container for `entry` elements that represent database schemas used in the Domain.

### Child Elements

| Element Name | Description |
|---|---|
| `<entry>` | A database schema or the default schema declaration. |

## 6.2.4 entry

The `entry` element in the schemaMap element declares a database schema for use in the Domain. The actual name of the schema is the value of the `string` child element, while the identifier used in the Domain is set as the `key` attribute.

If `key` is set to `defaultSchema`, the `entry` element is used to define the default schema.

### Child Elements

| Element Name | Description |
|---|---|
| `<string>` | The name of the schema in the database. |

### XML Attributes

| Attribute | Type | Description |
|---|---|---|
| key | String | (Required) Unique identifier for the schema in the Domain design file or the reserved string "`defaultSchema`". <br><br> For regular database schemas in the Domain, the `key` attribute is used by other elements in the presentation representation to identify the schema, via the `referenceId`. <br><br> If `key` is set to `defaultSchema`, this `entry` element is used to define the default schema. See **6.2.5.1, "Default Schema," on page 71** for more information. <br><br> The `key` XML attribute can contain alphanumeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. |

## 6.2.5 string

The `string` element in the `schemaMap` hierarchy is a string defining the database schema to be used in the Domain. The string can be one of the following:

- The exact name of a schema in the database.
- A JasperReports Server attribute. Not supported for `defaultSchema`. See **6.10, "Using Server Attributes in Design Files," on page 100** for more information.

### 6.2.5.1  Default Schema

If the `key` attribute of the parent `entry` element is set to `defaultSchema`, then `entry/string` declares the default schema for the Domain. The default schema is a fallback schema name that is substituted when a schema name is missing or not available. This can happen, for example, if you are using a JasperReports Server attribute for the data source or schema. It can also happen if you change the data source for some reason. You can only define one default schema for a database.

## 6.2.6  Examples

The following example shows a declaration of a data source and two schemas. Note that the data source name in the Design file does not need to be linked to the data source in the repository. That linkage is done outside of the Domain design file.

> Note that in the `key` attribute, an underscore (_) has been substituted for unsupported characters. Periods (.) are not supported in schema key attributes; numbers are supported, but not in the first character.

```
<dataSources>
  <jdbcDataSource id="myDataSource">
    <schemaMap>
      <entry key="schema_1">
        <string>schema.1</string>
      </entry>
      <entry key="_234schema">
        <string>1234schema</string>
      </entry>
    </schemaMap>
  </jdbcDataSource>
</dataSources>
```

The following example shows a file that uses an attribute for the database schema name, and sets a default schema name to use when the attribute is null.

```
<dataSources>
  <jdbcDataSource id="dsFoodMart">
    <schemaMap>
      <entry key="_attribute_schemaAttr_">
        <string>{attribute('schemaAttr')}</string>
      </entry>
      <entry key="defaultSchema">
        <string>public</string>
      </entry>
    </schemaMap>
  </jdbcDataSource>
</dataSources>
```

> If you use an attribute for the database schema name, and the attribute is undefined on the server, the Domain will fail.

## 6.2.7    jrQueryDataset

The `jrQueryDataset` element declares the data source for a Topic. `jrQueryDataset` does not directly reference a data source in the repository; instead, the actual link to the repository data source is attached as metadata to the Domain design in the repository. There can only be one `jrQueryDataset` or `jdbcDataSource` element in a Domain.

`jrQueryDataset` does not support schemas. For more information about Topics, see the *JasperReports Server User Guide*.

### 6.2.7.1  Element Hierarchy for jrQueryDataset

The following hierarchy is used to represent JDBC data sources and their database schemas. This hierarchy is under the `schema` element.

```
<dataSources> (1)
  <jrQueryDataset> (1)
```

### XML Attributes

| Attribute | Type | Description |
|-----------|------|-------------|
| id | String | (Required) Unique identifier for the data source in the Domain design file. |

## 6.3    resources

The `resources` element contains other elements that define the tables and rows that are accessed through the Domain. It corresponds to the first three tabs of the Domain Designer: Data Management, Joins, and Pre-filters.

The `resources` element contains the `jdbcTable` and `jdbcQuery` elements to represent database tables and derived tables, respectively. Join trees are represented as a `jdbcTable` element with additional contents to define the joins. In order for the Domain to be usable, there must be at least one table, whether as a jdbcTable or jdbcQuery element.

> Because the elements under `resources` refer to database objects, they must be externally consistent with the data source for the Domain. For example, the `schemaAlias` and `datasourceTablename` attributes of `jdbcTable` must match the schema and name of a table in the data source.

The `resources` element does not contain presentation elements. It is a direct child of the `schema` element.

### resources Hierarchy

The following hierarchy is used for the `resources` element:

```
<resources>
        <null> (0...1)
                <fieldList> (1)
                        <field> (1...n)
        <jdbcTable> (0...n)
                <fieldList> (1)
```

```
                           <field> (1...n)
                    <filterString> (0...n)
            <jdbcQuery> (0...n)
                    <fieldList> (1)
                           <field> (1...n)
                    <filterString> (0...n)
                    <query> (1)
            <jdbcTable> (0...n)
                    <fieldList> (1)
                           <field> (1...n)
                    <filterString> (0...n)
                    <joinInfo> (1)
                    <joinList> (1)
                           <join> (1...n)
                    <joinOptions> (0...1)
                    <tableRefList> (1)
                           <tableRef> (1...n)
```

### Child Elements

| Element Name | Description |
| --- | --- |
| jdbcTable | Represents one of the following:<br><br>• A table in the data source or table copy in the Domain (see **6.4, "Representing Tables in XML," on page 73**).<br>• A join in the Domain (see **6.6, "Representing Joins in XML," on page 79**) |
| jdbcQuery | (Only present if schema defines derived tables.) Represents a derived table that results from an SQL query. See **6.5, "Representing Derived Tables in XML," on page 76** for more information. |
| null | (Only present if schema defines constant calculated fields.) Container for constant fields. |

## 6.4   Representing Tables in XML

To represent tables, use the `jdbcTable` element. A table is a child of the `resources` element.

### Table Hierarchy

The following hierarchy is used for `jdbcTable` elements when representing a table from the data source.

```
<jdbcTable>
      <fieldList> (1)
             <field> (1...n)
      <filterString> (0...n)
```

### 6.4.1   jdbcTable

The `jdbcTable` element represents a table or a copy of a table in the data source. A Domain design must reference all the tables it needs to access, including tables that you need to construct the Domain but do not want to expose to

the end user. `jdbcTable` is a child of the `resources` element.

> The `jdbcTable` element is also used with different attributes to describe join trees. See **6.6, "Representing Joins in XML," on page 79**.

## XML Attributes

| Attribute | Type | Description |
|---|---|---|
| datasourceId | String | (Required) Identifier for the data source, as set in the `id` attribute of **jdbcDataSource**. When creating a design file, this alias may be any name you choose, but it must be identical for all tables and derived tables. When uploading the file, the `datasourceId` automatically becomes the alias associated with the data source defined for the Domain. |
| id | String | (Required) Unique identifier for the table in the Domain design file. If you copy a table to join it multiple times, each copy has the same `datasourceId`, `schemaAlias`, and `datasourceTableName` but must have a different `id`. To set a table `id` in the Domain Designer, right-click the table and select **Rename Table** from the context menu.<br><br>The `id` XML attribute can contain alphanumeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. When generated via the Domain Designer, `id` is a representation of the table name in the data source, with unsupported characters replaced by underscores. |
| schemaAlias | | Name of the database schema in the data source. Required if the data source uses schemas.<br>• For schemaless data sources, such as MySQL, `schemaAlias` is not required.<br>• For schema-based data sources, such as PostgreSQL and Oracle, the name is the literal name of the schema in the data source, for example, `public`.<br>• For schemaless data sources within a virtual data source, the schema name is of the form `dataSourcePrefix`, where `dataSourcePrefix` is the data source prefix defined when the virtual data source was created. For example, `FoodmartDataSource`.<br>• For schema-based data sources within a virtual data source, the name is in the form `dataSourcePrefix_schema`, where `dataSourcePrefix` is the data source prefix defined when the virtual data source was created. For example, `FoodmartDataSource_public`. |
| datasourceTableName | String | (Required) Literal name of the table in the data source. |

Child Elements

| Element Name | Description |
|---|---|
| <**fieldList**> | (Required) A container for the `field` elements in the table. Each `jdbcTable` element can contain only one `fieldList` element. |
| <**filterString**> | Expression that evaluates to true or false when applied to each row of values in the data source. For a table, the expression refers to columns using the `field_name` form of the column ID. See **6.7, "Representing Pre-filters in XML," on page 88** for more information. |

## 6.4.2   fieldList

`fieldList` is a container for one or more `field` elements.

In the Domain Designer, you can select only entire tables, not individual columns. In the XML design file, however, you can specify any subset of columns that you need.

Child Elements

| Element Name | Description |
|---|---|
| <**field**> | (Required) A column from the table specified in `jdbcTable`. |

### 6.4.3    field

The `field` element represents a column in the data source. The column must be from the table specified by `jdbcTable`. You must reference all the columns you use in the Domain, including columns that are used to construct the Domain but are not exposed to the end user.

XML Attributes

| Attribute | Type | Description |
|---|---|---|
| fieldDBName | String | Literal name of the column in the data source. |
| id | String | (Required) Identifier for the column. As in the JDBC model that the data source is based on, the `id` must be unique within the `jdbcTable`, but not necessarily within the Domain.<br><br>The `id` XML attribute can contain alphanumeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. When generated via the Domain Designer, `id` is a representation of the column name in the data source, where unsupported characters replaced by underscores. |
| type | String | (Required) The Java type of the column, as determined from the data source by the JDBC driver. The available types are shown below. |

#### 6.4.3.1  Supported Types

The following Java types are supported for the `type` attribute:

```
java.lang.Boolean       java.lang.Float       java.lang.String
java.lang.Byte          java.lang.Integer     java.math.BigDecimal    java.sql.Timestamp
java.lang.Character     java.lang.Long        java.sql.Date           java.util.Date
java.lang.Double        java.lang.Short       java.sql.Time
```

Unless you know the name and type of every column in the data source, it is often easier to select the tables you want in the Domain Designer and then export the XML design file. The Domain Designer accesses the data source to find the names of all tables and columns, as well as their types. You can then modify the exported design.

> If you have proprietary types in your database, the server may not be able to map its Java type from the JDBC driver. You can configure the mapping for proprietary types, as described in the *JasperReports Server Administrator Guide*.
>
> Alternatively, you can override any mapping by specifying the `type` attribute for any given field in the XML design file. The server uses this Java type for the field, regardless of its mapping. If your proprietary type can't be cast in the specified type, the server raises an exception.

## 6.5    Representing Derived Tables in XML

To represent derived tables, use the `jdbcQuery` element. This element is very similar to the **jdbcTable** element for tables, but contains an additional query element, used to represent the query for the derived table. `jdbcQuery`

is a child of `resources`.

### Table Hierarchy

The following hierarchy is used for `jdbcQuery` elements.

```
<jdbcQuery>
      <fieldList> (1)
             <field> (1...n)
      <filterString> (0...n)
      <query> (1)
```

## 6.5.1   jdbcQuery

The `jdbcQuery` element represents a derived table that results from an SQL query. `jdbcQuery` is similar to the jdbcTable element for a table, but has an additional child, `query`, used for the derived table query. `jdbcQuery` is a child of `resources`.

### XML Attributes

| Attribute | Type | Description |
|---|---|---|
| datasourceId | String | (Required) Identifier for the data source, as set in the `id` attribute of **jdbcDataSource**. When creating a design file, this alias may be any name you choose, but it must be identical for all tables and derived tables. When uploading the file, the `datasourceId` automatically becomes the alias associated with the data source defined for the Domain. |
| id | String | (Required) Unique identifier for the derived table in the Domain design file. The `id` of a derived table is used everywhere the `id` of `jdbcTable` is used. If you copy a derived table to join it multiple times, each copy must have a different `id`.<br><br>The `id` XML attribute can contain alphanumeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. |

### Child Elements

| Element Name | Description |
|---|---|
| **<fieldList>** | (Required) A container for the **field** elements in the derived table. A `jdbcQuery` element can contain only one `fieldList` element. |
| **<filterString>** | Expression that evaluates to true or false when applied to each row of values in the data source. For a derived table, the expression refers to columns using the `field_name` form of the column ID. See **6.7, "Representing Pre-filters in XML," on page 88** for more information. |
| **<query>** | (Required) The SQL query sent to the database server. |

## 6.5.2    fieldList

The `fieldList` is a container for the `field` elements in the table. When the derived table is created in the Domain Designer, the set of columns corresponds to the selection of columns in the query result on the Derived Tables tab.

### Child Elements

| Element Name | Description |
|---|---|
| `<field>` | (Required) A column from the table specified in `jdbcQuery`. |

## 6.5.3    field

The `field` element represents in the results of the query. A `field` element of a derived table must reference a column that is returned by the query. Only the columns represented by a `field` element are available for reference by other elements.

### XML Attributes

| Attribute | Type | Description |
|---|---|---|
| id | String | (Required) Literal name of the column in the query result. If the query gives an alias to the column in a `SELECT AS` statement, the `id` is the same as the alias. The `id` must be unique within the query results, but not necessarily within the Domain. |
|  |  | The `id` XML attribute can contain alphanumeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. |
| type | String | (Required) The Java type of the column, as determined from the data source by the JDBC driver. The available types are shown in **Supported Types** |

## 6.5.4    query

The `query` element contains the SQL query that is sent to the database server. You can use any valid SQL that returns results, as long as the tables and columns in the query exist in the data source, and the columns in the result match the `id` and `type` of all `field` elements of the derived table given in the `fieldList`. SQL queries for a derived table are with respect to the JDBC driver for the data source. The syntax for a valid SQL query does *not* include a closing semi-colon.

You can also use JasperReports Server attributes in queries. See **6.10, "Using Server Attributes in Design Files," on page 100** for more information.

> If you are working with a virtual data source, SQL queries are validated against Teiid SQL, which provides DML SQL-92 support with select SQL-99 features. For more information, see the Teiid Reference Guide under the Documentation link on the Jaspersoft Support Portal.

In many cases, the easiest way to create a derived table in the XML design file is to create the initial derived table in the Domain Designer and then export the file. When you create a derived table in the Domain Designer, it runs the query and generates columns based on the result set. These columns are present in the export file.

### 6.5.4.1 Example

The following sample query in PostgreSQL selects some columns from the result of a join and orders the results. One of the columns includes a field calculated in the SQL.

Only fields selected in the query – in this case, `exp_date`, `store_id`, `amount`, `currency`, `conv`, and `as_dollars` – can be exposed as columns of the derived table. Fields not selected in the query cannot be referenced in the Domain design.

```
<query>
 select e.exp_date, e.store_id, e.amount, c.currency, c.conversion_ratio conv,
  amount * c.conversion_ratio as_dollars
 from expense_fact e join currency c
  on (e.currency_id = c.currency_id and date(e.exp_date) = c.date)
 order by e.exp_date
</query>
```

> The preceding example is not valid for SQL Server. SQL queries for a derived table are resolved using the JDBC driver for the data source and the derived table becomes a subquery in the generated SQL. SQL Server requires a TOP or FOR XML clause in any subquery that uses ORDER BY.

## 6.5.5  Using Derived Tables

A derived table provides an alternate way to create joins and calculated fields. Here are some things to keep in mind when deciding how to implement the Domain:

*   Calculated fields created within derived tables may use any function call recognized by the RDBMS. Calculated fields created in the Domain using the `dataSetExpression` attribute of the `field` element are limited to the functions available in the DomEL language. See **"Domain Expression Language (DomEL)" on page 105** for more information.
*   The Domain mechanism applies filters, aggregation, and joins to derived tables by wrapping the SQL in a nested query, which may be less efficient on some databases than the equivalent query generated for a non-derived table.

## 6.6  Representing Joins in XML

A join is represented in the design file as a `jdbcTable` element. A join tree is a child of the `resources` element.

> The `jdbcTable` element is also used to represent tables from the data source. See **6.4, "Representing Tables in XML," on page 73**.

## 6.6.1  Element Hierarchy for a Join Tree

The following hierarchy is used to represent a join tree.

```
<jdbcTable>
       <fieldList> (1)
              <field> (1...n)
       <filterString> (0...n)
       <joinInfo> (1)
       <joinList> (1)
              <join> (1...n)
       <joinOptions> (0...1)
       <tableRefList> (1)
              <tableRef> (1...n)
```

## 6.6.2    jdbcTable

The `jdbcTable` element is also used to represents a join tree, that is, the results of one or more joins between tables. There is one `jdbcTable` element for each join tree. `jdbcTable` is a child of `resources`.

> `jdbcTable` can also be used to represent a table. See **6.4, "Representing Tables in XML," on page 73** for more information.

### XML Attributes

| Attribute | Type | Description |
|---|---|---|
| id | String | (Required) Unique identifier for the join tree in the Domain design. In the Domain Designer, each join tree is automatically given the ID `JoinTree_n`, where `n` is a sequential number. In the design file, you can give the join any name, as long as it is unique among all tables and derived tables.<br><br>The `id` XML attribute can contain alphanumeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. |
| datasourceId | String | (Required) Alias that identifies the data source for the Domain. When creating a design file, this alias may be any name you choose, but it must be identical for all tables and derived tables. When uploading the file, the `datasourceId` automatically becomes the alias associated with the data source defined for the Domain. |

| Attribute | Type | Description |
|---|---|---|
| schemaAlias | | Name of the database schema for the first table in the join tree. Required if the data source uses schemas.<br>• For schemaless data sources, such as MySQL, schemaAlias is not required.<br>• For schema-based data sources, such as PostgreSQL and Oracle, the name is the literal name of the schema in the data source, for example, public.<br>• For schemaless data sources within a virtual data source, the schema name is of the form dataSourcePrefix, where dataSourcePrefix is the data source prefix defined when the virtual data source was created. For example, FoodmartDataSource.<br>• For schema-based data sources within a virtual data source, the name is in the form dataSourcePrefix_schema, where dataSourcePrefix is the data source prefix defined when the virtual data source was created. For example, FoodmartDataSource_public. |
| datasourceTableName | String | (Required) Literal name of the first table in the join tree. |

If you use the jdbcQuery element to define derived tables, you may have additional join trees in your Domain.

The Domain Designer automatically exposes all columns of all tables in a join, but in the design file you need to specify only those columns you want to reference elsewhere in the Domain.

### Child Elements

| Element Name | Description |
|---|---|
| **<fieldList>** | (Required) A container for the field elements in the join tree. A jdbcTable element can contain only one fieldList element. |
| **<filterString>** | Expression that evaluates to true or false when applied to each row of values in the data source. For a join tree, the expression refers to columns using the table_ID.field_name form of the column ID. See **6.7, "Representing Pre-filters in XML," on page 88** for more information. |
| **<joinInfo>** | (Required) Gives the table ID and alias for the table specified by the schemaAlias and datasourceTableName attributes. The table ID and alias are used as the first table in the join definition. This element and its two attributes are required even if they are identical. |

| Element Name | Description |
|---|---|
| `<joinList>` | (Required) Container for the `join` elements. A `jdbcTable` element can contain only one `joinList` element. The left join in the first `join` in the `joinList` must be the table specified by the `schemaAlias` and `datasourceTableName` attributes of `jdbcTable`. |
| `<joinOptions>` | Specifies options for the join tree that influence how the joins that are pushed down to SQL in an Ad Hoc view. |
| `<tableRefList>` | (Required) Container for the list of tables and aliases used in the join. Must include a `tableRef` tag for every table used. A `jdbcTable` element can contain only one `tableRefList` element. |

### 6.6.3    fieldList

`fieldList` is a container for the `field` elements in the join tree. `fieldList` is a child of the `jdbcTable` element.

#### Child Elements

| Element Name | Description |
|---|---|
| `<field>` | (Required) Represents a column in the join tree. |

### 6.6.4    field

`field` represents a column in the join tree. When you export Domain from the Domain Designer, the design file includes every column in every table of the join. When you create your own design file, only the columns you want to reference are required. This includes fields that may not appear directly in the Domain, such as fields used in joins. `field` is a child of the `fieldList` element.

> 📝    `field` elements are also used to represent calculated fields. Calculated fields still appear as children of `fieldList`, but their usage and attributes are different. See **6.8, "Representing Calculated Fields in XML," on page 89** for more information.

#### XML Attributes

| Attribute | Type | Description |
|---|---|---|
| `id` | String | (Required) Identifier for the field, composed of the ID of the table in the design and the literal name of the column in the data source. The syntax is `table_ID.field_name`. The `id` XML attribute can contain alphanumeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. |

| Attribute | Type | Description |
|---|---|---|
| type | String | (Required) The Java type of the column, identical to the type in its table definition. |

## 6.6.5  joinInfo

joinInfo gives the table ID and alias for the table specified by the schemaAlias and datasourceTableName attributes of the **jdbcTable** element. This table is used as the first table in the join definition. The table named in joinInfo does not have a tableRef element.

Both attributes of joinInfo are required even if they are identical to the values in jdbcTable.

### XML Attributes

| Attribute | Type | Description |
|---|---|---|
| alias | String | (Required) Alias for the table identified in referenceId; used in the expression attribute of the join element. By default, the alias is the same as the referenceID. If you use a distinct alias, you must be careful to use the alias throughout the join element that defines the join. |
| datasourceId | String | (Required) Alias for the data source for the Domain. When creating a design file, this alias may be any name you choose, but it must be identical for all tables and derived tables. When uploading the file, the datasourceId automatically becomes the alias associated with the data source defined for the Domain. |

## 6.6.6  joinList

<joinList> is a container for join elements. There is a join element for each join in the join tree. The left attribute of the first join element in joinList must be the table specified by the schemaAlias and datasourceTableName attributes of jdbcTable.

### Child Elements

| Element Name | Description |
|---|---|
| **join** | (Required) Representation of a join statement. |

## 6.6.7  join

The join element represents a single SQL join statement. The left and right tables, join type, join expression, and optional weight are specified as attributes. Every join in the join tree must have a separate join element.

```
left="left_table_ID" right=right_table_ID" type="join_type" expr="expression"
```

⚠ When you add or modify joins in the Domain design file, make sure that all tables in a join element are actually connected. If you include a table that is not actually joined to any other tables, the unjoined table will be included in any Ad Hoc view that uses that data island. In this case, the fields in the unjoined table will show up in the Ad Hoc view. You will receive errors in your Ad Hoc view when you add unconnected fields from a poorly-configured join.

### XML Attributes

| Attribute | Type | Description |
|---|---|---|
| `left` | String | (Required) Identifier for the first table in the join definition. This table must have been declared previously in the `jdbcTable` element. The declaration can appear either in the `<joinInfo>` element or in another `join` element that precedes this `join` element in the `joinList`. |
| `right` | String | (Required) Identifier for the second table in the join definition. |
| `join_type` | String | (Optional, defaults to `inner`). One of `inner`, `rightOuter`, `leftOuter`, or `fullOuter`.<br><br>📝 `fullOuter` is not supported in MySQL. |
| `expr` | String | Expression that compares the columns on which the `join` is made. See **Options for the expr Attribute** for more information. |
| `weight` | Integer > 0 | Integer that specifies a weight to use when calculating the minimum join path. You can assign a weight to one, several, or all of the `join` elements in a join set that has `suppressCircularJoins` enabled. Higher weights indicate costlier, less-desirable joins. When `suppressCircularJoins` is set to `true`, `weight` is ignored. |

### Options for the `expr` Attribute

The `expr` XML attribute defines the expression used to compare the columns on which the `join` is made.

Each column used in the expression must come from the two tables in the current `join` element. When two columns are compared directly – for example, `store.store_id == employee.store_id` – the first column must come from the left table and the second column must come from the right table.

The following expressions are supported:

- Boolean operators – AND, OR, and NOT. See the other expressions for examples of how these are used.
- Comparison operators – Supports equal to (==), less than (&lt;), less than or equal to (&lt;=), greater than (&gt;), greater than or equal to (&gt;=), and not equal to (!=). Operators other than == must be used in conjunction with ==. For example:

```
expr="(store.store_id == employee.store_id) AND
(store.coffee_bar != store.salad_bar)"
expr="(employee.employee_id == salary.employee_id) AND (salary.overtime_paid
&gt; 2)
AND (salary.overtime_paid &lt;= 2.1)"
```

> You have to use the character entities for less than (&lt;) and greater than (&gt;)

- IN operator – Must be used in conjunction with ==. Supports the following:
    - a set of strings or values, separated by commas. Strings are enclosed in single quotes. For example:

        `expr="(store.region_id == region.region_id) AND (store.store_city IN ('San Francisco','Portland', 'Seattle'))"`

        `expr="(store.store_id == employee.store_id) AND (NOT store.coffee_bar IN ('true'))"`

    - a range of values, separated by a colon. For example:

        `expr="(employee.employee_id == salary.employee_id) AND (NOT employee.salary IN (7000 : 8000))"`

- joins between a table foreign key and a constant value. Must be used in conjunction with ==. For example:

    `expr="(employee.employee_id == salary.employee_id) AND (employee.salary == 5000)"`

Some join expressions, such as NOT, can not be edited in the Domain Designer. However they can be displayed. When you import the Domain into the Domain Designer, these joins are shown in read-only format. See **4.4.7, "Read-Only Joins," on page 50** for more information.

> If you are using a field of type Date with an Oracle database, use the JasperReports Server `Date()` function in your join expression. For example:
>
> `<join left="FOODMART_STORE" right="FOODMART_EMPLOYEE" type="inner" expr="(FOODMART_STORE.STORE_ID == FOODMART_EMPLOYEE.STORE_ID) AND (FOODMART_EMPLOYEE.BIRTH_DATE &gt; `**`Date('1914-02-02')`**`)"/>`
>
> This applies to Date literals only; it is not necessary for fields with date formats but other types, for example, Timestamp.

## 6.6.8  joinOptions

The `joinOptions` element allows you to influence the joins that are pushed down to SQL in an Ad Hoc view. A `jdbcTable` element can contain only one `joinOptions` element, which sets the options for the entire join tree. `joinOptions` lets you avoid circular joins in cases where there are N tables with N or more joins. See **4.4.8.2, "Join Tree Options," on page 51** for more information.

## XML Attributes

| Attribute | Type | Description |
|---|---|---|
| `suppressCircularJoins` | Boolean | (Default = `false`) When this attribute is set to `true` for a join tree, an Ad Hoc view created from the join tree uses a minimum join. When `suppressCircularJoins` is set to `true`, you can optionally assign weights to the joins in the join set.<br><br>When `suppressCircularJoins` is set to `false`, all joins in the join tree are used in Ad Hoc views.<br><br>For most situations, the best practice is to set this option to `true` to avoid circular joins. Set it to `false` only if you need backwards compatibility with schema_1_0.xsd of the Domain syntax. |
| `includeAllDataIslandJoins` | Boolean | (Default= `false`) Available for backwards compatibility. When this attribute is set to `true` for a join set/data island, an Ad Hoc view created from the data island includes all joins specified for that data island in the Domain. This attribute can only be set to `true` when `suppressCircularJoins` is `false`.<br>For most situations, the best practice is to set this option to `false` (default). |

### 6.6.9 tableRefList

The `<tableRefList>` element is a container for the list of tables and aliases used in the join tree. It must include a `tableRef` element for every table in the join tree.

### Child Elements

| Element Name | Description |
|---|---|
| **`<tableRef>`** | (Required) Represents a table used in the join tree. |

### 6.6.10 tableRef

The `<tableRef>` element represents a table used in the join tree. There must be a `tableRef` element for every table in the join tree.

XML Attributes

| Attribute | Type | Description |
|---|---|---|
| tableId | String | The ID of a table within the design. |
| tableAlias | String | Alternative name for the table used within the join expression. |
| alwaysIncludeTable | Boolean | (Default = `false`) Setting this to `true` forces this table to be included in all Ad Hoc views created from this data island. For most situations, the best practice is to set this option to `false` (default). |

## 6.6.11  Join Tree Example

The following example shows a join tree with three joins between three tables. This example shows how you might use the following:

- To avoid circular joins, `suppressCircularJoins` is set to true.
- The join between `region` and `customer` has a join `weight` of 2. This makes it a less desirable join than the other joins in the tree.
- `alwaysIncludeTable` is set to `true` for the region table.

```
<schema xmlns="http://www.jaspersoft.com/2007/SL/XMLSchema" version="1.3">

<resources>

   <jdbcTable id="JoinTree_1" datasourceId="FoodmartDataSourceJNDI" schemaAlias="public" data-
sourceTableName="customer">
     <fieldList>
       <field id="customer.account_num" fieldDBName="account_num" type="java.lang.Long" />
    ...
     <fieldList>
         <joinInfo alias="store" referenceId="store" />
         <joinOptions suppressCircularJoins="true"/>

         <tableRefList>
           <tableRef tableId="store" tableAlias="store"/>
           <tableRef tableId="customer" tableAlias="customer"/>
           <tableRef alwaysIncludeTable="true" tableId="region" tableAlias="region"/>
         </tableRefList>

      <joinList>
        <join left="store" right="customer" type="inner"
                   expr="store.region_id == customer.customer_region_id"/>
        <join left="store" right="region" type="inner"
                   expr="store.region_id == region.region_id"/>
       <join weight="2" left="region" right="customer" type="inner"
                   expr="region.region_id == customer.customer_region_id"/>
        </joinList>

   </jdbcTable>
</resources>

</schema>
```

## 6.6.12 Working With Joins

Keep the following in mind when working with join syntax:

- When you add or modify joins in the Domain design file, make sure that all tables in a join element are actually connected. If you include a table that is not actually joined to any other tables, the unjoined table will be included in any Ad Hoc view that uses that data island. In this case, the fields in the unjoined table will show up in the Ad Hoc view. You will receive errors in your Ad Hoc view when you add unconnected fields from a poorly-configured join.
- When you use a `join` tag, the `left` table must have been defined previously in the `jdbcTable` element. You can define the table either in the `joinInfo` tag or as the `right` table in a previous join in the same `joinList`. Follow these guidelines for your joins:
  - When you have a `join` that depends on the `right` table from another join in the `joinList`, make sure that the dependent `join` appears after the join it depends on.
  - When you have a `join` that refers to a table that has not yet been defined, make sure that the new table is referenced as the `right` table in the join.
- In some cases, you can start your Domain creation using the Domain Designer. To do this:
  a. Set up the rest of your Domain, such as derived tables and calculated fields, in the Domain Designer.
  b. Create a join that uses the same tables that you want to join in your Domain. This will set up the `jdbcTable` and `fieldList` tags for you.
  c. Export the schema from the Domain Designer.
  d. Manually edit the `tableRefList` and `joinInfo` sections of the exported design file to create the join you want.

## 6.6.13 Tips for Using Joins

- To avoid join cycles in queries, set `<joinOptions suppressCircularJoins=true/>`. This will find the lowest weight join set.
- To configure your Domain to use the fewest number of joins required to reach a given set of chosen fields, use `<joinOptions suppressCircularJoins=true/>` AND set all `<join weight="W">` values to be the same value. In this case, the lowest weight join set will be a set that has the minimum possible number of joins.
- You do not have to set the join weight explicitly. If the weight is not set, it defaults to 1.
- To give preferential treatment to joins that involve indexed columns for better join performance, assign lower join weights to the preferred joins and higher join weights to less desirable joins.
- To make sure that one or more specific table(s) always get included in your results, set `alwaysIncludeTable=true`. For example, to make sure tableA is always included, use
  `<tableRef tableId=tableA tableAlias=tableA alwaysIncludeTable=true/>`

## 6.7 Representing Pre-filters in XML

Pre-filters are defined as optional `filterString` elements inside of `jdbcTable` and `jdbcQuery` elements. They impose a condition on any results returned for that table, query, or join tree, thereby limiting the number of rows returned when accessing the data source. Whereas other settings mainly determine which columns are available for use in a report, a pre-filter determines which rows are available when running the report.

## 6.7.1    filterString

`filterString` contains an expression that evaluates to true or false when applied to each row of values in the data source. The parent of `filterString` is one of the following:

- a **`jdbcTable`** element for a table
- a **`jdbcTable`** element for a join
- a **`jdbcQuery`** element

The expression refers to columns using their `id` attribute. Thus, a filter on a table or derived table refers to the simple column name, but a filter on a join tree refers to the `table_ID.field_name`. The full syntax for the expression is documented in **"Domain Expression Language (DomEL)" on page 105**.

> Filters defined in the Domain Designer are limited to conditions on one column or comparisons of two columns, with more complex filters created by the conjunction (logical AND) of several conditions. Other filter expressions are not supported. You can upload a design file with more complex filters, but these filters are not editable in the Domain Designer.

### 6.7.1.1 Examples

The following is an example of `filterString` used in a `jdbcTable` element representing a table:

```
<jdbcTable datasourceId="SugarCRMDataSource" id="opportunities"
          schemaAlias="public" datasourceTableName="opportunities">
      <fieldList>
              ...
              <field id="opportunity_type" type="java.lang.String"/>
      </fieldList>
      <filterString>opportunity_type == 'Existing Business'</filterString>
</jdbcTable>
```

The following is an example of `filterString` used in a `jdbcQuery` element:

```
<jdbcQuery datasourceId="SugarCRMDataSource" id="p1cases">
      <fieldList>
              ...
              <field id="status" type="java.lang.String"/>
      </fieldList>
      <filterString>status != 'closed'</filterString>
      <query>...</query>

</jdbcQuery>
```

## 6.8    Representing Calculated Fields in XML

Calculated fields are defined as `field` elements with an additional XML attribute, `dataSetExpression`, that holds the calculation for the field.

## 6.8.1   field

The `field` element is a child of the `fieldList` element. The location of the `fieldList` element for calculated fields depends on the location of the fields used in the calculation:

- Calculated fields where all the columns in the calculation come from a single table appear twice in the design file:
  - Under the `jdbcTable` element for the join tree that contains the table.
  - Under the `jdbcTable` or `jdbcQuery` element for that table.
- Calculated fields that use columns from different tables appear only once, under the `jdbcTable` element for the join tree contains both tables. Remember, it is not possible to create calculated fields from columns in different join trees.
- Constant fields appear under `resources` as a `null` element.

  Constant fields are calculated fields that do not reference any column names. Constant fields are defined as `field` elements in a `fieldList` under the `null` element. Because constant fields are not dependent on any column values, they may be used in any join tree or data island. They can also be used in other calculated fields and pre-filters.

### XML Attributes

| Attribute | Type | Description |
|---|---|---|
| dataSetExpression | String | (Required) Expression that calculates a value. This value is either based on other columns or is a constant value. For a description of the syntax for the expression, including how to reference columns, see **"Domain Expression Language (DomEL)" on page 105**. For information on using attributes in calculated fields, see **6.10, "Using Server Attributes in Design Files," on page 100**. |

| Attribute | Type | Description |
| --- | --- | --- |
| id | String | (Required) Identifier for the calculated field. The format of the `id` depends on how the calculated field appears in the design file:<br><br>1. If all columns used in the expression are from the same table, and the table appears in a join, there are two `field` elements for the calculated field, with different ids:<br><ul><li>When the field appears under the `jdbcTable` or `jdbcQuery` element for the table, the `id` is a simple column name `field_name`.</li><li>When the field appears under the `jdbcTable` element for the join tree that uses the table, the `id` has the form `table_ID.field_name`.</li></ul>2. If the expression references columns from different tables, the field appears only in the join tree that contains those tables and the `id` has the form `jointree_ID.field_name`.<br>3. If the expression is a constant value, the field appears in the `null` element under `resources` and the `id` is `constant_fields_level.field_name`.<br><br>The `id` XML attribute can contain alphanumeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. |
| type | String | The Java type of the value calculated by the expression, for example `java.lang.String`. This type must be compatible with the result of the DomEL expression and among the JDBC-compatible types listed in **6.4.3.1, "Supported Types," on page 76**. |

### 6.8.1.1  Example

The following example shows the XML for a calculated expression that only references one table, the `accounts` table. Because it references only the columns of `accounts`, it appears in that table and in the join tree.

```
<jdbcTable datasourceId="SugarCRMDataSource" id="accounts"
             schemaAlias="public" datasourceTableName="accounts">
 <fieldList>
  ...
  <field dataSetExpression="concat( billing_address_city, ', ',
     billing_address_state )" id="city_and_state" type="java.lang.String"/>
  ...
 </fieldlist>
</jdbcTable>
...
<jdbcTable datasourceId="SugarCRMDataSource" id="JoinTree_1"
             schemaAlias="public" datasourceTableName="anything">
 <fieldList>
  ...
```

```
  <field dataSetExpression="concat( billing_address_city, ', ',
     accounts.billing_address_state )" id="accounts.city_and_state"
     type="java.lang.String"/>
 ...
 </fieldlist>
</jdbcTable>
```

## 6.8.2    null

The `null` element is a container for constant fields. `null` is a child of the **6.3, "resources," on page 72** element.

### Child Elements

| Element Name | Description |
|---|---|
| **\<fieldList\>** | (Required) A container for the `field` elements for constant fields. The `null` element can contain only one `fieldlist` element. |

## 6.8.3    fieldList

As a child of `null`, `fieldList` is a container for one or more `field` elements that represent constant fields.

### Child Elements

| Element Name | Description |
|---|---|
| \<field\> | (Required) A constant field. Constant fields may be used in any join tree or data island. They can also be used in other calculated fields and pre-filters. |

### 6.8.3.1  Example

The following example shows the XML for two constant calculated fields.

```
<resources>
  <null id="constant_fields_level" datasourceId="FoodmartDataSourceJNDI">
    <fieldList>
      <field id="CF_Constant" dataSetExpression="1 + 1 + 1" type="java.math.BigInteger" />
      <field id="CF_Constant_2" dataSetExpression="12" type="java.math.BigInteger" />
    </fieldList>
  </null>

...

</resources>
```

# 6.9    Representing Sets and Items in XML

The hierarchy of data islands, sets, and items are defined through the `dataIslands` and `itemGroups` elements and their children. These two elements, which are defined together, are direct children of the top-level `schema`

element.

### Presentation Hierarchy

The following hierarchy is used to represent data islands, sets, and items. This hierarchy is at the top level, directly under the schema element.

```
<schema>
        <dataIslands> (0...1)
                <itemGroup> (1...n)
        <itemGroups> (1...n)
                <itemGroup> (0...n)
                        <itemGroups> (0...n)
                                <itemGroup> (0...n)
                                        <itemGroups> (0...n)
                                <items> (0...n)
                                        <item> (1...n)

                                        ...
                        <items> (0...n)
                                <item> (1...n)
                <items> (0...n)
                        <item> (1...n)
        <items> (0...n)
                <item> (1...n)
```

The itemGroups and items elements are recursive elements that are equivalent to the sets and items on the Data Presentation tab of the Domain Designer. They create a hierarchy of sets, subsets and items and hold attributes that define all the properties available on sets and items. For a description of each possible property, see **"Properties" on page 36**.

## 6.9.1    dataIslands

The dataIslands element lists all the data islands on the Data Presentation tab. A schema can have at most one dataIslands element. If there are no data islands in the presentation, the dataIslands element is not required.

Each data island in the Domain must be represented by an itemGroup child of the dataIslands element. dataIslands is a child of schema.

### Child Elements

| Element Name | Description |
| --- | --- |
| **<itemGroup>** | (Required) As a child of dataIslands, each itemGroup represents a data island. |

## 6.9.2    itemGroup

As a child element of dataIslands, itemGroup represents a data island. Each itemGroup in the dataIslands element corresponds to an itemGroups element that describes the hierarchy of sets and items in the data island. There can be at most one data island for each join tree or unjoined table.

> itemGroup is also used to represent a set in the Domain presentation. See **6.9.4, "itemGroup," on page 95** for more information.

XML Attributes

| Attribute | Type | Description |
|---|---|---|
| id | String | (Required) Unique identifier for the data island in the Domain design file. The `id` XML attribute can contain alphanumeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. |
| | | The `id` attribute is used by other elements in the presentation representation to identify the data island, via the `referenceId`. In XML files exported from Domain Designer, the `id` attribute corresponds to the data island's **ID** property on the Data Presentation tab. |
| label | String | The data island's name, visible to users of the Domain. If the label is missing, the Ad Hoc Editor displays the `id`. Corresponds to **Label** on the Data Presentation tab. |
| description | String | A description of the set, visible to users as a tooltip on the set name in the Ad Hoc Editor. Corresponds to **Description** on the Data Presentation tab. |
| labelId | String | The internationalization key for the label in the Domain's locale bundles. Corresponds to **Label Key** on the Data Presentation tab. Can contain alphanumeric characters along with any of the following: .`~_ |
| descriptionId | String | The internationalization key for the description in the Domain's locale bundles. Corresponds to **Descriptions Key** on the Data Presentation tab. Can contain alphanumeric characters along with any of the following: .`~_ |
| resourceId | String | (Required) `resourceId` of the `jdbcTable` or `jdbcQuery` element for the join tree or unjoined table that corresponds to the data island. jdbcQuery is only used in the case when your data island comes from a derived table that isn't joined to anybody. |

## 6.9.3    itemGroups

`itemGroups` – A container for `itemGroup` elements and/or `items` elements. Each instance of `itemGroups` corresponds to a data island defined as an `itemGroup` in `dataIslands`.

All descendents of a specific `itemGroups` element must refer to the same join tree or unjoined table. This means that the `resourceId` attribute of all descendents of an `itemGroups` element must reference the `id` attribute of the same `jdbcTable` or `jdbcQuery`.

You can upload a Domain with an empty `itemGroups` element, but for the corresponding data island to be used, it must contain at least one `itemGroup` or `items` element.

A recursive relationship between `itemGroups` and `itemGroup` is used to show sets and subsets. The top-level element in a set hierarchy is always an `itemGroups` element. Every `itemGroups` element must contain at least one `items` or `itemGroup` element.

### Child Elements

| Element Name | Description |
| --- | --- |
| `<itemGroup>` | A set in the data presentation. |
| `<items>` | A container for `item` elements. |

## 6.9.4 itemGroup

As a child of the `itemGroups` element or `itemGroup` element, `itemGroup` represents a set or subset. `itemGroup` contains `itemGroups` elements and/or `items` elements. Set properties are represented as attributes.

A recursive relationship between `itemGroups` and `itemGroup` is used to show sets and subsets. The top-level element in a set hierarchy is always an `itemGroups` element. Every `itemGroups` element must contain at least one `items` or `itemGroup` element.

> `itemGroup` is also used to represent a data island in `dataIslands`. See **6.9.2, "itemGroup," on page 93** for more information.

### Child Elements

The hierarchy of `itemGroups` and `items` elements defines the hierarchy of sets and items in the data island.

| Element Name | Description |
| --- | --- |
| `<itemGroups>` | A container for `itemGroup` elements, which represent sets. |
| `<items>` | A container for `item` elements, which represent items. |

### XML Attributes

The attributes of `itemGroup` specify the properties of the set or data island. See **3.6.5, "Properties," on page 36** for more information.

| Attribute | Type | Description |
| --- | --- | --- |
| `id` | String | (Required) The unique identifier of the set among all set and item IDs. The `id` XML attribute can contain alphanumeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. |
| | | The `id` attribute is used by other elements in the presentation representation to identify the set. In XML files exported from Domain Designer, the `id` attribute corresponds to the set's **ID** property on the data presentation tab. |

| Attribute | Type | Description |
|---|---|---|
| label | String | The set's name, visible to users of the Domain. If the label is missing, the Ad Hoc Editor displays the id. Corresponds to **Label** on the Data Presentation tab. |
| description | String | A description of the set, visible to users as a tooltip on the set name in the Ad Hoc Editor. Corresponds to **Description** on the Data Presentation tab. |
| labelId | String | The internationalization key for the label in the Domain's locale bundles. Corresponds to **Label Key** on the Data Presentation tab. If blank, the default key is used. Can contain alphanumeric characters along with any of the following: .`~_ |
| descriptionId | String | The internationalization key for the description in the Domain's locale bundles. Corresponds to **Descriptions Key** on the Data Presentation tab. If blank, the default key is used. Can contain alphanumeric characters along with any of the following: .`~_ |
| resourceId | String | (Required) id of the join tree or unjoined table that contains this set, as specified in dataIslands. The resourceId attribute must be the same for all item and itemGroup elements that descend from the same itemGroups element. |

When an internationalization key is defined for the label or description, the label or description is replaced with the value given by the key in the local bundle corresponding to the user's locale. See **Chapter 9, "Localizing Domains," on page 129**.

## 6.9.5    items

The items element is a container for item elements, which represent items on the Data Presentation tab. items can be a child of itemGroup or schema.

- As a child of schema, items is a container for items in a single data island that do not belong to any a set. All item elements in an items element must belong to the same data island. There is one items element in schema for each data island that has items that are not in a set.
- As a child of itemGroup, items is a container for the items in the set represented by itemGroup.

### Child Elements

| Element Name | Description |
|---|---|
| **<item>** | An item in the data presentation. |

## 6.9.6    item

The item element represents an item in the data presentation of the Domain. An item is the representation of a database field or a calculated field along with the display name and formatting properties defined in the Domain.

## XML Attributes

The attributes of `item` specify the properties of the item. See **3.6.5, "Properties," on page 36** for more information.

| Attribute | Type | Description |
|---|---|---|
| `id` | String | (Required) The unique identifier of the item among all set and item IDs. The `id` XML attribute can contain alpha-numeric characters along with any combination of the following: @#$^`_~? It cannot start with a digit. |
|  |  | By default, the Domain Designer uses the name of the `field` as the id. If there are multiple instances of the field in the data island, then the Domain Designer appends a number to the name, such as `field2`. |
| `label` | String | The item's name, visible to users. If the label is missing, the Ad Hoc Editor displays the `id`. |
| `description` | String | An optional description of the item, visible as a tooltip on the item name in the Ad Hoc Editor. |
| `labelId` | String | The internationalization key for the label in the Domain's locale bundles. Can contain alphanumeric characters along with any of the following: .`~_ |
| `descriptionId` | String | The internationalization key for the description in the Domain's locale bundles. Can contain alphanumeric characters along with any of the following: .`~_ |
| `resourceId` | String | (Required) A reference to the column on which the item is based. This defines the connection between what the user sees and the corresponding data in the data source. The precise format depends on the location of the column in the data structure:<br>• When the item refers to a column in an unjoined table, `resourceId` has the form `table_id.field_ID`.<br>• When the item refers to a column in a join tree, `resourceId` has the form `jointree_id.table_ID.field_name`.<br><br>In the Domain Designer, `table_id` and `jointree_id` can be set using the **Rename...** option from the context menu on the Joins tab.<br><br>The `table_id` or `jointree_id` component of the `resourceId` attribute must be the same for all `item` and `itemGroup` elements that descend from the same `itemGroups` element. |

| Attribute | Type | Description |
|---|---|---|
| dimensionOrMeasure | Field or Measure | Corresponds to the **Field or measure** setting in the user interface. Its possible values are Dimension (equivalent to field) or Measure.<br><br>This attribute is optional and necessary only when overriding the default behavior, for example to make a numeric item explicitly not a measure. By default, all numeric fields are treated as measures in the Ad Hoc Editor. |
| defaultMask | See table | A representation of the default data format to use when this item is included in a report. The possible values depend on the type attribute of the column referenced by the resourceId. See **Table 6-1 on page 98** for the data formats available for each type. |
| defaultAgg | See table. | The name of the default summary function (also called aggregation) to use when this item is included in a report. The possible values for the defaultAgg depend on the type attribute of the column referenced by the resourceId. See **Table 6-2 on page 99** for the possible summary functions based on the column type. The **Appearance** column shows the equivalent setting in the Data Presentation tab. |

Labels and descriptions may contain any characters, but the ID property value of both itemGroup and item elements must be alphanumeric.

**Table 6-1  Data Formats for Aggregation by Type**

| Content Type | Attribute Value | Appearance |
|---|---|---|
| Integer | #,##0<br>0<br>$#,##0;($#,##0)<br>#,##0;(#,##0) | -1,234<br>-1234<br>($1,234)<br>(1234) |
| Double | #,##0.00<br>0<br>$#,##0.00;($#,##0.00)<br>$#,##0;($#,##0) | -1,234.56<br>-1234<br>($1,234.56)<br>($1,234) |
| Date | short<br>hide medium<br>hide long<br>medium | 3/31/09<br>Mar 31, 2009<br>March 31, 2009<br>23:59:59 |
| All others | *Not allowed* | |

**Table 6-2  Default Summary Functions By Type**

| Attribute Value | Appearance |
|---|---|
| Max | Maximum |
| Min | Minimum |
| Average | Average |
| Sum | Sum |
| CountDistinct | Distinct Count |
| CountAll | Count All |

## 6.9.7    Example

For example, suppose you have a join tree, JoinTree_1, with the following hierarchy:

- A set labeled Accounts
- A subset of Accounts, with the label Account Address
- A single element, Account Creator, that is in the join tree, but is not in any set.
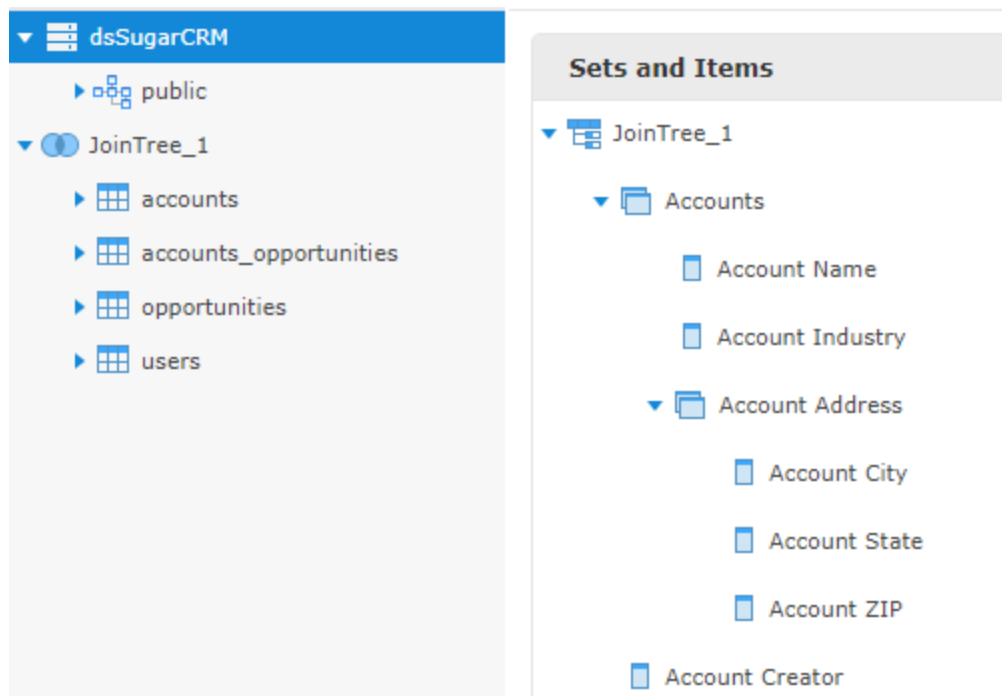


**Figure 6-1  Example of a Join Tree Hierarchy**

The XML for this join tree might look like this:

```
<dataIslands>
  <itemGroup id="JoinTree_1" resourceId="JoinTree_1" />
```

```
</dataIslands>
<itemGroups>
  <itemGroup id="AccountsSet" label="Accounts" resourceId="JoinTree_1">
    <itemGroups>
      <itemGroup id="AddressSubset" label="Account Address" resourceId="JoinTree_1">
        <items>
          <item id="billing_address_city" label="Account City" resourceId="JoinTree_1.ac-
counts.billing_address_city" />
          <item id="billing_address_state" label="Account State" resourceId="JoinTree_1.ac-
counts.billing_address_state" />
          <item id="billing_address_postalcode" label="Account ZIP" resourceId="JoinTree_1.ac-
counts.billing_address_postalcode" />
        </items>
      </itemGroup>
    </itemGroups>
    <items>
      <item id="name" label="Account Name" resourceId="JoinTree_1.accounts.name" />
      <item id="industry" label="Account Industry" resourceId="JoinTree_1.accounts.industry" />
    </items>
  </itemGroup>
</itemGroups>
<items>
    <item id="created_by" label="Account Creator" resourceId="JoinTree_1.accounts.created_by" />
</items>
```

## 6.10   Using Server Attributes in Design Files

A JasperReports Server attribute is a name-value pair associated with a user, organization, or server. Attributes let you create Domains that are customized for each user based on the values of their attributes. For example, if the database schema is defined using an organization-level attribute, the Domain will access data specific to each user's organization. This lets you create one Domain for many organizations, provided they all use the same tables in the schema.

To define or modify attributes on the server, see the section on attributes in the *JasperReports Server Administrator Guide*.

You can use attributes in the Domain design file. When you use an attribute, the server derives the value at run time, based on the user, organization, or server. Attributes are supported in DomEl expressions in the following locations:

•   Calculated field calculations
•   Join expressions
•   Pre-filter expressions

They can also be used in other locations in the Domain design, including:

•   Schema entry keys
•   Derived table queries

If you want to use a JasperReports Server attribute in a data source, you must specify the attribute when you create the data source, not in the Domain design file. See the *JasperReports Server Administrator Guide* for information about using attributes in data sources.

## 6.10.1  Attribute Syntax

When referring to an attribute, 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. You can specify these attribute categories:
  - User – In the attributes defined on the logged-in user.
  - Tenant – In the attributes defined on the organization of 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 filter searches attributes hierarchically and uses the value for the first attribute it finds with the given name. This search starts with the logged-in user, then proceeds to the user's organization and parent organization, and finally to the server level.

For join expressions, calculated fields, and pre-filters, the DomEL syntax is used. See **Chapter 7, "Domain Expression Language (DomEL)," on page 105** for more information.

- `attribute('attributeName', 'server')` for a server-level attribute.
- `attribute('attributeName', 'tenant')` for an organization-level attribute.
- `attribute('attributeName', 'user')` for a user-level attribute.

If no level is specified, the server will search for the attribute hierarchically, starting at the `'user'` level:

- `attribute('attributeName')`

For schema names and derived tables, the attribute must be escaped with curly brackets ({}):

- `{attribute('attributeName'), 'server'}` for a server-level attribute.
- `{attribute('attributeName', 'tenant')}` for an organization-level attribute.
- `{attribute('attributeName', 'user')}` for a user-level attribute.
- `{attribute('attributeName')}` to search for an attribute hierarchically.

For more information on attributes, see the section "Managing Attributes" in the *JasperReports Server Administrator Guide*. For information about using JasperReports Server attributes in Domain security files, see **8.4.3, "User Attributes," on page 115** and **8.6, "Creating a Domain Security File," on page 118**.

## 6.10.2  Examples

To use a JasperReports Server attribute as the name of a schema in the data source, use the attribute syntax in the `string` element for the `entry` for that schema. You can use a constant or an attribute for the XML `key` attribute of `entry`.

```
<jdbcDataSource id="FoodmartDataSourceJNDI">
  <schemaMap>
    <entry key="defaultSchema">
      <string>public</string>
    </entry>
    <entry key="_attribute_schemaAttribute_">
      <string>{attribute('schemaAttribute')}</string>
    </entry>
  </schemaMap>
</jdbcDataSource>
```

For pre-filters, JasperReports Server attributes can appear in the expression for the `filterString` element in the `jdbcTble` or `jdbcQuery` element:

```
<jdbcTable id="region" datasourceId="FoodmartDataSourceJNDI" schemaAlias="public" data-
sourceTableName="region">
  <fieldList>
    <field id="region_id" type="java.lang.Integer" />
    <field id="sales_city" type="java.lang.String" />
    <field id="sales_country" type="java.lang.String" />
  </fieldList>
  <filterString>sales_country == attribute('attributeCountry')</filterString>
</jdbcTable>
```

For calculated fields, JasperReports Server attributes can appear in `dataSetExpression` in the `field` element:

```
<field id="Sample_Calculated_Field" dataSetExpression="salary_paid &gt; attribute('numer-
icAttribute')" type="java.lang.Boolean" />
```

For derived tables, JasperReports Server attributes can appear in the `query` element in `jdbcQuery`. For example, if you have defined an attribute with the name of the table you want called `tableAttribute`, a derived table that uses that attribute might look like this:

```
<jdbcQuery id="inventory_fact_1998_2" datasourceId="dsFoodMart">
      <fieldList>
        <field id="warehouse_cost" type="java.math.BigDecimal" />
        <field id="warehouse_profit" type="java.math.BigDecimal" />
        <field id="warehouse_sales" type="java.math.BigDecimal" />
      </fieldList>
      <query>select

  {attribute('tableAttribute')}.warehouse_sales as warehouse_sales,
  {attribute('tableAttribute')}.warehouse_cost as warehouse_cost,

  ( {attribute('tableAttribute')}.warehouse_sales -  {attribute('tableAttribute')}.warehouse_
cost) as warehouse_profit

from
  {attribute('tableAttribute')}</query>
    </jdbcQuery>
```

You can use attributes in the secondary joins in a complex join `expression`. However, you can't use attributes in the first join in the expression:

```
<joinInfo alias="account" referenceId="account" />
<joinList>
    <join expr="account.account_id == expense_fact.account_id and
        account.account_description == attribute('tableAttribute')"
        left="account" right="expense_fact" type="inner" weight="1" />
</joinList>
<joinOptions />
```

### 6.10.3 Attribute Casting Functions

A JasperReports Server attribute value is always a string. You can use the following functions to cast it to other types in DomEl expressions, for example if you want to compare it to a field value.

| Function | Syntax | Example |
|----------|--------|---------|
| Integer | `Integer('string')` | Integer('345') = 345; |
| Decimal | `Decimal('string')` | Decimal('24.5') = 24.5; |
| Boolean | `Boolean('string')` | Boolean('true') = true; |
| Date | `Date('string')` | Date('2008-02-08') = d'2008-02-08'; |
| Timestamp | `Timestamp('string')` | Timestamp('2008-02-08 11:30:00') = ts'2008-02-08 11:30:00'; |
| Time | `Time('string')` | Time('11:30:00') = t'11:30:00'; |

## 6.10.4  Guidelines for Using Attributes

Note the following when using attributes:

- The JasperReports Server attribute value must match the expected data type. For example, if you are creating a calculated field of type Integer, the attribute value must be an Integer.
- Attribute collections are not supported in the main Domain design. The JasperReports Server attribute must be a single value. Attribute collections can be used in Domain security, however.
- If you are using JasperReports Server attributes for the name of a schema, make sure that any dependent tables and columns in the Domain will be available for every possible value of the attribute. See **4.6.2, "Changing the Data Source," on page 56** for more information.

> If you use an attribute in your data source definition, you must ensure that referenced schemas, tables, and columns will be available. Data source attributes are defined outside the Domain Designer. See the section on defining attributes in the *JasperReports Server Administrator Guide* for more information.

- If a specified attribute does not exist, any item that uses it will fail. For example, if you specify the attribute `Country` for a schema name, and a `Country` attribute does not exist anywhere on the server, you will be unable to load the schema. This is similar to referring to column that does not exist.
- If an attribute exists but is not defined (empty) for a particular user, the behavior depends on where the attribute is used. An empty attribute can occur, for example, if you have set up a `Country` attribute, but the current user has no value defined for `Country`.
  - For pre-filters, an empty attribute is interpreted based on the field's data type, as described below.
    - For a field of type String, an empty attribute is interpreted as an empty string: `""`
    - For a Boolean field, and empty attribute is interpreted as `FALSE`.
    - For numeric and date fields, an error will occur.
  - For derived table queries and custom joins, the attribute will fail in most cases. However, in some cases, JasperReports Server will consider the resulting expression to be valid, with potentially unexpected results, so be cautious when using attributes in this part of your design.
  - For schema names, the default schema is used when the attribute is empty. The default schema must be defined in the XML Domain design file. See **6.2.5.1, "Default Schema," on page 71** for more information.

## 6.11    UTF-8 Support In Domains

JasperReports Server supports international characters in the database schema, for example in the table and column names. The server supports UTF-8 (8-bit Unicode Transformation Format) character encoding as described in the Localization chapter in the *JasperReports Server Administrator Guide*.

If your database is properly configured and the appropriate fonts are available to your browser, the Domain Designer displays the international characters on the various tabs where table and column names appear. This means that your data architects can work in their native language within the Domain Designer.

> As with all values from your reporting database, table and column names are not localizable. They can only be displayed in the Domain Designer as they appear in the database. You can, however, have localized labels for these table and column names, so that they appear in the Ad Hoc designer according to the user's locale.

IDs have the following character restrictions:

*   All UTF-8 characters are allowed in IDs except for the following symbols, which will cause errors:

    ```
    ().,"=!+-></:[]*|?$
    ```

    This limitation applies to table IDs in the database as well as user-created IDs in design files and in the Domain Designer.
*   The straight single quote (`'`, Unicode 0027) can appear in database table IDs but not in user-created IDs.
*   In general, the best practice is to use alphabet characters, not punctuation or arithmetic symbols, when creating Domains.

# CHAPTER 7    DOMAIN EXPRESSION LANGUAGE (DOMEL)

A DomEL expression is a shorthand way of writing a complex query. Many components of a Domain need to compute values based on some expression involving constants, field values, and environment variables. The Domain Expression Language (DomEL) was created to fill this need. Currently, the following features in XML design files are expressed in DomEL:

- The `IN` clause for custom joins
- Calculated fields
- Filter expressions in Domains and Domain topics (equivalent to `where` clauses)
- Server attribute values (see **6.10, "Using Server Attributes in Design Files," on page 100**)
- Row-level security (see **Chapter 8, "Securing Data in a Domain," on page 111**)

When processing a report based on a Domain, the server interprets DomEL expressions to generate parts of the SQL expression that perform the desired query. Depending on the data policy, either the augmented SQL is passed to the data source, or the server performs a simpler query and applies the DomEL expressions to the full dataset in memory.

This chapter contains the following sections:

- **Datatypes**
- **Field References**
- **Operators and Functions**
- **SQL Functions**
- **The groovy() Function**
- **Complex Expressions**
- **Return Types**

## 7.1    Datatypes

The following simple datatypes may be declared as constants, used in expressions, and returned as values:

| Simple Type | Description | Example of Constant |
|---|---|---|
| boolean | Expressions such as comparison operators return boolean values, but `true` and `false` constants are undefined and cannot be used. | none |

| Simple Type | Description | Example of Constant |
|---|---|---|
| integer | Whole numbers. | `123` or `-123` |
| decimal | Floating point numbers. Decimal separator must be a period ( . ); other separators such as comma ( , ) are not supported. | `123.45` or `-123.45` |
| string | Character string entered with single quotes ( ' ); double quotes ( " ) are not supported. | 'hello world' |
| date | ANSI standard date. | `d'2009-03-31'` or `Date('2009-03-31')` |
| timestamp | ANSI standard date and time. | `ts'2009-03-31 23:59:59'` or TimeStamp('2009-03-31 23:59:59') |

The composite datatypes, shown in the following table, may be declared as constants and used with the `in` *set* or `in` *range* operator. The values in these composite types are not necessarily constant, they could be determined by field values.

| Composite Type | Description | Example |
|---|---|---|
| set | Contains any number of any simple type above. | `(1, 2, 3)('apples','oranges')` |
| range | Inclusive range applicable to numbers and dates, including fields that are number or date types. | `(0:12)` or `(0.0:12.34)(d'2009-01-01':d'2009-12-31')(limit_min:limit_max)` |

## 7.2    Field References

DomEL expressions are stored in the Domain design and interpreted when the server prepares to run a query to retrieve data from the data source. Therefore, all references to field values in an expression are based on the `id` attributes defined in the Domain design. Field references have the following format, depending on where the expression appears:

| Appears In | Field Reference | Explanation |
|---|---|---|
| Derived table | `table_id.field_name` | The SQL query that defines a derived table can refer to any previously defined table or derived table in the Domain. Therefore, you must include the table ID. |
| Join expression | `table_id.field_name` | Within a join expression, tables are given alias names that must be used. |

| Appears In | Field Reference | Explanation |
|---|---|---|
| Calculated field on a table or derived table | `field_name` | Calculated fields can only appear on a table if they refer exclusively to fields of the table, in which case no table ID is needed. However, the table ID is not forbidden, and the Domain Designer sometimes includes it. |
| Calculated field on a join tree | `table_id.field_name` | Calculated fields declared in join trees refer to fields prefixed with their table ID. |
| Filter on a table or derived table | `field_name` | Filters that are evaluated within the table or derived table do not need the table ID. |
| Filter on a join tree | `table_id.field_name` | Filters that refer to fields in separate tables of the join tree need to use the table ID on each field name. |
| Calculated field in an Ad Hoc view | `"Field Label"` | Calculated fields declared in Ad Hoc views can refer to fields using their item **Label** entered with double quotes (`"`). Item labels are created on the Data Presentation tab of the Domain Designer. |
| | `table_ID.field_name` | Calculated fields declared in Ad Hoc views can refer to fields prefixed with their table ID. |

## 7.3    Operators and Functions

DomEL provides the following operators, listed in order of precedence. Operators higher in this list are evaluated before operators lower in the list:

| Operator | Syntax | Description |
|---|---|---|
| multiply, divide | `i * j / k` | Arithmetic operators for numeric types only. |
| percent | `i % j` | Calculates i as a percent of j; numeric types only. |
| add, subtract | `i + j - k` | Arithmetic operators for numeric types only. |
| equal | `i == j` | Comparison operators for string, numeric, and date types. |
| not equal | `i != j` | |
| less than | `i < j` | Comparison operators for numeric and date types only. |
| less than or equal | `i <= j` | |
| greater than | `i > j` | |
| greater than or equal | `i >= j` | |

| Operator | Syntax | Description |
|---|---|---|
| IN *set* | `i IN ('apples','oranges')` | Sets can be of any type. |
| IN *range* | `i IN (j:k)` | Ranges must be numeric or date types. |
| NOT | `NOT( i )` | Boolean operators. Parentheses are required for `NOT`. |
| AND | `i AND j AND k` | |
| OR | `i OR j OR k` | |
| parentheses | `()` | Used for grouping. |

DomEL also defines the following operations as functions.

| Function | Syntax | Description |
|---|---|---|
| startsWith | `startsWith(i, 'prefix')` | Comparison operators for strings. |
| endsWith | `endsWith(j, 'suffix')` | |
| contains | `contains(k, 'substring')` | |
| concat | `concat(i, ' and ', j, ...)` | Returns the string of all parameters concatenated. |

Additional functions are supported in Ad Hoc views.

## 7.4    SQL Functions

DomEL is the primary language used inside the tags in the Domain design file. You can use SQL in the following situations:

- The `query` tag in the derived tables section of the Domain design file expects an SQL function. See **4.2, "Derived Tables," on page 42** for more information.
- In addition, you may use SQL functions in a DomEL expression, but only in limited circumstances:
  - The functions must be supported by the database. See the vendor documentation for available functions and their syntax.
  - The functions must follow the convention of comma-separated parameters. For example, you can use `TRIM(person.name)`, but not `TRIM('Jr' FROM person.name)`
  - The type of the return values must be appropriate, either within the expression or for the type of the calculated field.
  - The SQL context must be appropriate for the functions. For example, you cannot use aggregation functions such as `COUNT` in a calculated field because there is no `GROUP BY` clause.

Except for the comma-separated parameter pattern, the DomEL validation cannot enforce these criteria. You must ensure that any SQL functions meet these criteria, otherwise the expression causes errors when using the Domain to create a report.

## 7.5     The groovy() Function

Groovy is an interpreted language for the JVM. Domains and DomEL use Groovy in the following ways:

- The `<principalExpression>` tag in an access grant in the Domain Security file uses a Groovy expression to get the current authentication object and determine its access privileges, along with the user and roles associated with the object. In this case, Groovy is used directly inside the tag. See **Chapter 8, "Securing Data in a Domain," on page 111** for more information.

- You can use the DomEL `groovy()` function as part of a DomEL expression. The DomEl `groovy` function takes a single string argument that is interpreted as Groovy code. The output of the function is a string that is put into the SQL. To include Groovy, use the following syntax:

  ```
  groovy('your groovy code here')
  ```

- For example, the following simple expression could be used to set the value of the calculated field `e.groovyEval` to the SQL string corresponding to the value of 5.0/6:

  ```
  <field id="e.groovyEval" dataSetExpression="groovy('(5.0/6).toString()')"
  type="java.lang.String" />
  ```

DomEL validation cannot check your Groovy code. You must ensure that any Groovy code meets these criteria, otherwise the expression causes errors when using the Domain to create a report.

## 7.6     Complex Expressions

Complex expressions are written by grouping any of the operators or functions above. Parentheses `()` may be used for grouping boolean operators, but arithmetic expressions that rely on parentheses are not supported. To compute complex arithmetic expressions, you may need to define several expressions as separate calculated fields, and then reference them in a simpler expression in another calculated field.

The following examples show complex expressions suitable for filters. This first one selects only stores in Western states of the US:

```
s1.store_country in ('USA') and s1.store_state in ('WA', 'OR', 'CA', 'NV')
```

The following filter expression uses a date range:

```
s1.first_opened_date in ( Date( '2000-01-01' ) : Date( '2004-12-31' )) and not(
s1.closed )
```

As shown in these examples, field values are often compared to constant values such as `'USA'`. Therefore, the author of the design file must ensure that values used in a DomEL expression exist in the data source. Otherwise, a wrong value might go undetected and impact the quality of data in reports based on the Domain. The Domain Designer determines values for comparison by accessing the data source, so if you export a design file, you can use the values it has found. Another way to reduce errors and also support future data changes is to use more general expressions such as:

```
s1.store_country in ('US', 'USA', 'United States')
```

### 7.6.1     Return Types

DomEL expressions are used in different contexts for different purposes. The expected return type depends on where the expression appears:

| Appears In | Expected Return Type | Explanation |
|---|---|---|
| Join expression | Boolean | Within a join expression, the `on` clause contains a comparison of fields from each table that has a boolean result. The `on` clause may also contain other DomEL expressions logically associated with `and` or `or` to create a complex join condition. |
| Calculated field | any type | The expression must evaluate to a type that is compatible with the SQL type declared in the Domain Designer or in the design file. For example, if the declared type is `java.lang.Float`, the expression must compute a decimal value. |
| Pre-filter | Boolean | Filters must be true or false overall. When there are several conditions, they must be logically associated with `and` or `or` (currently, only `and` is supported in the Domain Designer). |

# CHAPTER 8    SECURING DATA IN A DOMAIN

You may need to restrict access to the data in a Domain accessed by multiple users. For example, you may allow managers to analyze data across their department but allow individual contributors to see only their own data. For this purpose, Domains support security files.

> 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 chapter describes tasks only administrators can perform.

When Domain security is properly configured, a user sees only the data they're meant to see. You define Domain security by writing data access filtering rules in XML and uploading them as a new security file in the Domain Designer. These rules are powerful and flexible, and can be based on multiple aspects like user roles or attributes.

The power of this solution is best presented as an example business case. This section describes a fictional company's implementation of Domains in JasperReports Server—from both a business perspective and an implementation perspective.

> As of reslease 6.0, JasperReports Server supports hierarchical attributes (user, organization, and server levels). The examples in this chapter still work, but they do not demonstrate the cascading functionality of hierarchical attributes. See **6.10, "Using Server Attributes in Design Files," on page 100** for information on implementing hierarchical attributes.

For details about the basics of Domains, see **Chapter 2, "Understanding Domains," on page 9**, **Chapter 3, "Working with the Domain Designer," on page 13**, **Chapter 5, "Working with Domain Files," on page 63**, and **Chapter 6, "XML Design File Reference," on page 67**. For information about how recent changes to application configuration may effect Domain security, see the *JasperReports Server Security Guide*.

This chapter includes the following sections:
* **Business Case**
* **Process Overview**
* **Sales Domain**
* **Roles, Users, and Attributes**
* **Setting Up Logging and Testing**
* **Creating a Domain Security File**
* **Testing and Results**
* **Domain and Security Recommendations**

## 8.1    Business Case

CZS is an up-and-coming consumer electronics company with operations in the U.S. and Japan. CZS uses JasperReports Server to track sales revenue and operating cost.

The CZS Sales organization employs the following personnel:

- Rita is the regional sales manager in the Western U.S. She uses the Sales Domain to create reports that track sales trends in her region.
- Pete is a sales representative selling televisions in Northern California. He uses reports based on the same Domain to track his quarterly progress.
- Yasmin is a sales representative selling cell phones in Northern California. She uses reports based on the same Domain to track her quarterly progress.
- Alexi is the regional sales manager in Kansai, Japan. He uses reports based on the same Domain to track sales trends in his region.

CZS stores its data in a MySQL database. The data is exposed by the Sales Domain, which displays information about CZS's consumer electronics sales across the world. It's filtered depending on each employee's cities of operation and product. And only managers can access cost information.

## 8.2    Process Overview

The table below summarizes the steps CZS could take to create the Sales Domain and configure it to secure their data using user attributes and roles.

| Steps | Described in… |
|---|---|
| 1.  Define a Domain. The CZS business case is met by a Sales Domain that includes the following fields from their JDBC data source: city, state, product department, sales amount, cost amount, and unit sales. | **8.3, "Sales Domain," on page 113** |
| 2.  Identify and create access roles. CZS needs two roles: one for managers, and another for sales representatives. Both are granted access to the Sales Domain. | **8.4.1, "Roles," on page 114** |
| 3.  Create users and assign appropriate roles to each one. | **8.4.2, "Users," on page 115** |
| 4.  Identify and create attributes that determine each user's access to data in the Domain. CZS needs two attributes: `Cities` and `ProductDepartment`. | **8.4.3, "User Attributes," on page 115** |
| 5.  Prepare to test the security implementation by enabling logging and creating an example report. | **8.5, "Setting Up Logging and Testing," on page 116** |
| 6.  Iteratively create, upload, and test an XML file that defines the access granted to users based on the attributes defined in **step 4**. | **Creating a Domain Security File** |
| 7.  Test the Domain as various users. | **Testing and Results** |

## 8.3    Sales Domain

The first step is to create a Domain that presents the relevant data. CZS is primarily interested in the volume and revenue of their sales, as well as their operational cost. These metrics are represented in the Sales Domain as fields: unit sales, store sales, and store cost. The Domain also includes fields to establish context for the sales data, such as product department, city, and state. The following figures show the configuration of this Domain in the designer.
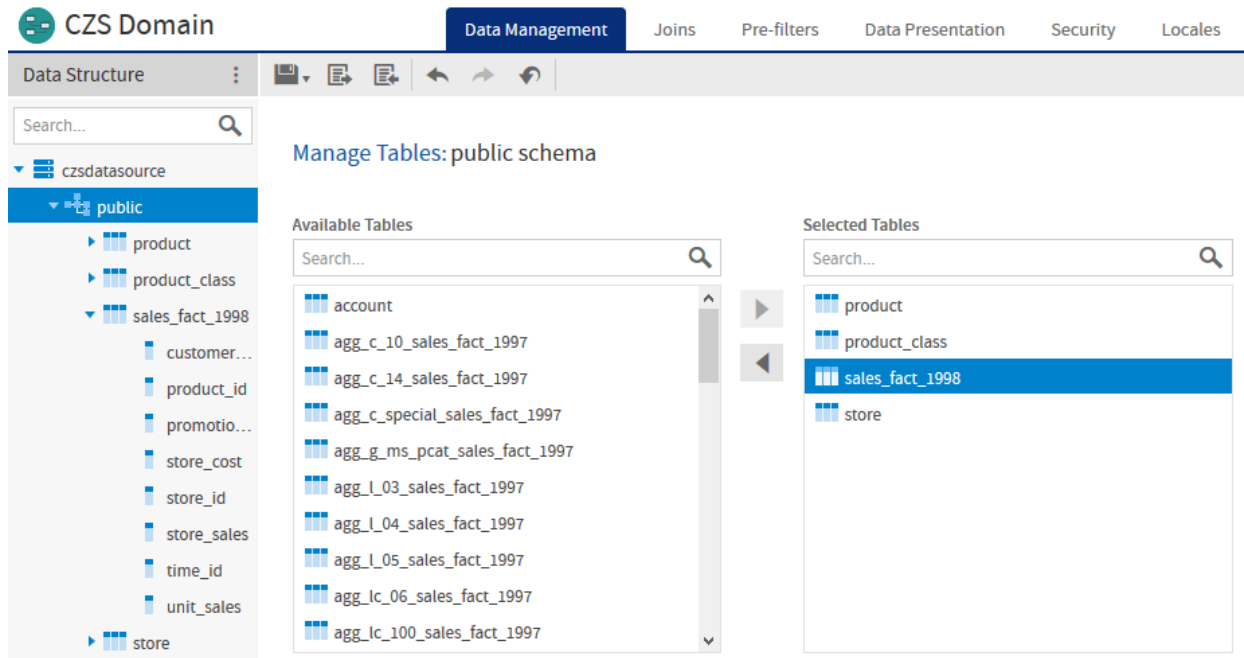


**Figure 8-1**  **Data Management Tab for the CZS Domain**
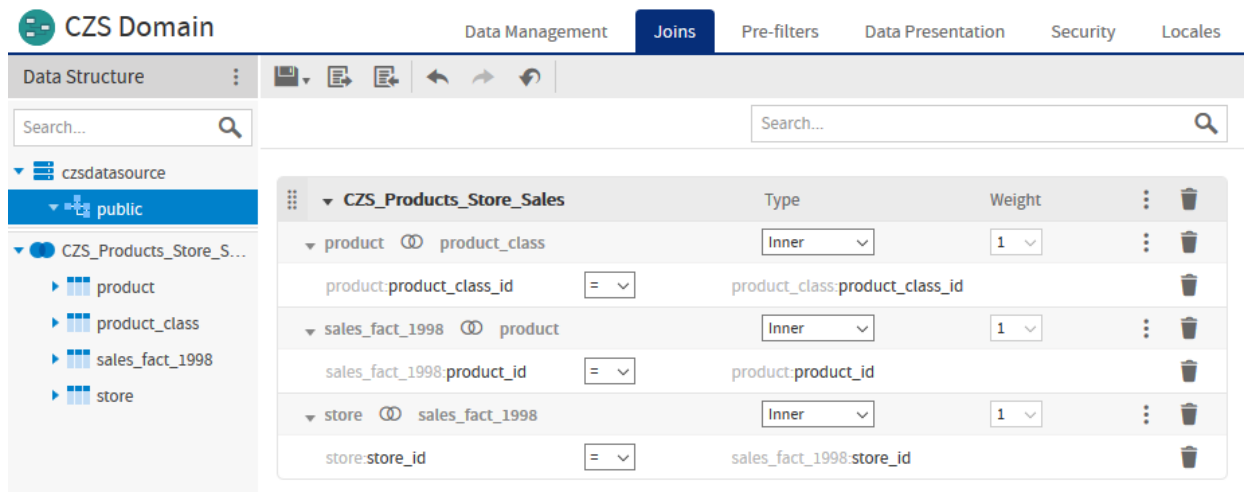


**Figure 8-2**  **Joins Tab for the CZS Domain**
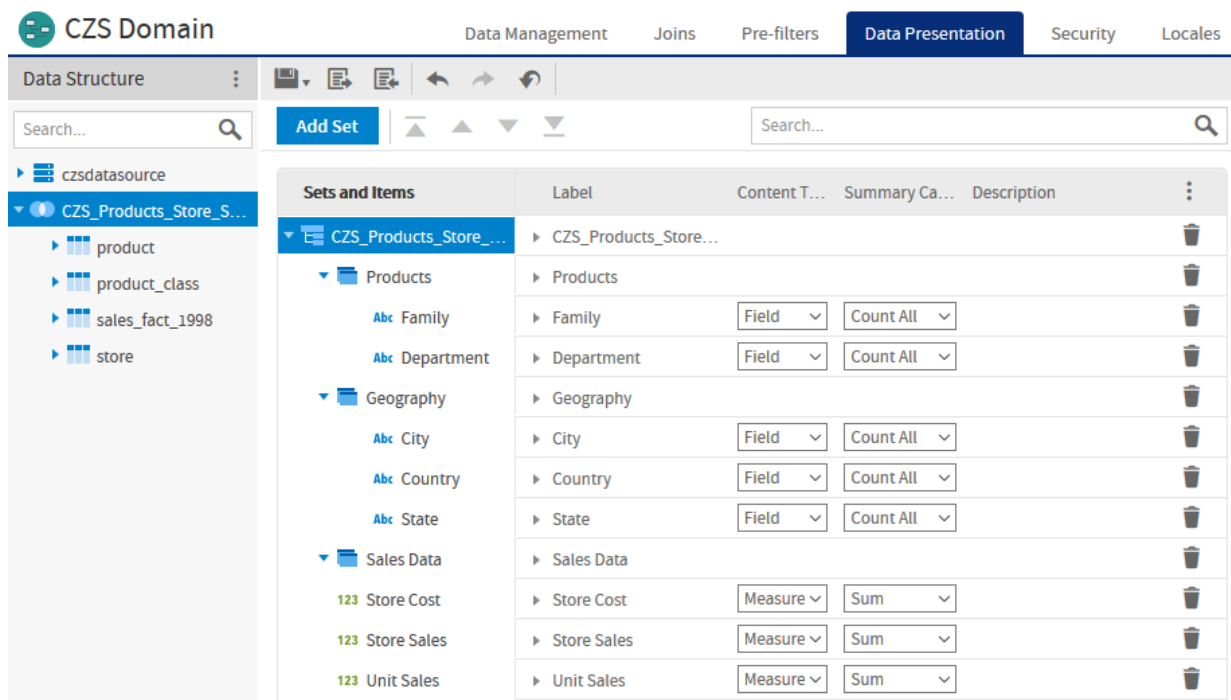
**Figure 8-3  Data Presentation Tab for the CZS Domain**

# 8.4    Roles, Users, and Attributes

## 8.4.1    Roles

Domain security can reference a user's roles to determine the access permissions to grant. The following roles meet CZS's needs:

- ROLE_SALES_MANAGER is assigned to sales managers.
- ROLE_SALES_REP is assigned to sales representatives.

CZS grants each role access to view the Sales Domain. For details about creating roles and assigning privileges, refer to the *JasperReports Server Administrator Guide*. The following shows CZS's ROLE_SALES_REP:
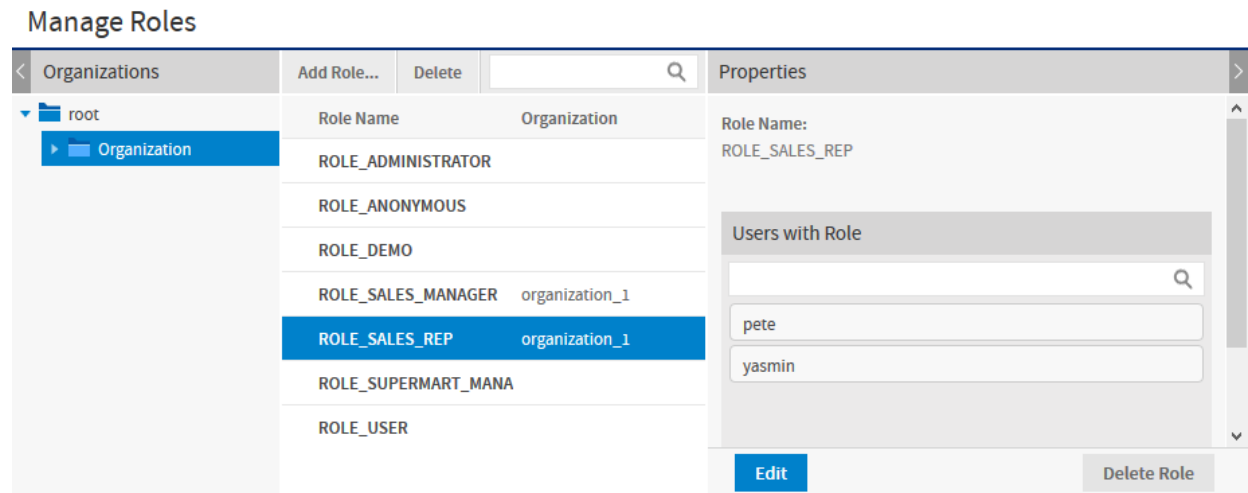
**Figure 8-4**  CZS Sales Representative Role

## 8.4.2    Users

CZS created a user for each of their employees and assigned roles based on each employee's level of responsibility:

| User | Role |
| --- | --- |
| Alexi | ROLE_SALES_MANAGER |
| Pete | ROLE_SALES_REP |
| Rita | ROLE_SALES_MANAGER |
| Yasmin | ROLE_SALES_REP |

For details about creating users, refer to the *JasperReports Server Administrator Guide*.

## 8.4.3    User Attributes

A user attribute is a name-value pair defined at the user level that corresponds to some data in a Domain. CZS wants to be able to describe their users in terms of product lines that they sell and the cities where they sell them. So each user is assigned two attributes in addition to a role:

• The `Cities` profile attribute corresponds to the City field in the Geography item group in the Sales Domain.
• The `ProductDepartment` attribute corresponds to the Department field in the Product item group in the Sales Domain.

**Table 8-1  UserAttributes of All CZS Users**

| User | Profile Attributes | |
| --- | --- | --- |
| | **Cities** | **Product/Department** |
| Rita | San Francisco, Los Angeles, Sacramento | Television, Wireless Devices |
| Pete | San Francisco | Television |
| Yasmin | San Francisco | Wireless Devices |
| Alexi | Osaka, Sakai | Wireless Devices |

When these attributes are used in the security file in an access grant definition, each user's attributes determine the data returned for them by the Domain. For example, Rita's attribute value for `Cities` is `San Francisco,Los Angeles,Sacramento`. So she sees data for all those cities.

The following figure shows the configuration of Rita's user account. Notice Rita's attributes listed below her roles:
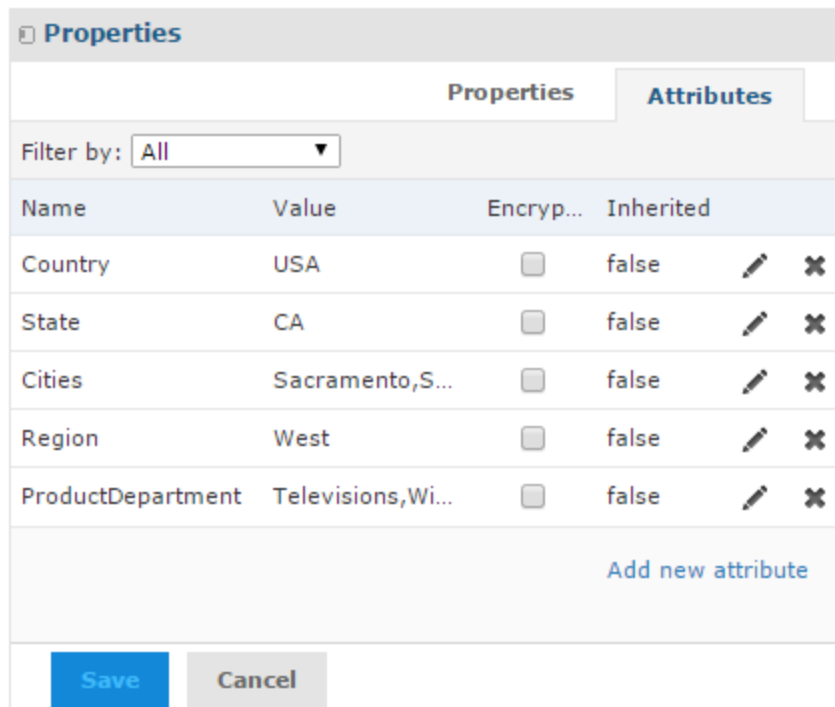


**Figure 8-5  CZS User Rita's Configuration**

## 8.5    Setting Up Logging and Testing

Before creating a security file, CZS prepares for the implementation by:

- **Enabling Logging**
- **Creating a Test Report**

## 8.5.1    Enabling Logging

To assist in the iterative creation of their security file, CZS enables more verbose logging to help troubleshoot problems with the Sales Domain and security file. Such logging features are disabled by default to minimize the log size. They should be enabled in test environments when defining security.

**To enable Domain security logging:**

1.  Locate and open the `log4j.properties` file and scroll to the bottom.

    You'll find this file in the WEB-INF folder; if you use Tomcat as your application server, the default path to this location is:

    <js-install>\apache-tomcat\webapps\jasperserver-pro\WEB-INF.

2.  Add the following lines after the last line in the file:

```
log4j.logger.com.jaspersoft.commons.semantic.datasource.impl.
    SemanticLayerSecurityResolverImpl=debug
log4j.logger.com.jaspersoft.commons.semantic.dsimpl.JdbcTableDataSet=DEBUG, stdout, fileout
log4j.logger.com.jaspersoft.commons.util.JSControlledJdbcQueryExecuter=DEBUG, stdout, fileout
```

3.  Save the file.
4.  Restart JasperReports Server.

Information about Domains and their security will now be written to the log and to the console.

> The additional information written to the log can be very verbose, and your log files will grow more quickly with these properties enabled. You can manage your logs in the file system, in the WEB-INF/logs folder under your JasperReports Server installation. For more information, refer to the log4j documentation, which is available at:
>
> http://logging.apache.org/log4j/docs/manual.html
>
> Because these options are so verbose, we recommend using them only during debugging and disabling them in your production environment.

## 8.5.2    Creating a Test Report

CZS creates an Ad Hoc crosstab based on the Sales Domain to assist in testing the security file as they create each access grant. The report displays store sales amount, store sales cost, and store units sold for all cities and departments.



**Figure 8-6**  **Fields added to CZS Ad Hoc crosstab**

Each user's limited view of this report is shown in **Testing and Results**.

# 8.6 Creating a Domain Security File

A Domain's security file contains item and resource access grants that specify access based on certain aspects of a user, such as attributes or roles. Typically, access grants check a user's roles or attributes and grant access to the items (columns) and resources (rows) of the Domain. This mechanism tailors the data that is available to each user.

A Domain's security file has two types of access definitions:

- Row-level access determines which rows in the data source can be displayed to a specific user.
- Column-level access determines which columns in the data source can be displayed to a specific user.

This section describes the access grant syntax and illustrates both kinds of access grant.

> Note the comments in the XML examples in this section; for example: `<!-- Comment -->`. It's good practice to comment the access grants you define, and to format your XML neatly. We recommend using an XML editor when creating security files. See **Domain and Security Recommendations**.

## 8.6.1 Downloading the Security File Template

To begin writing your security file, download the template provided on the Security tab.

1. Open your Domain in the Domain Designer and navigate to the Security tab.

2. Click </> to display the generic template in the editor.
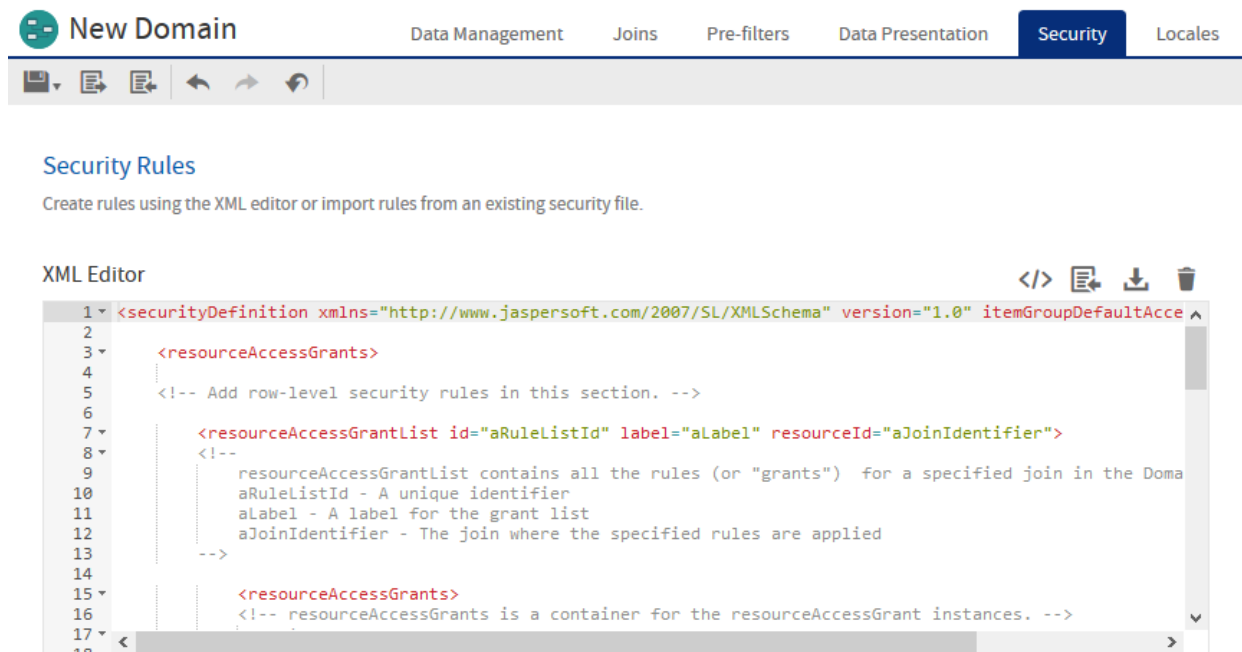


**Figure 8-7  Security Tab with the Generic Template**

3. Click ⤓ to download the template and save it as an XML file on your computer.

Modify the template as shown in the code samples in the following sections. The samples do not strictly follow the template, but the template gives you the structure of the file.

## 8.6.2    Access Grant Syntax

All access grants take a `principalExpression` that gets the user's attributes or roles and evaluates them. When the `principalExpression` evaluates to true, that means the column-level grant or row-level grant applies to the current user. Column-level grants then have a list of columns, called items, that are denied or granted access. Row-level grants have a `filterExpression` that filters rows based on the attributes or roles. In practice, when the `principalExpression` of a row-level grant is true, the `filterExpression` is added to the `WHERE` clause of the database query to restrict the rows that are returned.

You can use the following services in a `principalExpression` or `filterExpression`:
* `authentication.getPrincipal().getRoles()` – Use this function to access the roles of the current user.
* `attributesService` – Use to retrieve or test for an attribute you have defined at the user, organization, or server level.

The following service is only used in a `filterExpression` to filter rows:
* `testProfileAttribute` – Compares a given attribute to a field in the data.

### 8.6.2.1  The getRoles() Function

To retrieve information about roles, you must access Spring's currently authenticated principal object. You do this using `authentication.getPrincipal().getRoles()` to get a list of all roles defined for the user. Your expression must process this list to compare it to the desired role names. For example, the following principal expression checks whether the user has the ROLE_ADMINISTRATOR or ROLE_SALES_MANAGER role.

```
<principalExpression>
  authentication.getPrincipal().getRoles().any{ it.getRoleName()
    in ['ROLE_ADMINISTRATOR','ROLE_SALES_MANAGER'] }
</principalExpression>
```

The `authentication.getPrincipal()` function can access other information about the user, but Jaspersoft recommends using only the functions described in this section.

### 8.6.2.2  The attributesService

The `attributesService` retrieves the value of a given attribute for a user. A user can inherit attributes from their organization or the server in addition to any attributes assigned to the user directly. When specifying an attribute, you can either choose the category (user, tenant (organization), or server) in which the server should look for its value, or allow the server to locate the value hierarchically.

When determining an attribute hierarchically, the server first searches for attributes defined at the user level, then at the organization-level, then on any parent organizations, then finally at the server level. It will return the first value it finds, such that an attribute defined at a lower level can override the same attribute at a higher level.

This function has the following syntax:
```
attributesService.getAttribute('AttrName',Level[,required])
```
where:
* `AttrName` – String that specifies the attribute to check. This can be any customer-defined attribute, such as `Cities`.

- Level – Category that specifies a level in the hierarchy to check for attributes. One of: null, 'USER', 'TENANT', or 'SERVER'. To search for attributes hierarchically from all levels, use null.
- required (optional) – Boolean that specifies whether or not the attribute is required.
  - When set to true, an error message is displayed in the UI if the attribute is not present.
  - When set to false (default), if the attribute is not present, no error is thrown and the filterExpression is not performed. In this case, unfiltered information which the user is not explicitly authorized to view may be displayed.

> attributesService is implemented in Groovy. For more information about Groovy, see www.groovy-lang.org.

For example, the following expression tests whether the user has myValue set for myAttribute anywhere in the hierarchy.

```
<principalExpression>
  attributesService.getAttribute('myAttribute',null,true)?.getAttrValue().equals('myValue')
</principalExpression>
```

### 8.6.2.3  The testProfileAttribute Function

Within a filterExpression, you often want to compare an attribute to a database field value. The testProfileAttribute function provides an easy way to do so:

```
testProfileAttribute(table_ID.field_name, 'attribute'[,Level])
```

where:

- table_ID.field_name – The table name and field name of a field whose value you're comparing to an attribute.
- attribute – The name of an attribute.
- Level (optional) – A specific level where the attribute should be defined, one of 'USER', 'TENANT', or 'SERVER'. When this argument is omitted, the attribute value is determined hierarchically across all levels.

**Table 8-2  Filter expression using `testProfileAttribute`**

```
<filterExpression>testProfileAttribute(store1.store_country,'country')</filterExpression>
```

> JasperReports Server 6.0 added support for hierarchical attributes and changed the behavior of testProfileAttribute to use them by default.

You can also use attributesService in a filter expression. The following filter expression gives the same results as the filter expression in **Table 8-2**

**Table 8-3  Filter expression using `attributesService`**

```
<filterExpression>
  store1.store_country ==(groovy('attributesService.getAttribute("country", null).attrValue'))
</filterExpression>
```

## 8.6.3   Row-level Security

This section gives an overview of row-level security and then shows how CZS uses row-level security to restrict access based on Cities and ProductDepartment.

### 8.6.3.1  Understanding Row-level security

Row-level access determines which rows in the data source can be displayed to a specific user.

For example, consider a table that includes values for the cities where products are sold. You could define a resource access grant that finds users for which a city has been defined as a profile attribute and, for each such user, limits access to rows where the city value is the user's specific city.

For example, take Rita and Alexi. Both have the same role and the same access to the Sales Numbers analysis view, but CZS doesn't want them to see the same data—Rita should see data about San Francisco, Sacramento, and Los Angeles; and Alexi should see data about Osaka and Sakai. Without attributes, this would be possible only if CZS's access roles were defined along geographic lines.

> Each access grant ID must be unique within the scope of the security file.
>
> You can define several similar resource access grants for each resource defined in your Domain. By default, the server assumes access grants are combined with a logical AND. You can force the server to use a logical OR by setting the `orMultipleExpressions` property to TRUE.

To implement row security, CZS uses `attributesService` to check for attributes.

For example, CZS used the following XML to define a principal expression and filter expression that grant access to users based on their `Cities` profile attribute:

```
<resourceAccessGrant id="Jointree_1_row_access_grant_20">
  <principalExpression>attributesService.getAttribute('Cities', null, true) != null
  </principalExpression>
  <filterExpression>testProfileAttribute(store.store_city,'Cities')
  </filterExpression>
</resourceAccessGrant>
```

The principal expression gets the values of the `Cities` attribute for the logged-in user. Since `attributesService` supports hierarchical attributes, CZS set the `null` parameter to indicate that they want to look at the values from all levels. The optional `true` parameter ensures that if a user without any values for the Cities attribute accesses the view, they will receive an error.

The filter expression checks the user's `Cities` profile attribute as well, but it compares this value with the values in the Domain's `store_city` field. The Domain then returns all the rows that match the user's `Cities` profile attribute.

### 8.6.3.2  CZS's Resource Access Grants

CZS uses the access grant above to determine data access based on a user's `Cities` attribute. Because CZS defines all their attributes in the same manner, they can use a similar resource access grant to determine data access for users based on their `ProductDepartment` attribute.

The resulting security file included these two resource access grants.

```
<!-- Row level security -->
<!-- What access do roles/users have to the rows in the resource? -->
<resourceAccessGrantList id="JoinTree_1_List" label="ListLabel"
  resourceId="JoinTree_1">
  <resourceAccessGrants>
    <!-- Row level for Cities -->
    <resourceAccessGrant id="Jointree_1_row_access_grant_20">
```

```
      <principalExpression>attributesService.getAttribute('Cities', null, true) != null
        </principalExpression>
      <filterExpression>testProfileAttribute(store.store_city,'Cities')
        </filterExpression>
    </resourceAccessGrant>
    <!-- Row level for Product Dept -->
    <resourceAccessGrant id="Jointree_1_row_access_grant_30">
      <principalExpression>
        attributesService.getAttribute('ProductDepartment', null, true) != null
      </principalExpression>
      <filterExpression>testProfileAttribute(product_class.product_department,
        'ProductDepartment')</filterExpression>
    </resourceAccessGrant>
  </resourceAccessGrants>
</resourceAccessGrantList>
```

## 8.6.4    Column-level Security

Column-level access determines which columns in the data source can be displayed to specific users.

### 8.6.4.1  Understanding Column-level Security

Consider a table that includes employee contact and salary information. You could define item group access grants that check the user's role and grant access to the salary field only if the user has the Human Resources role. For example, the following code sample modifies access for the ROLE_SALESREP role, first by revoking the default access for that role and then granting access to sales information only. The principle expression determines which users the item group access grant applies to (users with the ROLE_SALES_REP role). The item access grants determine the specific access of the users. All role-specific access is revoked then access to the `StoreSales` and `StoreCost` item is granted:

```
<itemGroupAccessGrant id="Jointree_1_item_group_access_grant_2" access="granted">
  <principalExpression>authentication.getPrincipal().getRoles().any
    { it.getRoleName() in ['ROLE_SALES_REP'] }</principalExpression>
    <itemAccessGrantList id="Jointree_1_grant2_item_group_items"
      defaultAccess="denied">
      <itemAccessGrants>
        <itemAccessGrant id="Jointree_1_grant2_items_grant1" itemId="StoreSales"
          access="granted" />
        <itemAccessGrant id="Jointree_1_grant2_items_grant2" itemId="UnitSales"
          access="granted" />
      </itemAccessGrants>
    </itemAccessGrantList>
  </itemGroupAccessGrant>
</itemGroupAccessGrants>
```

## 8.6.5    CZS's Item Group Access Grants for Sales Data

To ensure that sales representatives don't have access to cost information, CZS adds item group access grants; the first grants full access to managers and the administrator:

```
<!-- Column-level access for Sales Manager and Admins-->
<itemGroupAccessGrant id="Jointree1_item_group_access_grant_MNG" access="granted">
  <principalExpression>authentication.getPrincipal().getRoles().any
```

```
    { it.getRoleName() in ['ROLE_ADMINISTRATOR','ROLE_SALES_MANAGER'] }
    </principalExpression>
</itemGroupAccessGrant>
```

CZS then adds an item group access grant that grants limited access to sales representatives; the following XML grants access to the Store Sales and Sales Units fields while revoking access to the Store Cost field:

```
<!-- Column-level access for Sales Reps-->
<itemGroupAccessGrant id="Jointree_1_item_group_access_grant_REP"
  access="granted">
  <principalExpression>authentication.getPrincipal().getRoles().any
    { it.getRoleName() in ['ROLE_SALES_REP'] }</principalExpression>
  <itemAccessGrantList id="Jointree_1_grant2_item_group_items"
    defaultAccess="denied">
    <itemAccessGrants>
      <itemAccessGrant id="Jointree_1_grant2_items_grant1" itemId="StoreSales"
        access="granted" />
      <itemAccessGrant id="Jointree_1_grant2_items_grant2" itemId="UnitSales"
        access="granted" />
    </itemAccessGrants>
  </itemAccessGrantList>
</itemGroupAccessGrant>
```

## 8.6.6    Uploading the Security File

CZS uploads the security file each time they add a new access grant. You can upload the security file when you add or edit a Domain.

You can also store the security file in the repository as a file resource, and then add it to your Domain. When stored in the repository, the security file can be shared with several Domains, as long as they have the same structure and IDs. This can be useful when the only differences between the Domains is managed with attributes, so a Domain can be copied to several similar organizations.

1.   Open your Domain in the Domain Designer and navigate to the Security tab.

2.   Click ⬅ to add the security file, then select the Repository or Local File tab at the top of the dialog.
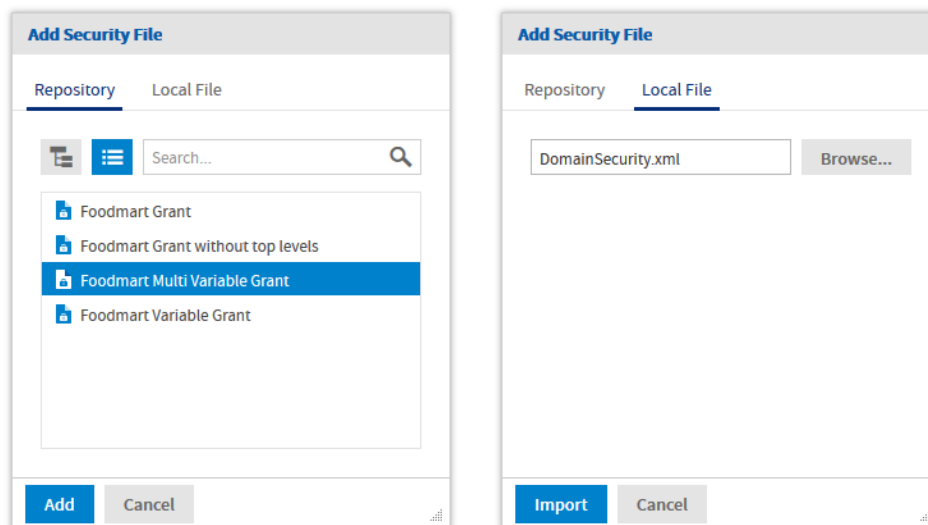
**Figure 8-8  Add Security File Dialogs**

3.  If you stored the security file in the repository, use the controls to navigate the tree or search for your file, then click **Add**.

    If the file is on your computer, browse to select it, then click **Import**.

4.  The server uploads the file and validates it. If there was any XML in the Security tab editor, you are prompted to replace it.

The editor checks the XML syntax and Domain IDs of your file and lets you fix any errors. Upon saving the Domain, the contents of the editor are validated for join references and principal expressions, then becomes the Domain's security file.

Use the same procedure to modify or replace the security file when you make any changes to the Domain that affect the access grants.

## 8.7     Testing and Results

Finally, CZS verifies Domain access as various users by clicking the **Login as User** button on the Manage Users page.

**To test the access granted to users on data in the Domain:**

1.  Log in as administrator (`jasperadmin`) if necessary.
2.  Click **Manage > Users**.
3.  In the list of user names, click the name of the user you want to test.
4.  In the User page, click **Log in as User**. The selected user's Home page appears.
5.  Click **View > Reports**.
6.  In the list of reports, click the test report you created when defining your security file.
7.  Review the report to ensure that it shows only the data this user should see. Also verify that you have not restricted data that the user should see. The figures below show CZS's results.
8.  Click **Logout** to return to the administrator view.

When viewing the test report created from the Sales Domain:

- Rita can see all data pertaining to California and the three Californian cities where CZS has offices (Los Angeles, Sacramento, and San Francisco):

**Sales Data by City**

| Department | Measures | Los Angeles | Sacramento | San Francisco | Totals |
|---|---|---|---|---|---|
| Televisions | Store Sales | 5,065.10 | 4,823.88 | 4,314.26 | 14,203.24 |
| | Store Cost | 2,014.67 | 1,953.55 | 1,694.05 | 5,662.27 |
| | Unit Sales | 2,560.00 | 2,422.00 | 2,120.00 | 7,102.00 |
| Wireless Devices | Store Sales | 39,305.74 | 39,187.46 | 36,699.97 | 115,193.17 |
| | Store Cost | 15,677.91 | 15,589.18 | 14,713.26 | 45,980.35 |
| | Unit Sales | 18,369.00 | 18,294.00 | 16,993.00 | 53,656.00 |
| Totals | Store Sales | 44,370.84 | 44,011.34 | 41,014.23 | 129,396.41 |
| | Store Cost | 17,692.58 | 17,542.74 | 16,407.31 | 51,642.62 |
| | Unit Sales | 20,929.00 | 20,716.00 | 19,113.00 | 60,758.00 |

**Figure 8-9**  **Rita's view of the CZS Test Report**

- Pete can see only Television data about San Francisco; he sees zeros for Store Cost because he is denied access to that field:

**Sales Data by City**

| Department | Measures | San Francisco | Totals |
|---|---|---|---|
| Televisions | Store Sales | 4,314.26 | 4,314.26 |
| | Store Cost | 0.00 | 0.00 |
| | Unit Sales | 2,120.00 | 2,120.00 |
| Totals | Store Sales | 4,314.26 | 4,314.26 |
| | Store Cost | 0.00 | 0.00 |
| | Unit Sales | 2,120.00 | 2,120.00 |

**Figure 8-10**  **Pete's View of the CZS Test Report**

- Yasmin can see only Wireless Devices data about San Francisco; she sees zeros for Store Cost because she is denied access to that field:

**Sales Data by City**

| Department | Measures | San Francisco | Totals |
|---|---|---|---|
| Wireless Devices | Store Sales | 36,699.97 | 36,699.97 |
| | Store Cost | 0.00 | 0.00 |
| | Unit Sales | 16,993.00 | 16,993.00 |
| Totals | Store Sales | 36,699.97 | 36,699.97 |
| | Store Cost | 0.00 | 0.00 |
| | Unit Sales | 16,993.00 | 16,993.00 |

**Figure 8-11**  **Yasmin's view of the CZS Test Report**

- Alexi can see Wireless device data pertaining to the two Japanese cities where CZS has stores (Osaka and Sakai):

## Sales Data by City

| | City | Osaka | Sakai | Totals |
|---|---|---|---|---|
| **Department** | **Measures** | | | |
| Wireless Devices | Store Sales | 39,619.66 | 62,945.01 | **102,564.67** |
| | Store Cost | 15,800.79 | 25,166.66 | **40,967.45** |
| | Unit Sales | 18,632.00 | 29,905.00 | **48,537.00** |
| Totals | Store Sales | **39,619.66** | **62,945.01** | **102,564.67** |
| | Store Cost | **15,800.79** | **25,166.66** | **40,967.45** |
| | Unit Sales | **18,632.00** | **29,905.00** | **48,537.00** |

**Figure 8-12** **Alexi's view of the CZS Test Report**

- Finally, make sure that any user who doesn't have the Cities attribute set can't see any data. For example, joeuser receives an error:
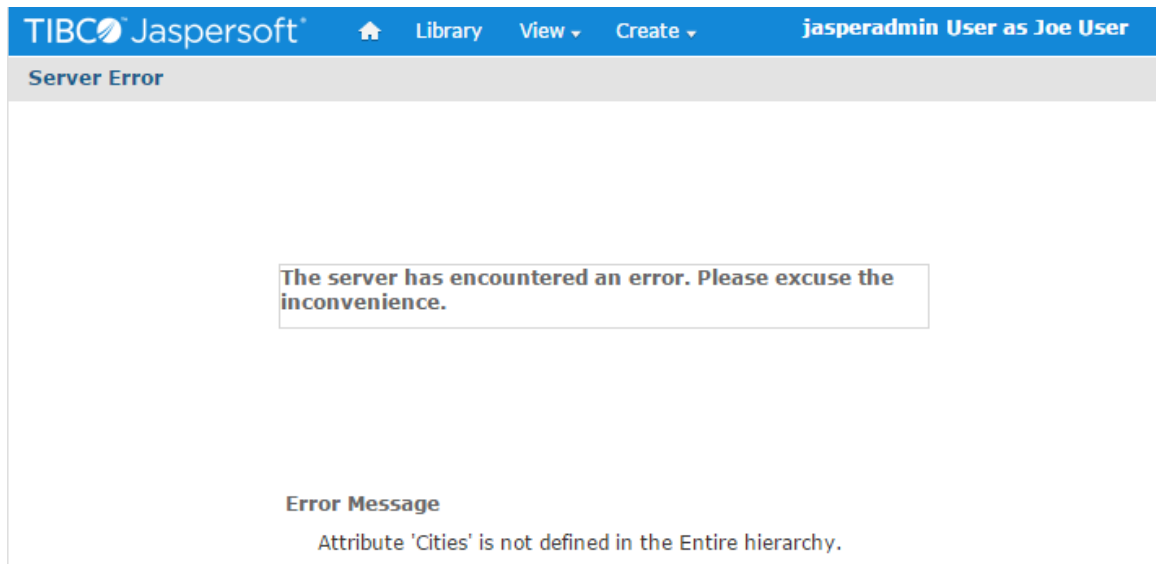
**Figure 8-13** **joeuser's view of the CZS Test Report**

## 8.8    Domain and Security Recommendations

When defining a Domain and its security, keep these recommendations in mind:

- A Domain should cover a large subject area and include data with multiple uses. Define joins to create data islands that each contain related information; the data islands themselves can contain completely unrelated data. For example, you could include both human resources and sales data in a single Domain; users would see

only the information relevant to their job responsibilities. For an example of this type of Domain, refer to the SuperMart example that can be installed with JasperReports Server.

- When defining a Domain, don't create too many item groups, and avoid very deep structures with many levels. Such complexity makes the Domain harder to use.
- Logging can help you troubleshoot any problems you encounter while implementing Domain security. For more information, refer to **Enabling Logging**.
- Refer to http://groovy.codehaus.org for information on the Groovy expressions that Domain security files support. Note that, while the server does validate Groovy expressions, the validation is very light weight and doesn't detect all improperly formed expressions.
- If the names of tables and fields in your data source change, you can edit the Domain design XML file so that the resource names match the new names in the database. Then, upload the new version of the file; your reports that rely on the Domain will work properly without being updated individually. If you have defined a security file for this Domain, you must also edit the resource names in the security file.
- Start with the simplest item or resource grant, and when that works, expand upon it. Start simple and iterate until you have the full set of access grants needed. Follow good troubleshooting practices, such as changing only a single aspect of the security file before testing the results of the change.
- Use an XML editor to create your security file. While the server validates the schema against its own XML definition, a typical XML editor can identify issues like unclosed tags. For example, open the security file with Internet Explorer; if it returns errors, use them to identify and correct your XML.
- Once your Domain is created, create several Domain Topics that focus on specific aspects of the Domain or specific data your end-users will want to review regularly. To do so, click **Create > Ad Hoc Report**, select your Domain, and use the Data, Filters, and Display pages to customize the contents and the way it's displayed, then use the Topics page to save the new Domain Topic.
- When creating a security file, be sure to use the IDs of items and groups as they are defined in the Domain design file exported from the Domain Designer. For more information.
- If you modify the Domain, you should also export the design file and update the security file with any IDs that have changed. Update the security file using the **Change** function on the Edit Domain page of the Domain Designer.

A typical security file has the following structure:

```
<securityDefinition xmlns="http://www.jaspersoft.com/2007/SL/XMLSchema"
          version="1.0" itemGroupDefaultAccess="granted">
 <resourceAccessGrants>    <!-- Begin row-level security -->
  <resourceAccessGrantList id="expense_join_resource_access_grant" label="aLabel"
              resourceId="expense_join">
   <resourceAccessGrants>
    <resourceAccessGrant id="expense_join_ROLE_SUPERMART_MANAGER_store_row_grant">
     <principalExpression>
      authentication.getPrincipal().getRoles().any{ it.getRoleName() in
      ['ROLE_SUPERMART_MANAGER'] }
     </principalExpression>
     <filterExpression>s.store_country in ('USA') and s.store_state in ('CA')
     </filterExpression>
    </resourceAccessGrant>
    ...
   </resourceAccessGrants>
  </resourceAccessGrantList>
  ...
 </resourceAccessGrants>
 <itemGroupAccessGrants>    <!-- Begin column-level security -->
```

```
 <itemGroupAccessGrantList id="expense_join_item_group_access_grant_group"
       label="aLabel" itemGroupId="expense_join" defaultAccess="denied">
  <itemGroupAccessGrants>
   <itemGroupAccessGrant id="expense_join_super_user_item_group_grant"
              access="granted">
    <principalExpression>
     authentication.getPrincipal().getRoles().any{ it.getRoleName() in
     ['ROLE_ADMINISTRATOR'] }
    </principalExpression>
   </itemGroupAccessGrant>
   ...
  </itemGroupAccessGrants>
 </itemGroupAccessGrantList>
 ...
 </itemGroupAccessGrants>
</securityDefinition>
```

# CHAPTER 9    LOCALIZING DOMAINS

Domains let you organize and label your data for your users. You can give your sets and items meaningful names either through the Domain Designer or in the Domain XML file. When report designers use the Domain in the Ad Hoc editor, and when end-users view a generated report, they see the names of fields and measures that you have defined. Domains also proved a simple mechanism to translate these labels into other languages, as part of the process called localization. When a user logs in with a different locale, or a report is generated in a different locale, the translated labels appear automatically in the interface and report output.

The translated strings for a given language are defined in a properties file called a locale bundle. The Locales tab of the Domain Designer is where you manage locale bundles, allowing you to upload, attach, or remove locale bundles from the Domain. You can also download the template for locale bundles when you are ready to translate your Domain into another language. This chapter explains the syntax of the locale bundle, how to create one, and how to upload them to the Domain.

This chapter contains the following sections:
- **Planning for Localization**
- **Creating Locale Bundles**
- **Properties File Syntax**
- **Adding Locale Bundles to a Domain**

## 9.1    Planning for Localization

Creating and editing translation files for Domains can be a complicated process that often happens incrementally. The best strategy for you depends on your deployment. The following sections describe a common workflow that is useful for most situations.

Each label and description in the Domain has an internationalization key that names a property in a properties file, and the value of that property is the translated string for the item.

In the Domain Designer, the name of the internationalization keys are defined by the **Label Key** and **Descriptions Key** properties on each set and item on the Data Presentation tab. By default, these properties are blank. You can name the keys in any way you want, as long as each key is unique among all keys within the Domain. However, because keys are used in a Java properties file, they may only use characters from the ISO Latin-1 set, digits, and underscores (_).

You can create and edit the properties file in any text editor. Some editors provide syntax highlighting for editing properties files. There are also Java translation editors that manage all of the properties files together, allowing you to translate in parallel and find missing keys and missing translations.

### 9.1.1    Designing a Domain for Localization

1. If you know you will be translating, create your labels and descriptions in a properties file instead of in the Domain Designer. This makes it easier to create and edit additional properties files in the future. You can do this even if you are not translating at this time. See **9.3.2, "Property Values," on page 134** for more information on evaluation order of values.

> If you have no current need for translation, you can create your labels and descriptions within the Domain Designer. If you later need to translate, you can create key names and properties file as you need them.

2. Decide whether you want to explicitly assign keys for each element, or if you want to use generated keys.

3. If you are using generated keys, give the data islands and sets in your presentation meaningful **ID**s. It is easier to work with keys such as `ACCOUNTS_MASTER.ACCOUNTS.LABEL` rather than `JOINTREE_1.SET1.ACCOUNTS.LABEL`.

### 9.1.2    Overview of Localizing a Domain

1. Once you have finalized the naming of the keys or **ID**s, use the **download the template** link on the Locale tab in the Domain Designer to generate a file with the keys you want. See **9.2.1, "Downloading the Bundle Template," on page 131** for more information.

2. Using the downloaded file as a starting point, create a file with blank values as a template for your properties files. If you did not explicitly define any labels or definitions, the values in the downloaded file are blank. If you have explicitly defined values, you need to remove them to create a file with blank values.

3. Make sure to save a properties file with blank values or the default locale bundle as a template for future translations.

4. Create a master file with the strings for your base locale. Set this as the default properties file.

> Make sure to create a default properties file. If every file has a locale code in the filename, the values will not appear for users in unsupported locales.
>
> If you prefer, you can define labels and descriptions in the Domain Designer, which display in unsupported locales. However, this makes it harder to create and maintain a blank version of the properties file.

5. For each additional locale you want to support, copy the blank file and change the `_<locale>` designator to create a basis for the translated file. For information about file naming, see **9.2.2, "Properties File Names," on page 132**.

6. Use specialized software or a translation service to create the Java localization properties files for each locale you want to support.

7. Use the Locales tab in the Domain Designer to attach the properties files to the Domain, as described in **9.4, "Adding Locale Bundles to a Domain," on page 135**.

8. Make sure to save a blank properties file or the default locale bundle as a template for future translations.

## 9.2    Creating Locale Bundles

To localize Domain labels and descriptions, you must create a properties file for each locale you want to support. You can also create an optional "default" file that holds untranslated text common to all locales. The default file is also used as a fallback whenever a message is missing in the properties file for the current locale.

To get started, download the template file.

## 9.2.1  Downloading the Bundle Template

The Domain Designer lets you download a template with all keys currently defined in the Domain.

1.  Open the Domain for editing and navigate to the Locales tab.
2.  Click **download the template**.
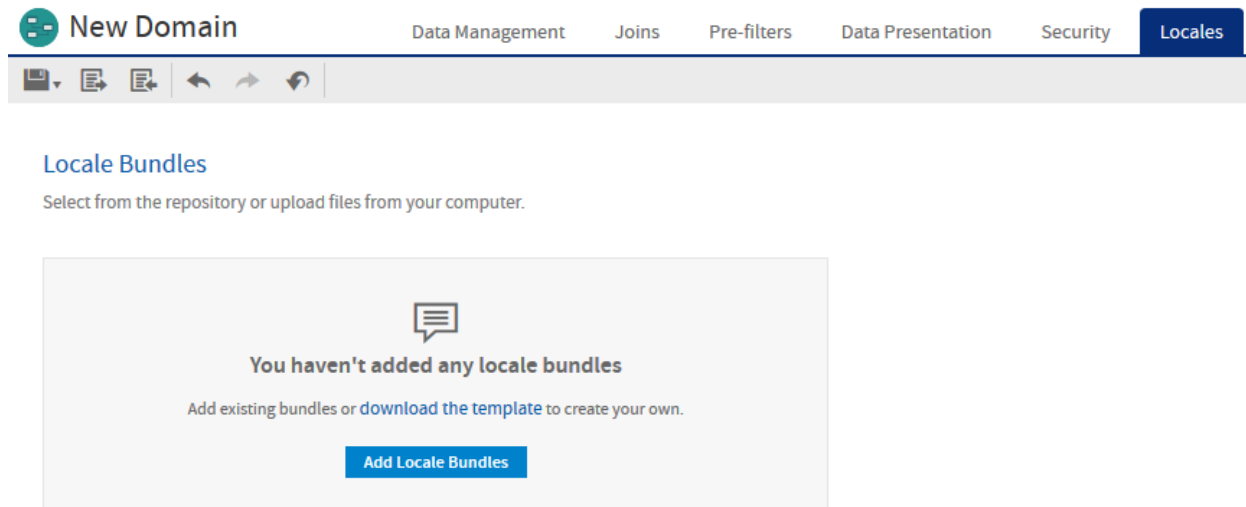


**Figure 9-1**  **Template Download Link on the Locales Tab**

3.  In the File Options dialog box, select which keys to export:
    - **Generate missing label keys** – Automatically generates label key names for any elements where **Label Key** is not defined.
    - **Generate missing description keys** – Automatically generates description key names for any elements where **Description Key** is not defined.
    - When both options are deselected, keys are not generated automatically. Only keys explicitly defined as **Label Key** or **Descriptions Key** are exported.
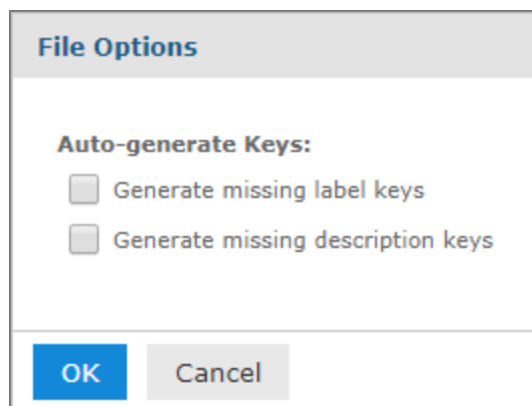
**Figure 9-2** **File Options for Downloading a .properties Template**

4. Click **OK**.

The bundle is exported with a generated file name such as "domain name.properties". You can change the name to match the name you want to use for your properties files.

The file contains the following:

- All explicitly defined keys.
- Optional generated keys, if selected. For the format of generated keys, see **9.3.1, "Key Names," on page 133**.
- Any values that are explicitly defined as the **Label** or **Description** field for an element on the Presentation tab, or the `labelId` or `descriptionId` attributes on an `itemGroup` or `item` element in the XML design file.

The exported file is a Java properties file in the proper format containing the selected keys. If you did not set any label or descriptions in the Domain Designer or XML design file, then the file contains blank values and is ready for translation.

> If the design file does not load properly in the Domain Designer, you are not able to use the exported design file. In this case, do not overwrite the design file, but save the exported file to another name and attempt to merge in the generated keys.

## 9.2.2 Properties File Names

The file names of a locale bundle is of the form <any_name>_<locale>.properties, where:

- <any_name> is arbitrary and should be the same for all bundle files in a Domain.
- <locale> is any Java-compliant locale identifier, for example `fr` or `fr_CA`.
- You can optionally include a default .properties file, which is used for the default locale or when a key is missing or empty in one of the other .properties files. The default properties file name is <any_name>.properties. It does not have a locale identifier.

The following table shows an example relationship between file names, key names, and values in two files, a default file in English and a file of French translations.

| Language | Locale | File Name | Sample Contents |
|---|---|---|---|
| English | default | Labels.properties | `JOINTREE_1.SET1.STORE_CITY.LABEL=City`<br><br>`JOINTREE_1.SET1.STORE_CITY.DESCR=City or town where the store is located` |
| French | fr | Labels_fr.properties | `JOINTREE_1.SET1.STORE_CITY.LABEL=Ville`<br><br>`JOINTREE_1.SET1.STORE_CITY.DESCR=Ville ou commune d'exploitation` |

## 9.3    Properties File Syntax

The locale bundle for a JasperReports Server Domain is a standard Java properties file. Inside the properties files, each entry is of the form:

```
key=value
```

Each .properties file is a text file containing these key-value pairs for a locale:

- The keys, when present, are the same across all the .properties files for the supported locales. A file can omit some of the keys.
- By default, the keys are based on IDs in the Domain presentation, but you can create your own keys if you prefer.
- The values are the labels and descriptions in the language of the target locale.

The Java language specifies the format and encoding of the file, but the meaning of each key name is based on the Domain to which it is attached.

### 9.3.1    Key Names

You can use the default key names generated by JasperReports Server or you can assign your own names to the label and description keys. Key names must satisfy the following rules:

- Each key name must be unique among all key names in the Domain.
- Key names can only use characters from the ISO Latin-1 set, digits, and underscores (_).

In the Domain Designer, you can define key names using the **Label Key** and **Descriptions Key** properties on the Data Presentation tab. In the XML design file, these correspond to the `labelId` and `descriptionId` attributes on the `itemGroup` or `item` element.

If no name has been assigned to the label key or description key, JasperReports Server expects key names in a hierarchical format, for example:

```
<dataIslandID>.LABEL
<dataIslandID>.DESCR
<dataIslandID>.<setID1>...<setIDn>.LABEL
<dataIslandID>.<setID1>...<setIDn>.DESCR
<dataIslandID>.<setID1>...<setIDn>.<item>.LABEL
<dataIslandID>.<setID1>...<setIDn>.<item>.DESCR
```

## 9.3.2    Property Values

Values are strings that specify the text to use for the locale associated with the file. Values must satisfy the following rules:

- Everything after the = symbol is part of the translated text.
- The properties files use the ISO-8859-1 (Latin-1) encoding that supports most, but not all, accented characters.
- For characters that are not in ISO-8859-1, use Unicode escape sequences. For example, the œ character has the Unicode escape sequence \u0153.

You can define values in several places. JasperReports Server looks for values in the following order, and displays the first non-empty value it finds:

1. In the properties file for the current user's locale.
2. In the default properties file.
3. In the **Label** or **Descriptions** property on the Data Presentation tab in the Domain Designer. In the XML design file, these correspond to the label and description attributes on the itemGroup or item element.
4. If no value is found in any of the previous locations, then:
    - For a label, JasperReports Server uses the **ID** from the Data Presentation tab, which is the same as the value of the id property in the XML file.
    - For a description, the result is left blank.

## 9.3.3    Example of an Empty Properties File

The following example shows part of a typical properties file created from the Domain Designer, with internationalization keys generated automatically. Sets and items in the file appear in the same order as on the Data Presentation tab of the Domain Designer. If you have already entered values for the Label Key or Descriptions Key for an item, the value is filled in.

```
JOINTREE_1.ACCOUNTS.LABEL=
JOINTREE_1.ACCOUNTS.DESCR=
JOINTREE_1.ACCOUNTS.NAME.LABEL=
JOINTREE_1.ACCOUNTS.NAME.DESCR=
JOINTREE_1.ACCOUNTS.ACCOUNT_TYPE.LABEL=
JOINTREE_1.ACCOUNTS.ACCOUNT_TYPE.DESCR=
JOINTREE_1.ACCOUNTS.INDUSTRY.LABEL=
JOINTREE_1.ACCOUNTS.INDUSTRY.DESCR=
...
```

## 9.3.4    Example of a Localized File

The following example shows the French translation for the properties file fragment above, including accented characters.

> The single straight quote (') sometimes causes issues when processed in Domain properties files. To avoid this issue, use the right single quote (') by its Unicode sequence \u2019, as shown in the following example.

```
JOINTREE_1.ACCOUNTS.LABEL=Comptes
JOINTREE_1.ACCOUNTS.DESCR=Détails des comptes clients.
JOINTREE_1.ACCOUNTS.NAME.LABEL=Client
```

```
JOINTREE_1.ACCOUNTS.NAME.DESCR=Nom du client
JOINTREE_1.ACCOUNTS.ACCOUNT_TYPE.LABEL=Type
JOINTREE_1.ACCOUNTS.ACCOUNT_TYPE.DESCR=Type du compte client.
JOINTREE_1.ACCOUNTS.INDUSTRY.LABEL=Industrie
JOINTREE_1.ACCOUNTS.INDUSTRY.DESCR=Industrie d\u2019activité primaire.
...
```

## 9.4    Adding Locale Bundles to a Domain

After you have created the locale bundle, use the Locales tab in the Domain Designer to upload and manage the bundles in your Domain. A Domain can have any number of locale bundles, one for each language you want to support.

Locale bundles can be uploaded from files for use with the current Domain, or they can be stored in the repository for use with multiple Domains:

- Uploading locale bundles directly to the Domain – When you upload a bundle file on the Locales tab, the files are uploaded from your computer and attached to the current Domain. These bundles are stored with the Domain and can't be accessed or shared by other Domains.
- Sharing bundles from the repository – The repository can also store locale bundles as file resources. Locale bundles uploaded to the repository are not attached to a specific Domain. Instead, they can be referenced by multiple Domains, allowing you to reuse your translations across Domains. Care must be taken to ensure that the keys for labels and descriptions are consistent across all Domains that share a locale bundle.

If you need to modify the bundle, the Locales tab lets you download the bundle, upload it again, or delete it from the Domain.
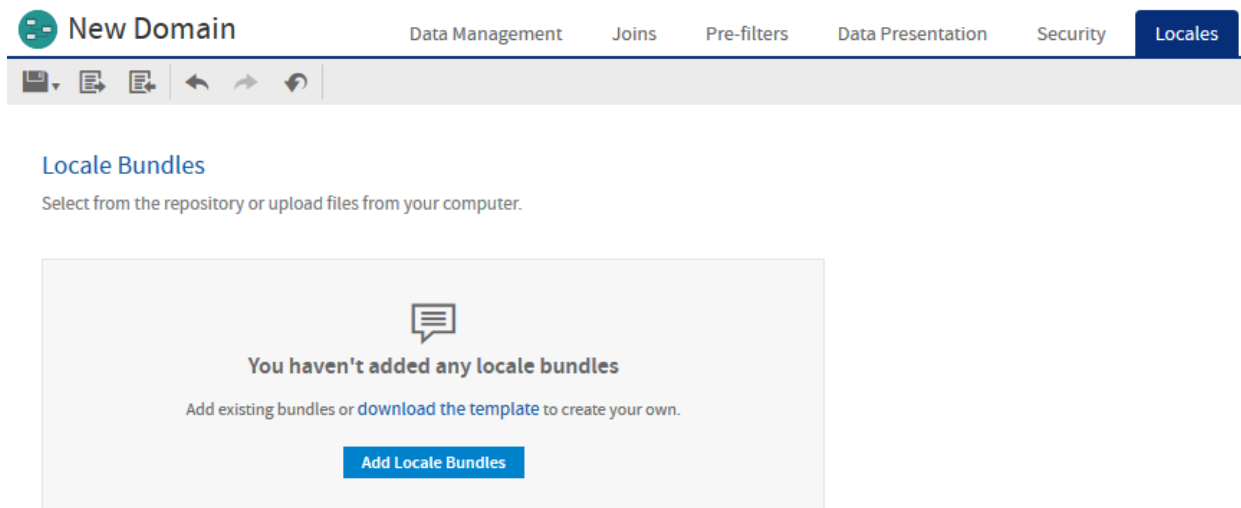


**Figure 9-3**  Locales Tab When No Resources Are Attached

### 9.4.1    Uploading Locale Bundles Directly to the Domain

Use the following procedures to add bundles directly to your Domain.

1. Edit the Domain, and on the Locales tab, click **Add Locale Bundles**.
2. On the Add Locale Bundle dialog, select the **Local File** category at the top.
3. Click **Choose Files to Upload**, and select one or more bundles from your file system.
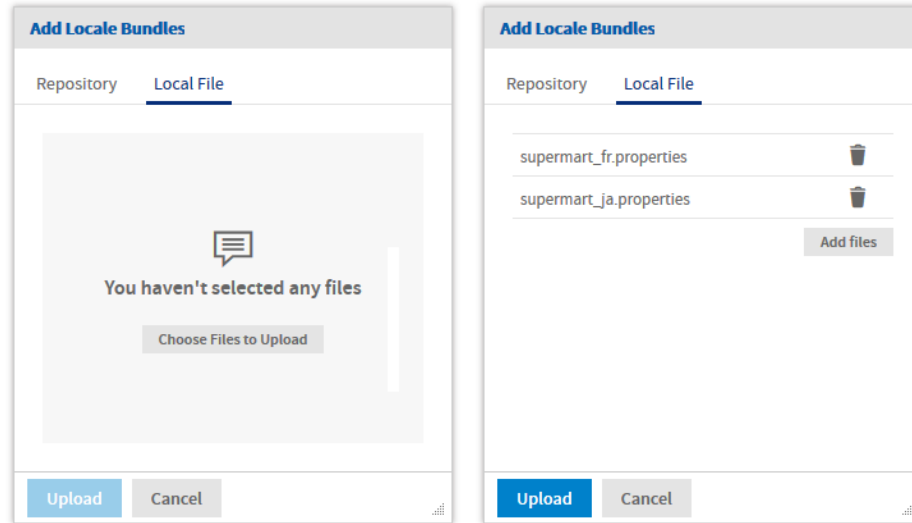


**Figure 9-4** Adding Bundles from your File System

The dialog displays a list of the files you have selected. You can add or remove items from this list to create a list of bundles to upload.

4. Click **Upload** to add these locale bundles to your Domain.
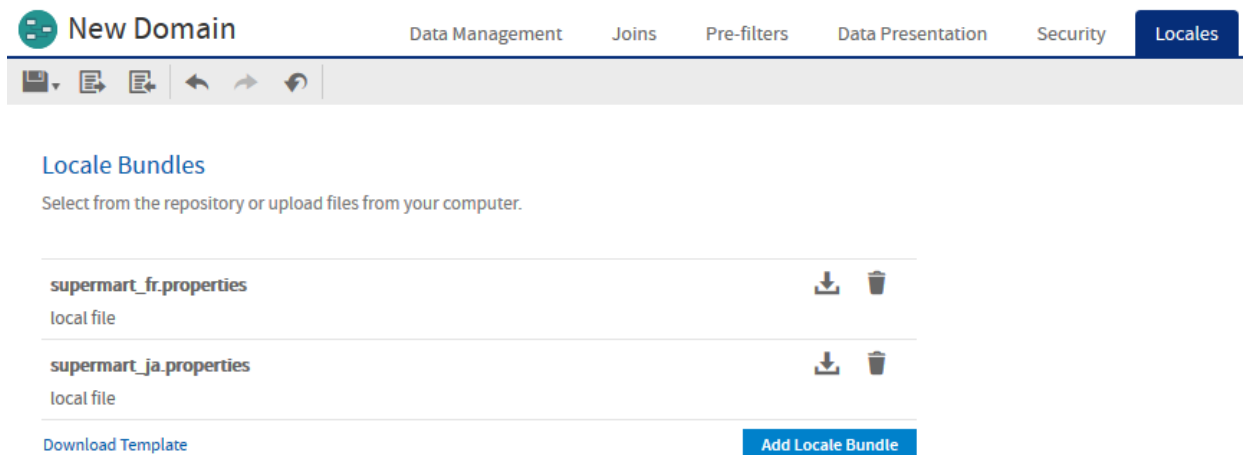


**Figure 9-5** Locales Tab with Bundles List

The server validates the file to make sure it matches the format of a locale bundle. If the file type is not recognized or there is a syntax error, you must select another file or click **Cancel**.

When validated, the bundles are added to the Domain and displayed on the Locales tab.

5.   After adding bundles to a Domain that is in use, you should clear the Ad Hoc cache of all queries based on the Domain. This removes any datasets that did not rely on the locale bundles. For instructions, see the *JasperReports Server Administrator Guide*.

Once locale bundles are associated with the Domain, they are active and used to display labels in the given languages. Whenever a user logs in with a locale corresponding to one of the bundles, they will see the labels in the bundle.

## 9.4.2   Uploading Locale Bundles from the Repository

Use the following procedures to add bundles to the repository as shared resources, then use them in your Domain.

1.   As an administrator, select **View > Repository** and navigate to the folder where you want to store locale bundles.

2.   Right-click the folder, and select **Add Resource > File > Resource Bundle**.

**Add File**

**Upload a File From Your Local Computer**

Choose the file to upload, set its properties, and specify its location.

Type:  Resource Bundle

Path to File (required):

Browse...    supermart_es.properties

Name (required):

supermart_es.properties

Resource ID (required):

supermart_es.properties

Description:

Spanish resource bundle

Save Location:

/public/Resources

Browse...

Submit    Cancel

**Figure 9-6**  **Adding a Locales Bundle to the Repository**

3.   In the Add File dialog, click **Browse** under Path to File, and select the locale bundle on your computer. The Name and Resource ID fields are automatically filled in with the file name. You must keep the `.properties` extension on the Name and ID.

4.   Give the resource a description if desired, and change the save location as needed, then click submit.

     The locale bundle is uploaded and stored in the repository.

> A warning is displayed if you attempt to delete a resource bundle file that is referenced by a Domain, but not if you replace the files with a different file. When you store these files in the repository, ensure that any updates to the files are compatible with the Domains that reference them.

5.   Edit the Domain, and on the Locales tab, click **Add Locale Bundles**.

6. On the Add Locale Bundle dialog, select the **Repository** category at the top.

The dialog displays a list of all the items of type Resource Bundle in the repository. You can mouse over a bundle to see its folder location and description, then click to select the bundles you want to use.
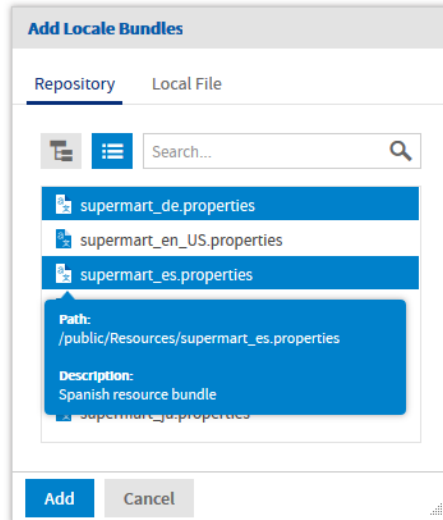


**Figure 9-7  Selecting Bundles from the Repository**

You can also click the hierarchy icon at the top and browse the folder tree of the Repository to find the bundles you want to use. Or enter a name to search the repository.

7. When you have selected all the bundles you need, click **Add**.

The server validates the resource to make sure it matches the format of a locale bundle. If the file type is not recognized or there is a syntax error, you must select another resource or click **Cancel**.

When validated, the bundles are added to the Domain and displayed on the Locales tab. You can build up a list of bundles by adding more any time. The list shows where the bundle is stored in the repository, or whether it's a local file uploaded directly to the Domain.
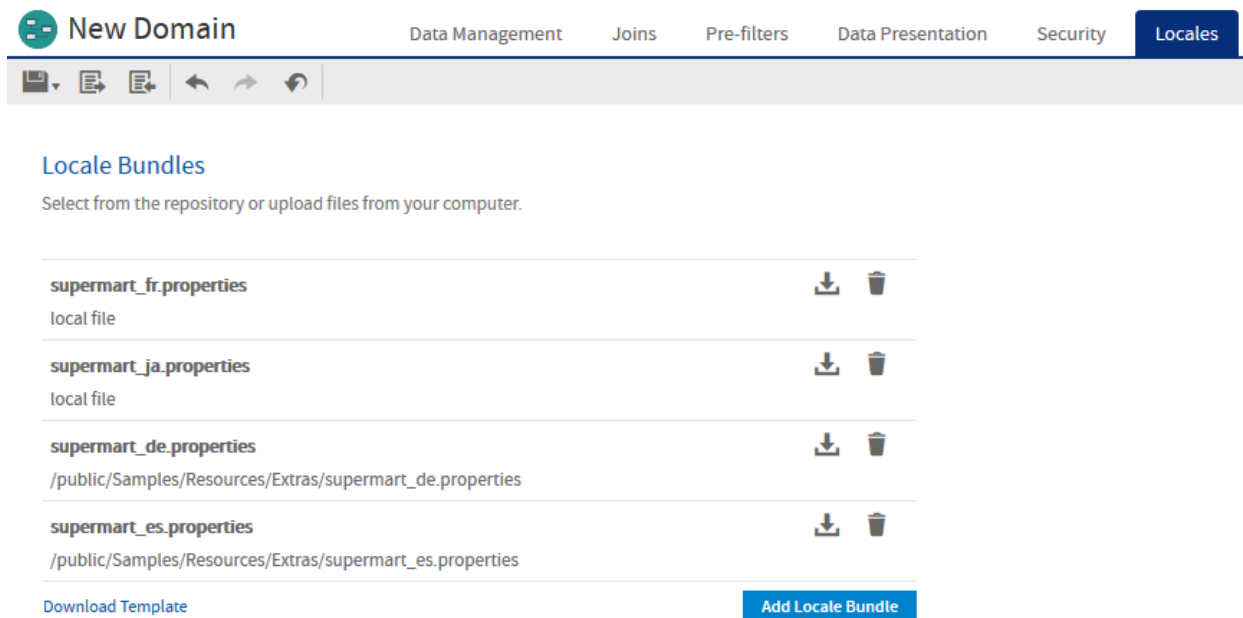
**Figure 9-8**  **Locales Tab with Local and Repository Bundles**

8.    After adding bundles to a Domain that is in use, you should clear the Ad Hoc cache of all queries based on the Domain. This removes any datasets that did not rely on the locale bundles. For instructions, see the *JasperReports Server Administrator Guide*.

## 9.4.3    Modifying amd Removing Locale Bundles

For each bundle listed on the Locales tab, there are two available actions represented by icons:

*    ⬇ – Downloads the selected resource as a .properties file.

*    🗑 – Removes the bundle from the Domain. Repository resources are not removed from the repository.

To modify a locale bundle, upload or add the new file with the same name. You are given a warning that the old bundle will be overwritten and can choose to proceed. When you agree, the old bundle with the same name is overwritten and no longer available in the Domain. If you replace a bundle added from the repository, the old bundle remains in the repository, but it is no longer referenced by this Domain. The new bundle is added to the Domain and takes effect.

There are three common cases:
*    When you modify a bundle that was uploaded as a local file, you can download it from the Locales Tab, then modify it, and upload it again. Uploading the file with the same name will replace the old version of the bundle.
*    If you need to modify a shared bundle, you can download it from the Locales tab, modify it, and replace it in the repository. This will change the bundle for all Domains that use it.
*    If you change the current Domain such that it can no longer use the shared bundle, you can download the bundle from the Locales tab, modify it, then upload it again as an unshared bundle file.

If you change the name of a locale bundle, remove the old one after uploading the new one.

The Download Template action lets you create an empty bundle when you first start localizing your Domain, as described in **9.2, "Creating Locale Bundles," on page 130**. When you already have locale bundles for your Domain, it is easier to start with an existing bundle and translate it. In that case, use its download icon as described above.

## 9.4.4    Maintaining Locale Bundles

Once you have the files you will need to update them as keys are added or changed, or as you add locales. Start from a saved properties file or the default properties file as a template. Alternatively, download one of the existing bundles and remove the translated text.

**To add or change keys:**

1.   If you change the Domain design, download the template again and compare it to your previous template to find all new keys.
2.   If you have defined descriptions and keys in the Domain design, delete the exported values to create a new blank template.
3.   Translate the new keys and add them to your properties files for your supported locales.
4.   Use the Locales tab in the Domain Designer to attach the properties files to the Domain.

**To add or change a locale:**

1.   Make another copy of the blank properties file, and save the original for
2.   Rename file to set the `_<locale>` designator and create a properties file for the new locale. For information about file naming, see **9.2.2, "Properties File Names," on page 132**.
3.   If you have defined descriptions and keys in the Domain design, delete the exported values to create a new blank template.
4.   Translate the new keys and add them to your properties files for your supported locales.
5.   Use specialized software or a translation service to create the Java localization properties files for each locale you want to support.
6.   Upload the new locale bundle to the Domain or to the repository.

For additional information about locales and properties files for the JasperReports Server user interface, see the Localization chapter in the *JasperReports Server Administrator Guide*.

# INDEX

defined 31

**J**

Jaspersoft OLAP prerequisites 7
join trees
    and data islands 35
joins 23, 113, 126
    best practices 50
    composite joins 47-48
    custom joins 48
    join trees 46
    join type 47

**L**

locale bundles 130
Localization 129
localization of Domains 130
log4j 117
logging 117

**O**

Oracle
    choosing schemas 20
    schema name in design file 65
    synonyms 19

**P**

prerequisites for Jaspersoft OLAP 7
principal expressions 119, 121
properties, in locale bundles 130

**R**

reports
    effect of access grants 11
    example 125
roles
    Domain security 114
    example 114
row-level security 118, 121, 124

**S**

schema element in design files 67
schemaLocation attribute in design files 68
schemaMap in Domain design files 70
schemas. See design files. 61

security
    Domains 111
security files
    principal expressions 119
    structure 11
sets
    creating in Domain Designer 32, 34
    defined 31
summary calculations
    in Domain XML files 99

**T**

testing Domain security 124-125
testProfileAttribute 120
Topics
    Domains and 127
troubleshooting 117

**U**

users
    attributes 115
    example 115
using the Ad Hoc Editor
    testing Domain security 117

**V**

version attribute in design files 68
virtual data source
    creating a Domain 41

**X**

xmlns attribute in design files 68