CDE dashboard overview

The Community Dashboard Editor (CDE) and its underlying technologies (CDF, CDA and CCC) enable rapid development and deployment of the Pentaho CTools dashboards. The CDE tool was created to simplify the creation, design, and rendering processes of the CTools dashboards. It is a powerful and complete tool, seamlessly integrating the user interface with data sources and custom components.
NoteSince CTools dashboards operate independently from the Pentaho Dashboard Designer, the dashboards are not compatible with one another.

The following tour of the Editor interface assumes you have activated CDE.

Parent Topic

- Products

Child Topics

- View a CDE dashboard sample

  This section highlights popular CDE capabilities by walking through the sample CDE dashboard called **Sales Breakdown (CTools Dashboard)**, located in the **Getting Started** pane.
- Open the CDE interface
- CDE **Menu** bar

  On the top-left of the window, the CDE **Menu** bar is always visible. Use this menu bar to create new dashboards, save your work, reload your current view, and define dashboard settings.
- CDE **Perspectives** toolbar

  The CDE **Perspectives** toolbar displays in the top-right of the window. Use this toolbar to switch between the Layout, Components, and Data Source perspectives, and to preview your dashboard.
- Community Dashboard Editor (CDE) perspectives

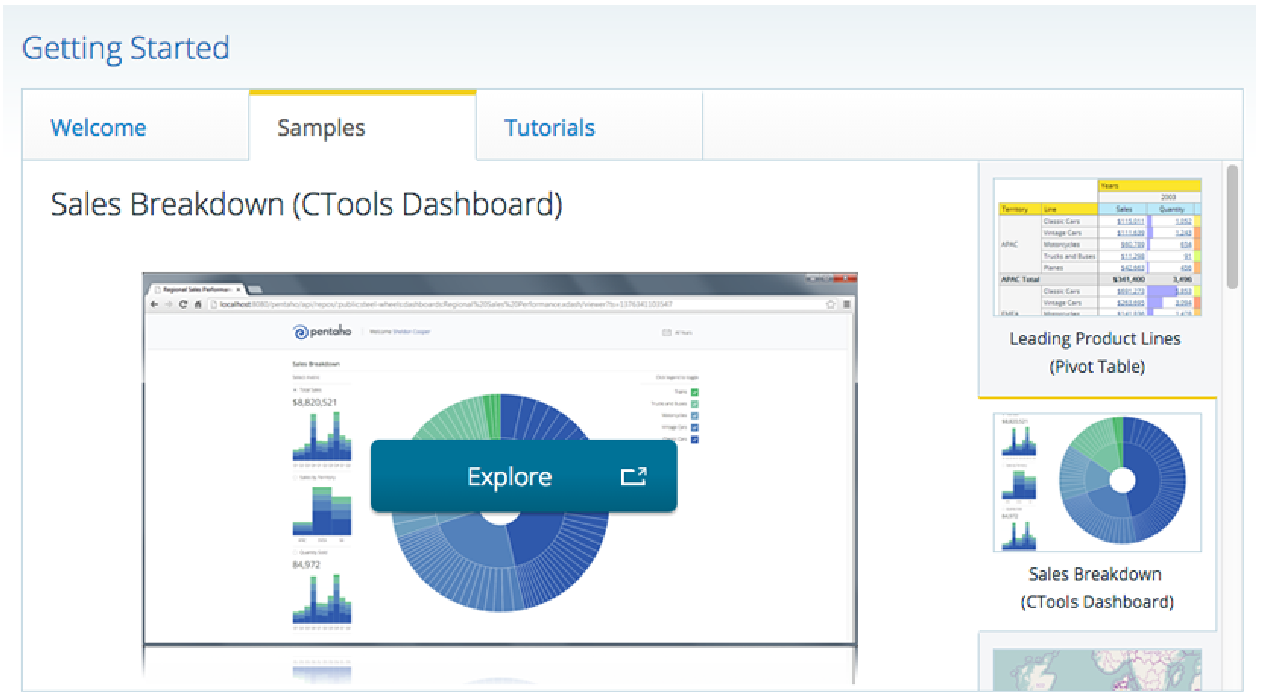
  For the design of your Dashboard, CDE offers three perspectives. Click the link to learn more about each perspective.

View a CDE dashboard sample

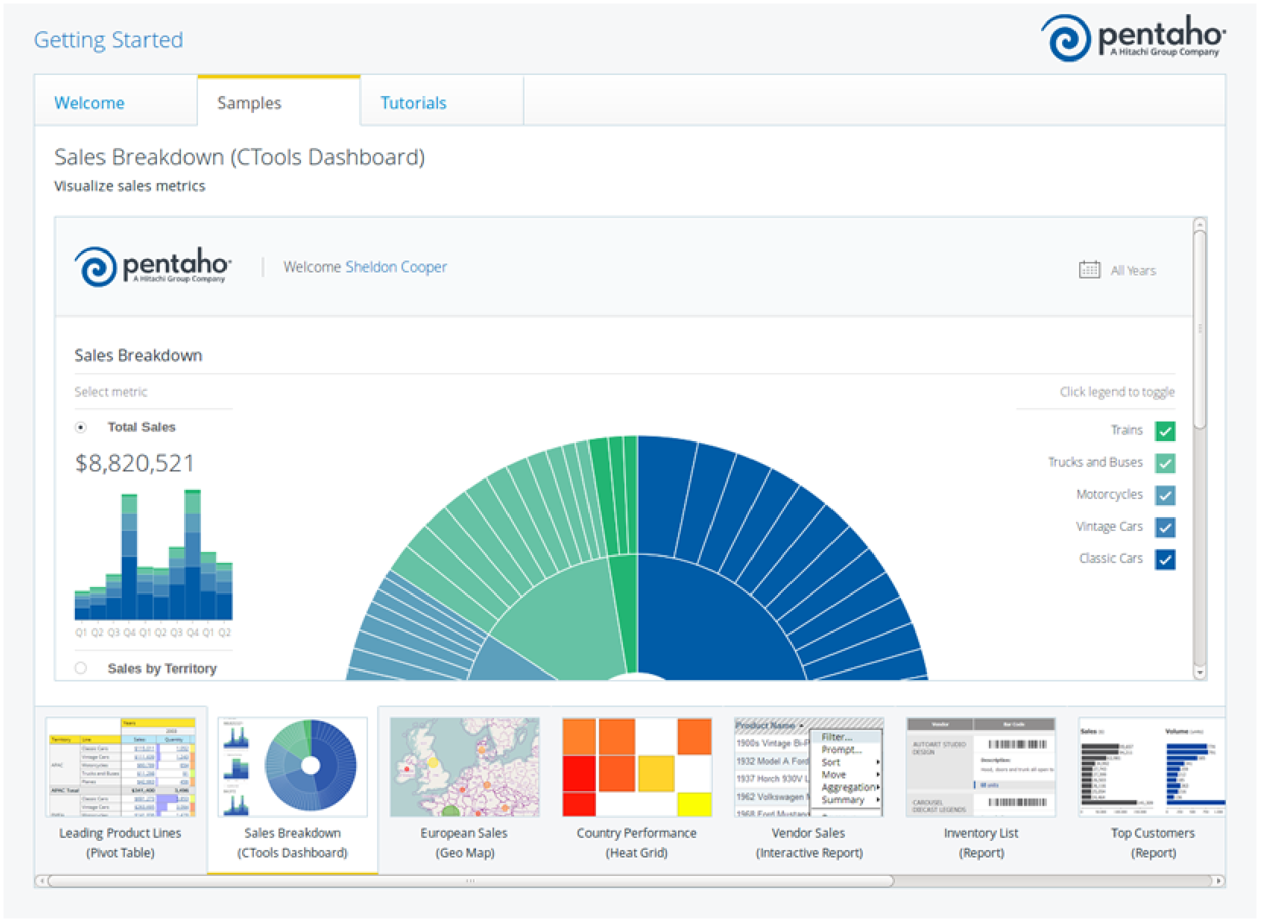
This section highlights popular CDE capabilities by walking through the sample CDE dashboard called **Sales Breakdown (CTools Dashboard)**, located in the **Getting Started** pane.

Procedure

1. Log on to the Pentaho User Console.
2. From the Home page, in the **Getting Started** widget, click the **Samples** tab.
3. From the list on the right, click to select **Sales Breakdown (CTools Dashboard)**.


Getting Started
Welcome
Samples
Tutorials
Sales Breakdown (CTools Dashboard)
Explore
Leading Product Lines (Pivot Table)
Sales Breakdown (CTools Dashboard)

4. Click **Explore** in the **Samples** tab.
   A new window opens showing the Sales Breakdown (CTools Dashboard) sample dashboard.



Results
You can view how the dashboard displays in Pentaho for users.

Parent Topic

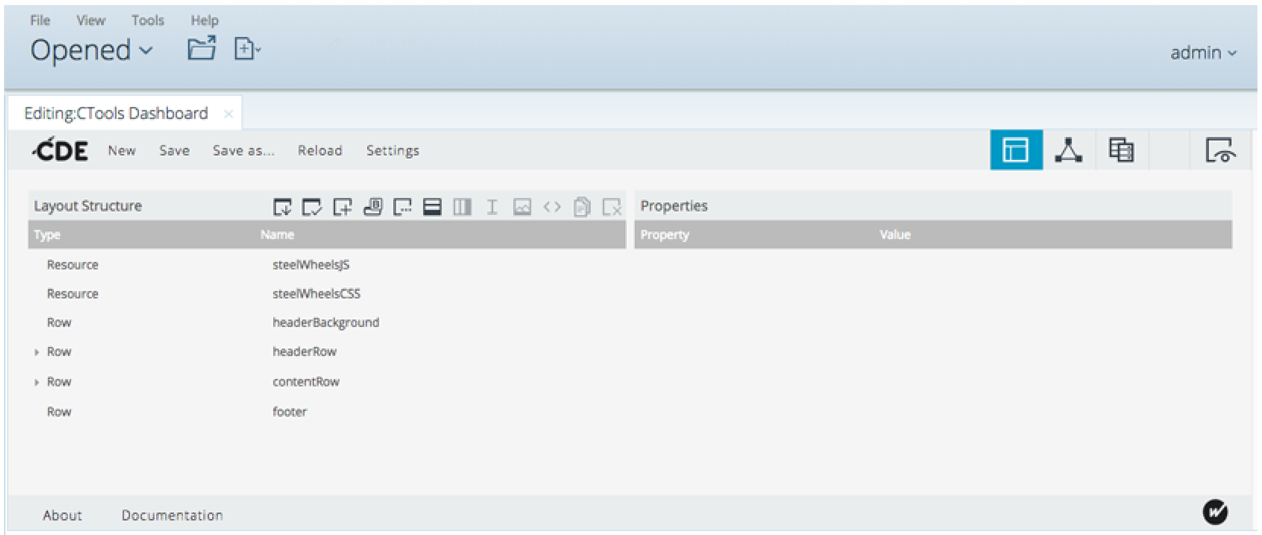
- CDE dashboard overview

Open the CDE interface

For editing, you will want to open the editable version of the Sales Breakdown (CTools Dashboard) in CDE Dashboard Editor.

Procedure

1. In the Pentaho User Console, navigate to the Browse Files perspective.
2. In the **Folders** pane, click to expand the `Public` folder, then click to highlight the `Steel Wheels` folder.
3. In the **Files** pane, click on `CTools Dashboard` and then, in the **File Actions** pane, click on **Edit**.
   You are now in the Opened perspective with the CDE in editing mode. The **Editing:CTools Dashboard** tab is now active in CDE.



Type	Name
Resource	steelWheelsJS
Resource	steelWheelsCSS
Row	headerBackground
Row	headerRow
Row	contentRow
Row	footer

Property	Value

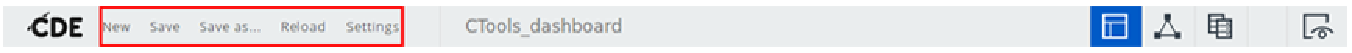
NoteYou can also open CDE in a new tab outside of Pentaho User Console by double-clicking the **Editing:CTools Dashboard** tab.

Parent Topic

- CDE dashboard overview

CDE Menu bar

On the top-left of the window, the CDE **Menu** bar is always visible. Use this menu bar to create new dashboards, save your work, reload your current view, and define dashboard settings.



Menu options include:

Menu	Description
New	Select to create a new CDE dashboard. When clicked, a blank dashboard appears.
Save	Select to save the CDE dashboard which you are currently editing. It is recommended that you continuously save your work while editing.If you have not previously saved this dashboard, the Save as dialog box displays. Choose the location to save your new dashboard and enter a file name for it. (Optional) You can enter the following: • Dashboard or Widget You can choose to save your work as a dashboard or as a widget. For now, it is recommend you keep the default option of **Dashboard**. The **Widget** option is an advanced feature which is described in an advanced topic. • Title Enter a name for your dashboard • Description Enter a small description of your dashboard's purpose and details. When finished, select **Ok** to save your dashboard or **Cancel** to close the dialog box without saving your changes.
Save as	Select to to save the current CDE dashboard in a new location and/or rename it. You can also modify the title and the description. When clicked, the Save as dialog box displays. Choose the location to save your new dashboard and enter a file name for it.In addition, you can choose to save your work as a **Dashboard** or as a **Widget**. For now, it is recommend you keep the default option of Dashboard. The Widget option is an advanced feature which is described in an advanced topic. (Optional) You can enter the following: • Title Enter a name for your dashboard • Description Enter a small description of your dashboard's purpose and details.

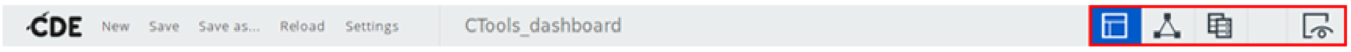
Menu	Description
	When finished, select **Ok** to save your dashboard or **Cancel** to close the dialog box without saving your changes.
Reload	Select to refresh the CDE interface to the last saved state. This is useful when you make changes which are not immediately reflected after saving, or when you want to discard your most recent changes and return to the last saved state.
Settings	Select to define settings for your dashboard, such as metadata information, HTML templates, and dashboard type. You can do the following: • Add or modify the **Title** and **Description** for your dashboard • Set the **Author** for your dashboard. • Select the **Style** (HTML template) which you want to apply to the dashboard. Clean is selected by default. • Select the **Dashboard Type**. Bootstrap is selected by default. • Click the **RequireJS Support** check box. By default, this check box is cleared. The Style and Dashboard Type settings are described in an advanced topic.

Parent Topic

- CDE dashboard overview

CDE Perspectives toolbar

The CDE **Perspectives** toolbar displays in the top-right of the window. Use this toolbar to switch between the Layout, Components, and Data Source perspectives, and to preview your dashboard.



Toolbar options include:

Icon	Name	Function
	Layout	Select to view the Layout Perspective in the editor. Here you can define the layout of your dashboard.

Icon	Name	Function
△	Components	Select to view the Components Perspective in the editor. Here you can add components to your dashboard.
▤	Data Sources	Select to view the Data Sources Perspective in the editor. Here you can add data sources to your dashboard.
👁	Preview	Select to test the look and feel as well as the behavior of your dashboard as you are working. When selected, your dashboard opens in the Preview window.

Parent Topic

- CDE dashboard overview

Community Dashboard Editor (CDE) perspectives

For the design of your Dashboard, CDE offers three perspectives. Click the link to learn more about each perspective.

Parent Topic

- CDE dashboard overview

Child Topics

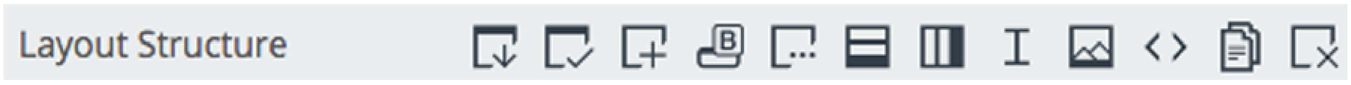
- Layout Perspective
- Dimensions
- Look and feel
- Components perspective
- Data Sources perspective

  In the Data Sources perspective you can find various types of data sources which you can employ in a dashboard. These allow you to access the data that you want to use in your dashboard.
- Final note

Layout Perspective

In the Layout perspective, you can design the layout of your dashboard from scratch or by using a CDE template. While defining the layout you can apply styles and add HTML elements as text or images.

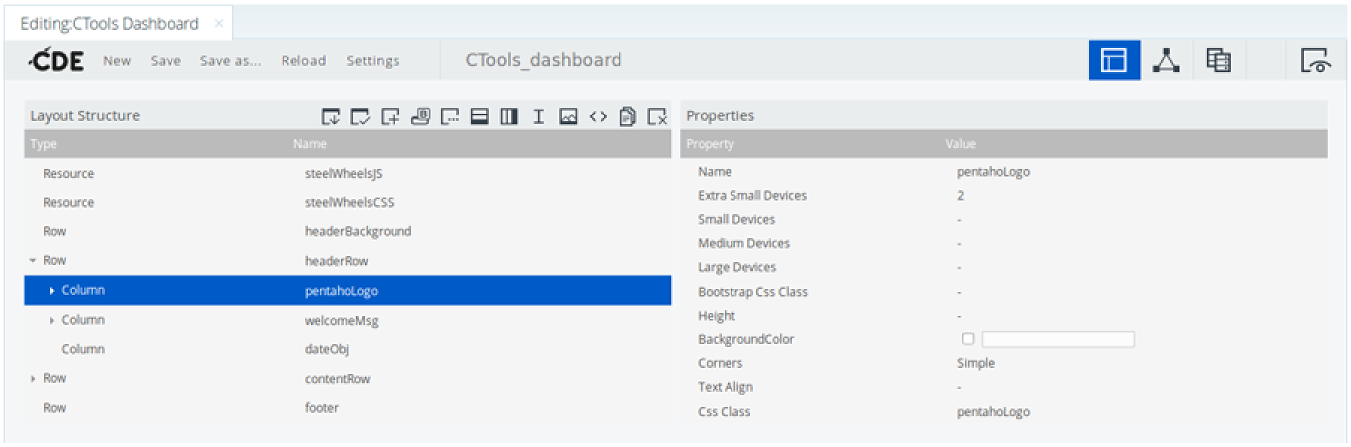
Use the **Layout Structure** toolbar in the top left of the window to add and delete rows, columns, HTML blocks, and images in your dashboard. To move elements around, drag and drop them within the Layout perspective. You can also add resource files (JS or CSS) or snippets.



Icon	Icon Name	Description
	Save as Template	Select to save the dashboard as a template which can be applied to other dashboards. The generated template can contain just the layout structure or may include components and data sources.
	Apply Template	Select to apply a template to a dashboard. You can select from a list of pre-defined templates, or use a template you created.
	Add Resource	Select to add CSS or Javascript resources to the dashboard. These can be snippets, which will be included in the dashboard's HTML as inline code, or external files, which are loaded with the dashboard's HTML.
	Add Bootstrap Panel	Select to add a Bootstrap panel to the dashboard layout. The Bootstrap panel contains a panel header, panel body and panel footer.
	Add FreeForm	Select to add an HTML free-form component to the dashboard. This is a user-defined HTML tag with user-defined CSS classes and user-defined attributes. For example, you can have a tag such as the following: `<div id="elementName" class="customClass" title="customTitle"></div>` This element can then receive other elements, providing limitless custom nesting potential, and you can view it easily in the layout structure, which is not always possible with an HTML element.

Icon	Icon Name	Description
⊟	**Add Row**	Select to add a row to the dashboard.
◫	**Add Column**	Select to add a column to the dashboard.
I	**Add Space**	Select to add a separator to the dashboard. Optimally used between rows to add a space.
🖼	**Add Image**	Select to add an image to the dashboard. Note that images are typically added via CSS instead of through the Add Image button.
< >	**Add Html**	Select to add inline HTML code to the dashboard, such as text. Note that an HTML element can only be added to a column, not a row.
⧉	**Duplicate Layout Element**	Select to duplicate a dashboard section as laid out in the Editor
☒	**Delete**	Select to delete a dashboard section as laid out in the Editor.

As you build the layout you will see the structure of nested rows and columns below the left toolbar in the **Layout Structure** panel. As you select one of the elements on the layout structure, the **Properties** panel on the right is updated. Here you can configure the element to your specifications.



Editing:CTools Dashboard

CDE New Save Save as... Reload Settings CTools_dashboard

Layout Structure

Type	Name
Resource	steelWheelsJS
Resource	steelWheelsCSS
Row	headerBackground
Row	headerRow
Column	pentahoLogo
Column	welcomeMsg
Column	dateObj
Row	contentRow
Row	footer

Properties

Property	Value
Name	pentahoLogo
Extra Small Devices	2
Small Devices	-
Medium Devices	-
Large Devices	-
Bootstrap Css Class	-
Height	-
BackgroundColor	
Corners	Simple
Text Align	-
Css Class	pentahoLogo

Parent Topic

- [Community Dashboard Editor (CDE) perspectives]

  For the design of your Dashboard, CDE offers three perspectives. Click the link to learn more about each perspective.

Dimensions

Due to the inclusion of Bootstrap libraries, the configuration of the columns in the layout is simple. The columns in a row must occupy 12 spans, such that in a single row you could have the following sample configurations:

- Twelve columns of size 1 (12x1)
- Two columns of size 6 (2x6)
- Three columns of size 4 (3x4)
- One column of size 8 and one column of size 4 (8+4)

Whatever your configuration, the spans must add up to 12 for Bootstrap. Other CSS libraries may have different rules. For instance, in the case of the Blueprint library, the total number of spans in a column is 24.

You can assign the width of a column across multiple devices where you will draw the components through the properties:

Category	Suggested Device	Width (in pixels)
Extra Small Devices	Phones	<768
Small Devices	Tablets	768-992
Medium Devices	Desktops	992-1200
Large Devices	Desktops	>1200

You only need to assign values to one of these types of devices. If you do not need to have a responsive dashboard, you can set the values only for the 'Extra Small Devices', for example. That way, all the other

devices will inherit the layout that you assign for that category. However, if you need a responsive dashboard which will fit well in a mobile phone and also in a desktop, you can specify a different layout for each device, assigning different values for the 'Extra Small Devices' and the 'Medium Devices' categories. For more information, visit http://getbootstrap.com/css/#grid.
NoteWhile these properties are specified in Bootstrap units, the height for the rows has to be supplied in pixels.

Parent Topic

- [Community Dashboard Editor (CDE) perspectives](#)

  For the design of your Dashboard, CDE offers three perspectives. Click the link to learn more about each perspective.

Look and feel

In the Layout perspective you may also provide some look and feel properties. For example, you can configure a background color for an element or a style for rows and columns. If you add a CSS resource, you can apply any of the styles defined in the CSS file to any of the elements in your layout by typing it in the Css Class and/or Bootstrap Css Class property.

CDE provides several simple properties which can be used to customize each specific component. There are some extra properties which you can set, such as background color or style for the corners of rows. Those properties will take precedence over any existing CSS rule which you may have included in the dashboard.

However, this type of direct customization of an element is not recommended for big production systems, as it requires specific manual reconfiguration of each property in the event of a change to the look and feel. For that reason, it is recommended that you use an external CSS resource file.

In the associated CSS style sheet, create a new class and define its properties. Then add this class in the desired element's **CSS Class** property field in CDE.

In summary, you an adjust the look and feel of you dashboard in the following ways:

- HTML to create dashboard elements.
- CSS to control the style and layout.
- JavaScript to add interactivity.
- jQuery to simplify all those tasks.

Parent Topic

- [Community Dashboard Editor (CDE) perspectives](#)

  For the design of your Dashboard, CDE offers three perspectives. Click the link to learn more about each perspective.

Components perspective

In the Components perspective, you add and set up the different components that make up your dashboard. These components are the central elements of a dashboard. They link the layout elements with the data sources.

There are three types of components:

- Visual and Data Components

  Components are displayed in your dashboard, including text boxes, tables, charts (such as pie, bar, and line), selectors (such as radio buttons and date pickers), OLAP views, and reports.

- Parameters

  Parameters represent values which are shared by the components. These are essential for the various types of component interaction.

- Scripts

  Pieces of JavaScript code which let you customize the look and feel or behavior of other components.

Parent Topic

- Community Dashboard Editor (CDE) perspectives

  For the design of your Dashboard, CDE offers three perspectives. Click the link to learn more about each perspective.

Child Topics

- Components overview
- Component lifecycle
- Adding and setting up components
- List of components

  This is the list of all available components, grouped as we can find them on the left pane.

Components overview

Components simplify the building of dashboard objects. A component is a simple JavaScript object which encapsulates all the object properties and behaviors, allowing for a finer degree control of the dashboard's components. For example, a component can adjust its behavior when reacting to changes in dashboard parameters that affect it. These adjustments in behavior can be defined before, during, and after the component's execution, which allows components to interact with each other. Customizing components allows you to personalize the look and feel of the whole dashboard.

Parent Topic

- Components perspective

Component lifecycle

Generally speaking, some components will react to events generated by the user, such as selector type components; others will react to changes in parameters, such as chart components; some will react to both types of changes, such as table components. A component’s response to such changes prompts the component to enter its lifecycle, where its execution and behavior is controlled via a set of properties which are common to most of the components.

You can set most of the following lifecycle properties on the **Properties** panel in the Components perspective. Note that you will want to click **Advanced Properties** to access all the available properties for a component.

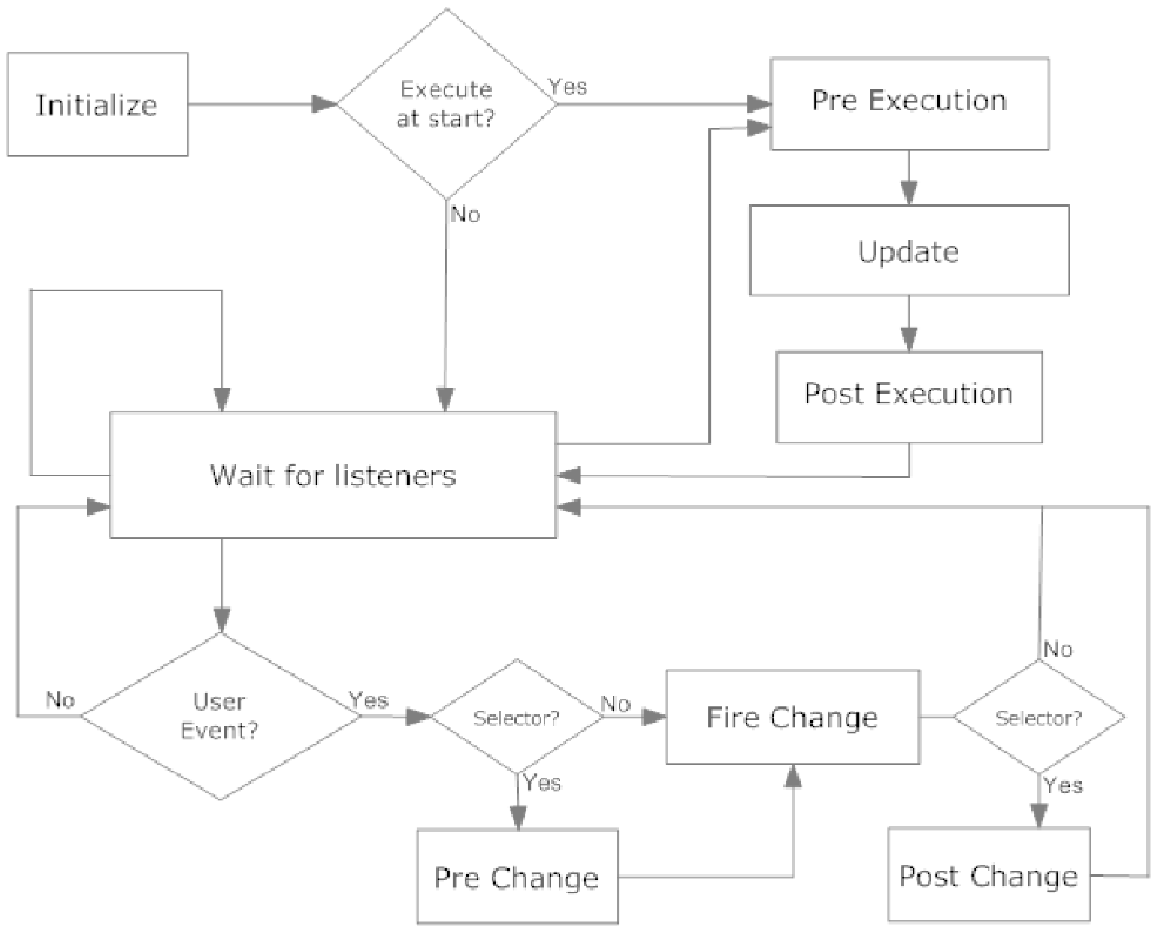
Property	Description
Type	This property assumes a variety of values such as `ComponentsParameter`, `ComponentsSelect`, and `ComponentscccBarChart`. You cannot edit this property through the CDE interface. It is set on the backend.
Name	This property is the identifier of the component. It is recommended you use camel case when entering a name for a component.
Parameter	For components such as the Select component, the Filter Component and others where user input is required, this field is where the input is stored for later use throughout the dashboard.
Listeners	This property is composed of the dashboard parameters which may trigger a component's reaction. These parameters allow for interaction between components, allowing you to control when some components will execute and in which order. For example, you may have a component which will only be executed after a change to a parameter which the component is listening to. A component can have more than one parameter in the **Listeners** property, and a parameter can be listened by more than one component. Every time the parameter changes, all the components which listen to it are updated in the dashboard.
Parameters	On the case of xAction components, PRPT components, Query components, and others, some parameters may be passed by specifying the desired value in an array of arrays.
HtmlObject	This is the ID of the HTML object on which the component will be appended to. This ID corresponds to the name you attributed to the HTML column element onto which you want to place the component.
Priority	Use this property to control the order in which the dashboard elements are executed. By default a component's priority is set to 5. The lower the number, the higher the priority the component has. Note that components with the same priority value may not be executed at the same time, so if order of execution is critical, assign priority.
Execute at Start	This property controls whether or not the component will execute when the dashboard loads itself. By default, this property is set to True. In the cases where this property is set to `False`, the component will only be executed when a change occurs in one of the parameters which the component is listening to.

Property	Description
Pre Execution/Post Execution	These functions are executed before/after the component is initialized, updated, or presented to the user. If the preExecution returns `false`, the component is not updated.
Pre Change/Post Change	These properties are for component selectors. Before/ after the input value is updated, these functions are executed. This property is useful for validating user input.
Post Fetch	This function is involved if the update stage calls queries. Once the query is executed, the data returned is passed to the postFetch function. The component is rendered only after the execution of this function.

To sum up, a component is initialized when a dashboard first loads. The component will check the value of the **Execute At Start** property, and if this value is set to `True`, the component will enter the pre-execution function and execute any code which is set here. For instance, in this phase the component can set itself to use a specific query, or fetch the contents of a parameter for its own use. The component will then update itself, entering the post-execution function. After completing this cycle, the component will wait for changes to the parameters to which it is listening to. A change to one of these parameters will cause the component to pre-execute, update, and post-execute.

When a parameter is changed, the component evaluates whether the change was triggered by a user event. If this is the case and the change has not originated in a selector, a fire change occurs. The information that a parameter has been changed is propagated to the dashboard and any components listening to that parameter will execute themselves.

If the parameter change occurred via a selector, the component will execute any pre-change instructions, as well as any post-change instructions, after the fire change event. Following this occurrence, the component will remain in wait of any changes to the parameters it is listening to. This lifecycle is further illustrated in the following image:



Parent Topic

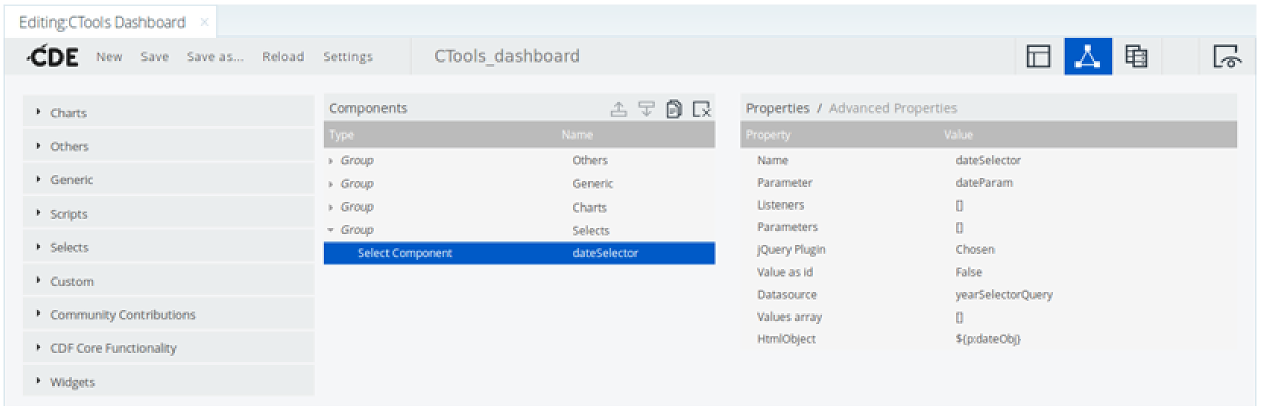
- Components perspective

Adding and setting up components

You can add components to your dashboard and customize them.

Procedure

1. From the Components perspective, in the left menu, expand the category where your component belongs.
   For example, if you want to add a date input component, expand the **Selects** category.
2. Select the component you want to add, such as **Date input Component**.
   The new component will be added to the **Components** pane in the middle of the screen.
3. Select the component in the **Components** pane to customize it in the **Properties** pane on the right.



Type	Name
Group	Others
Group	Generic
Group	Charts
Group	Selects
Select Component	dateSelector

Property	Value
Name	dateSelector
Parameter	dateParam
Listeners	[]
Parameters	[]
jQuery Plugin	Chosen
Value as id	False
Datasource	yearSelectorQuery
Values array	[]
HtmlObject	${p:dateObj}

For each component, there is a list of basic and advanced properties which you can set to define the component.

Parent Topic

- Components perspective

List of components

This is the list of all available components, grouped as we can find them on the left pane.
Parameters

- Simple Parameter
- Custom Parameter
- Date Parameter

Selects

- Filter Component
- Date Range Input Component
- Date Input Component
- Autocomplete Component
- Radiobutton Component
- Check Component
- Select Component
- Multiple Select Component
- Simple Autocomplete Component

Standard

- Table Component
- Query Component
- Button Component
- MultiButton Component
- Popup Component
- Export Popup Component
- Export Button Component
- Text Component

- Text Input Component
- Dashboard Component
- Map Component
- Template Component
- Freeform Component
- Textarea Input Component
- Visualization API Component

Charts

- Protovis Component
- CCC Area Chart
- CCC Bar Chart
- CCC Boxplot Chart
- CCC Bullet Chart
- CCC Dot Chart
- CCC Heat Grid
- CCC Line Chart
- CCC Metric Dot Chart
- CCC Metric Line Chart
- CCC 100% Stacked Bar Chart
- CCC Pie Chart
- CCC Stacked Area Chart
- CCC Stacked Dot Chart
- CCC Stacked Line Chart
- CCC Sunburst Chart
- CCC Treemap Chart
- CCC Waterfall Chart
- Dial Chart Component
- JFreeChart Component
- OpenFlashChart Component
- Timeplot Chart
- CGG Component
- CGG Dial Component

Others

- Analyzer Component
- Execute Analyzer Component
- PRPT Component
- Execute PRPT Component
- Schedule PRPT Component
- XAction Component
- Execute XAction Component

- Text Editor Component
- New Selector Component

Legacy

- AJAX Request Component
- Comments Component
- Current Version Component (Non-RequireJS dashboard only)
- Duplicate Component
- Month Picker Component
- Navigation Menu Component
- OLAP Selector Component
- Pivot Component
- Pivot Link Component
- Raphael Component
- Related Content Component
- Site Map Component
- Traffic Component
- Version Check Component (Non-RequireJS dashboard only)
- Widget Component (Non-RequireJS dashboard only)
- Widget Sample
- Mobile Navigation Component (Non-RequireJS dashboard only)

Scripts

- Function

Community Contributions

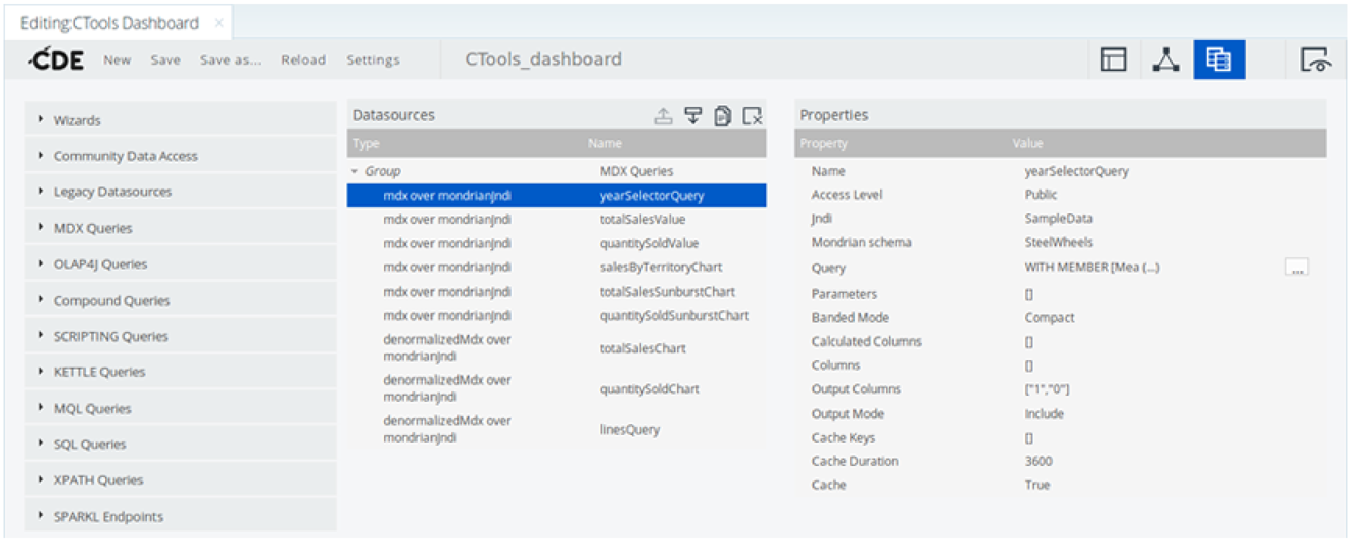
- Google Maps Overlay Component
- Google Analytics Component

Parent Topic

- Components perspective

Data Sources perspective

In the Data Sources perspective you can find various types of data sources which you can employ in a dashboard. These allow you to access the data that you want to use in your dashboard.


Editing:CTools Dashboard

CDE New Save Save as... Reload Settings CTools_dashboard

- Wizards
- Community Data Access
- Legacy Datasources
- MDX Queries
- OLAP4J Queries
- Compound Queries
- SCRIPTING Queries
- KETTLE Queries
- MQL Queries
- SQL Queries
- XPATH Queries
- SPARKL Endpoints

Datasources

Type	Name
Group	MDX Queries
mdx over mondrianJndi	yearSelectorQuery
mdx over mondrianJndi	totalSalesValue
mdx over mondrianJndi	quantitySoldValue
mdx over mondrianJndi	salesByTerritoryChart
mdx over mondrianJndi	totalSalesSunburstChart
mdx over mondrianJndi	quantitySoldSunburstChart
denormalizedMdx over mondrianJndi	totalSalesChart
denormalizedMdx over mondrianJndi	quantitySoldChart
denormalizedMdx over mondrianJndi	linesQuery

Properties

Property	Value
Name	yearSelectorQuery
Access Level	Public
Jndi	SampleData
Mondrian schema	SteelWheels
Query	WITH MEMBER [Mea (...)
Parameters	[]
Banded Mode	Compact
Calculated Columns	[]
Columns	[]
Output Columns	["1","0"]
Output Mode	Include
Cache Keys	[]
Cache Duration	3600
Cache	True

The data source perspective is divided into three sections. The left pane lists the supported data source types, grouped by category. Once selected, the data sources are grouped by category in the middle **Datasources** pane. The **Properties** pane on the right displays the properties you'll need to set for each data source, such as connection information, the actual query, parameters, column configuration, and output columns.

Parent Topic

- Community Dashboard Editor (CDE) perspectives

  For the design of your Dashboard, CDE offers three perspectives. Click the link to learn more about each perspective.

Child Topics

- List of data sources
- Common properties
- Queries with parameters

List of data sources

This is the list of all the available data sources, grouped in the left pane:

- **Wizards**

  A setup assistant to guide you through the steps of creating an OLAP selector or chart.

- **Community Data Access (CDA)**

  CDA allows data to be retrieved from multiple data sources and combined in a single output which can easily be passed on to dashboard components.

- **Legacy Datasources**

  Legacy data sources include PDI/Kettle transformations, OLAP MDX queries, SQL queries, and Xaction result sets.

- **Pentaho App Builder Endpoints**

The PAB's internal Kettle transformations and jobs.

- **MDX Queries**

  You can retrieve data from a Mondrian cube via an MDX query.

- **OLAP4J Queries**

  These data sources execute queries using the olap4j specification.

- **Compound Queries**

  These queries allows you to combine the result of two distinct queries. Compound queries can be either JOIN or UNION.

- **SCRIPTING Queries**

  Create ad hoc result sets for prototyping purposes using Beanshell scripts.

- **KETTLE Queries**

  Define a Kettle transformation file to fetch data.

- **MQL Queries**

  Pentaho Metadata defines a business model and query implementation so business users can query data sources using Pentaho reporting tools.

- **SQL Queries**

  Use this type of data source to access data from SQL databases if you have a JNDI connection or a JDBC driver setup.

- **XPATH Queries**

  Provides the ability to read data from any type of XML file using XPath specifications.

Parent Topic

- Data Sources perspective

  In the Data Sources perspective you can find various types of data sources which you can employ in a dashboard. These allow you to access the data that you want to use in your dashboard.

Child Topics

- **Wizards**
- **Community Data Access**
- **Legacy Datasources**
- The Pentaho App Builder endpoints
- **MDX Queries**
- **OLAP4J Queries**
- **Compound Queries**
- **SCRIPTING Queries**
- **KETTLE Queries**

- [MQL Queries]
- [SQL Queries]
- [XPATH Queries]

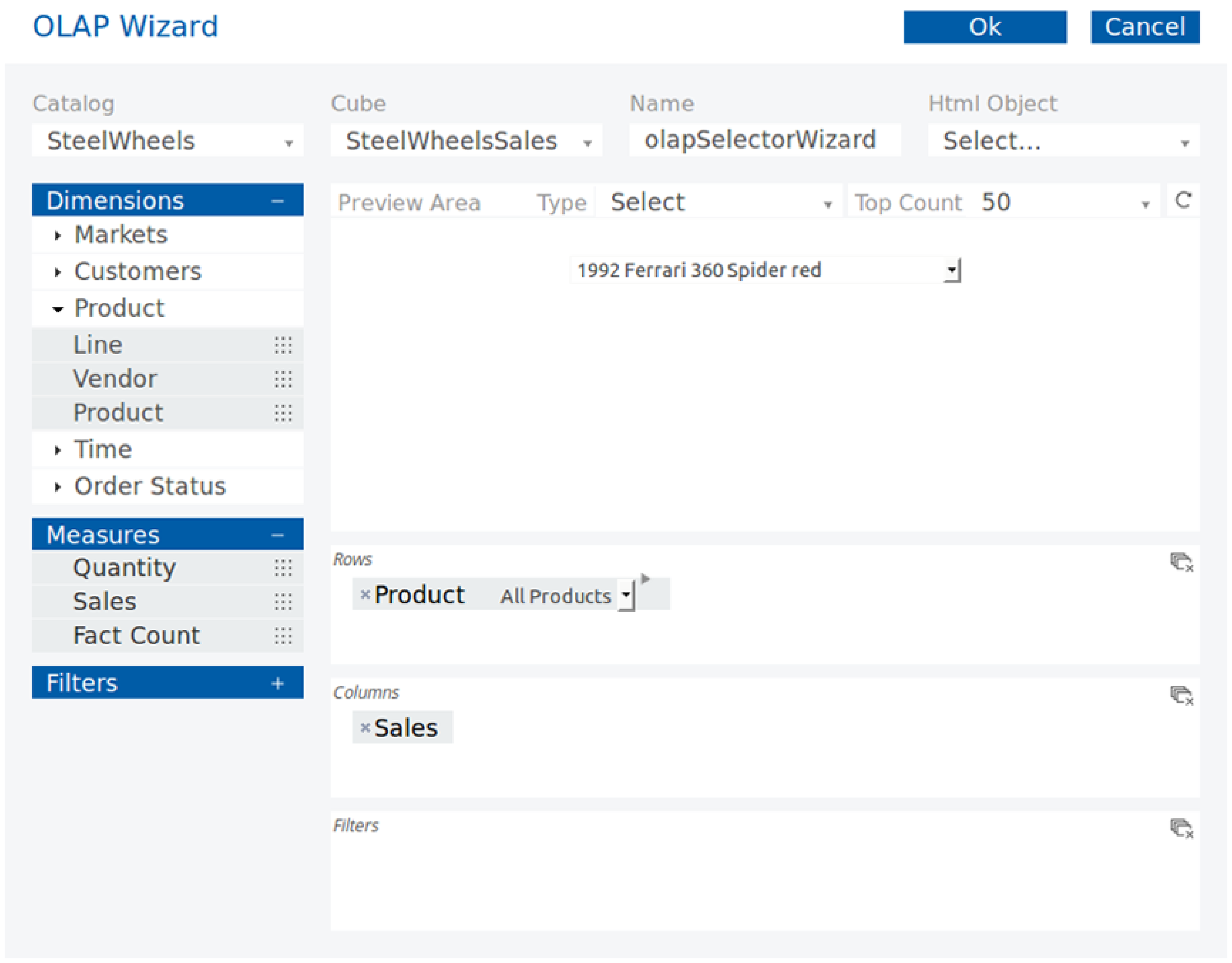
  This data source provides the ability to read data from any type of XML file using XPath specifications.

Wizards

These wizards can be used to create either a selector or a chart by setting a few properties. You can use the following types of wizards:

- OLAP Selector wizard
- OLAP Chart wizard
- Saiku OLAP Wizard

Using the configuration pane, you can select an MDX Cube, and from that cube you can select measures and metrics to generate a result set which you want to display in a selector or chart. The configuration pane features a preview area where you can view how your chart or selector will work.


OLAP Wizard

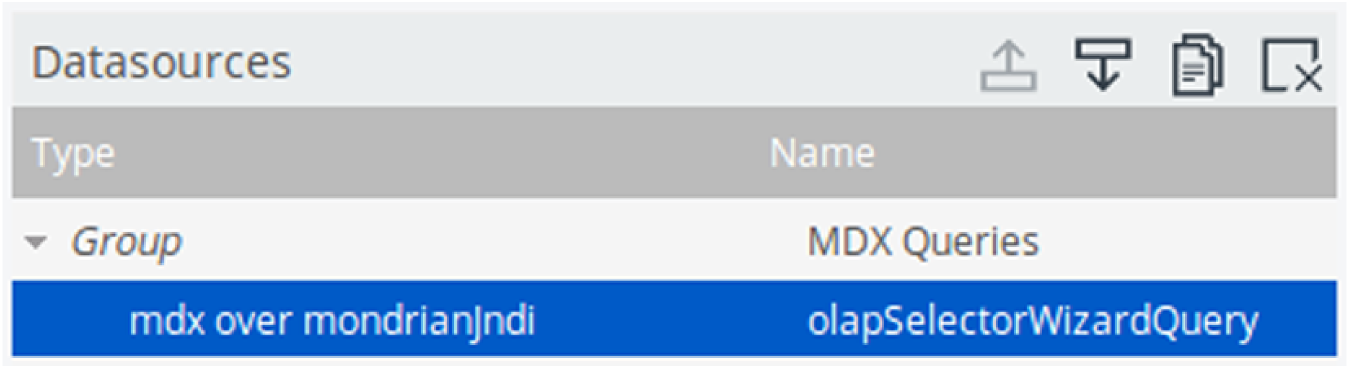
Catalog: SteelWheels
Cube: SteelWheelsSales
Name: olapSelectorWizard
Html Object: Select...

Preview Area — Type: Select — Top Count: 50

1992 Ferrari 360 Spider red

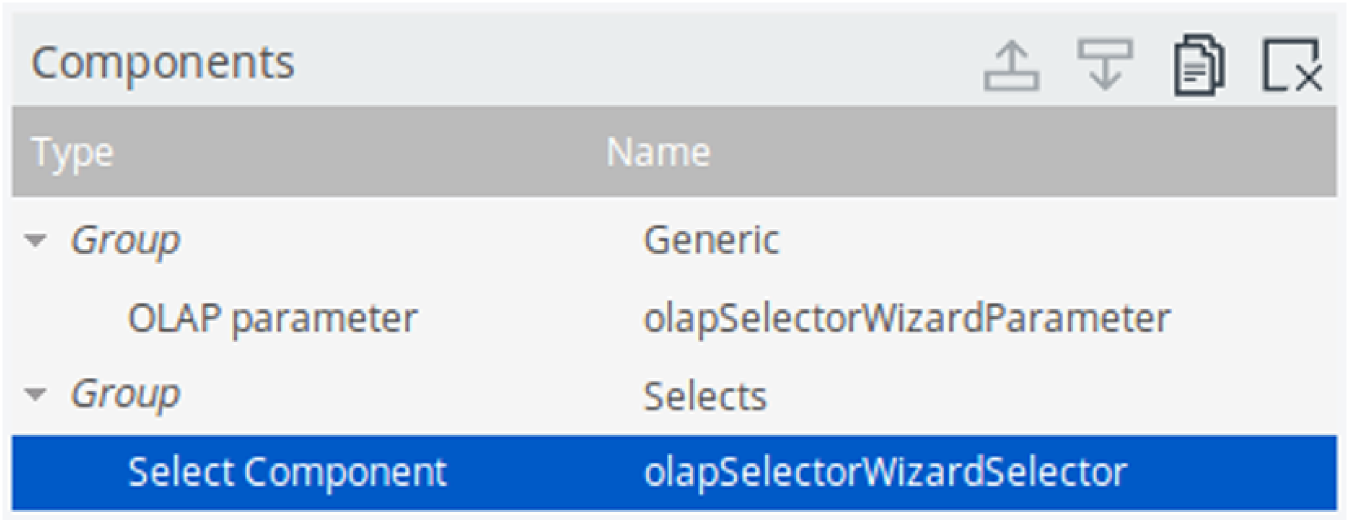
Rows: Product — All Products
Columns: Sales
Filters:

There are specific settings which can be set for either the selector or the chart wizard. The selector wizard allows you pick from select, radio box, or multiple selector options whereas the chart wizard provides selection options for bar, pie, line, and dot charts. After setting your options, clicking **Ok** adds a `mdx over mondrianJndi` data source to the **Datasources** pane.



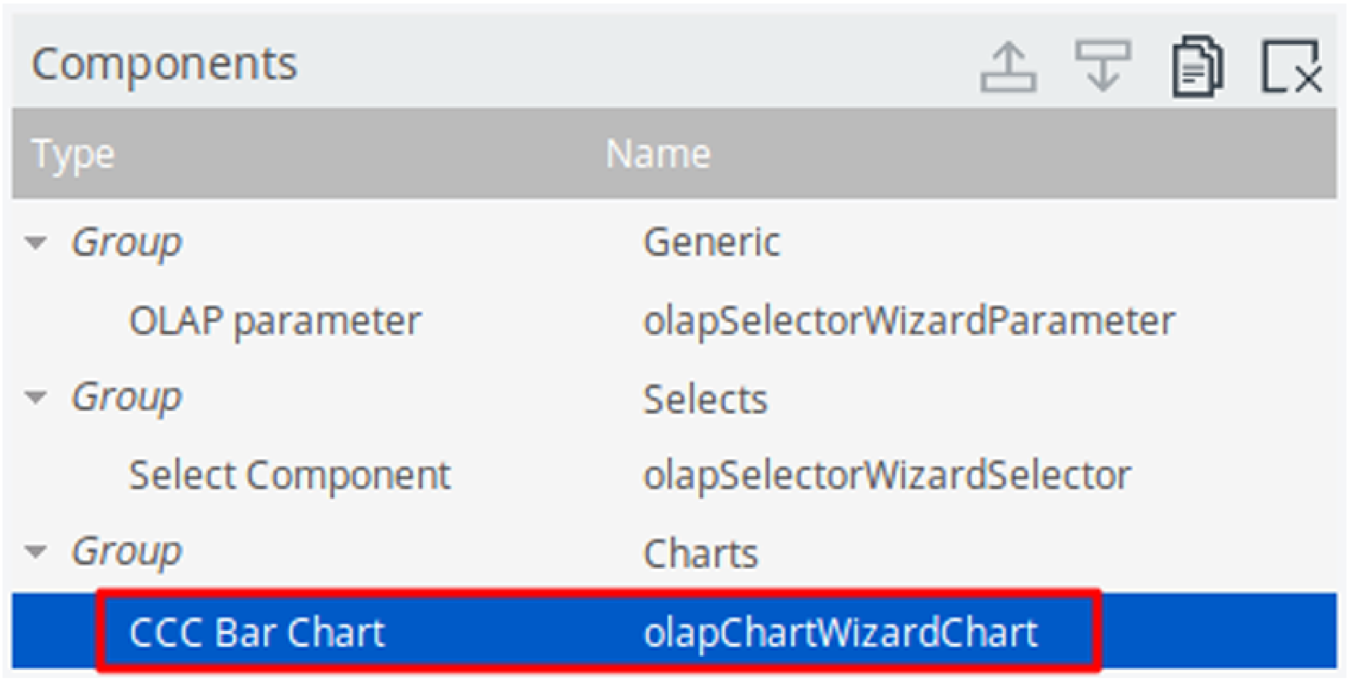
Type	Name
Group	MDX Queries
mdx over mondrianJndi	olapSelectorWizardQuery

The wizard creates this data source, setting all the necessary parameters as well as the query for the data source to properly execute. The selector wizard also creates a parameter and a select component from the selections we made in the wizard, in the **Components** pane on the Components perspective.



Type	Name
Group	Generic
OLAP parameter	olapSelectorWizardParameter
Group	Selects
Select Component	olapSelectorWizardSelector

When creating a chart using the OLAP Chart wizard, a chart component is generated rather than the OLAP parameter and select component.



Type	Name
Group	Generic
OLAP parameter	olapSelectorWizardParameter
Group	Selects
Select Component	olapSelectorWizardSelector
Group	Charts
CCC Bar Chart	olapChartWizardChart

Parent Topic

- List of data sources

Community Data Access

CDA allows you to access any of the many Pentaho data sources as well as allowing you to join different data sources just by editing an XML file, caching queries to boost performance, or delivering data in different file formats, such as CSV and XLS, through the Pentaho User Console. These tasks can be accomplished by selecting a CDA data source in this category.

Parent Topic

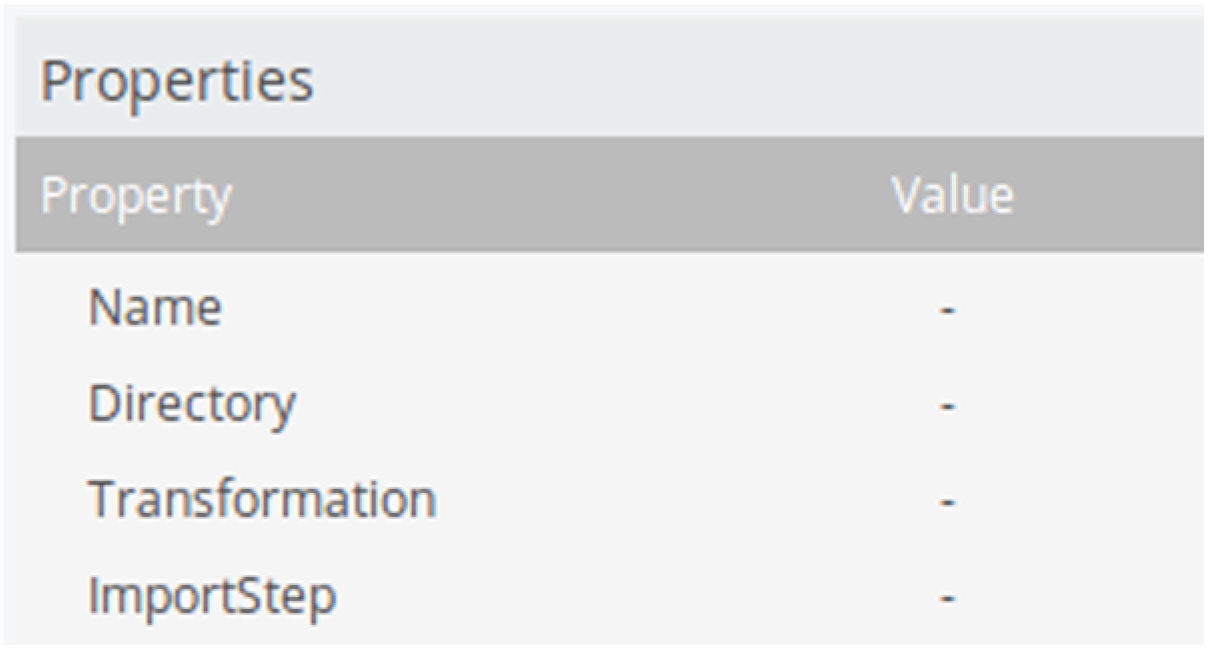
- List of data sources

Legacy Datasources

The following options are available under this heading:

- Kettle transformation

  This data source executes a PDI (Kettle) transformation. Theoretically, you can get data from any source through a Kettle transformation, such as from plain files, Excel spreadsheets, and web services. To access data from Kettle, you will need to provide the name and location of the KTR file and the name of the transformation step which will provide the data. You will also need to define a kettle.TransFromFile connection.

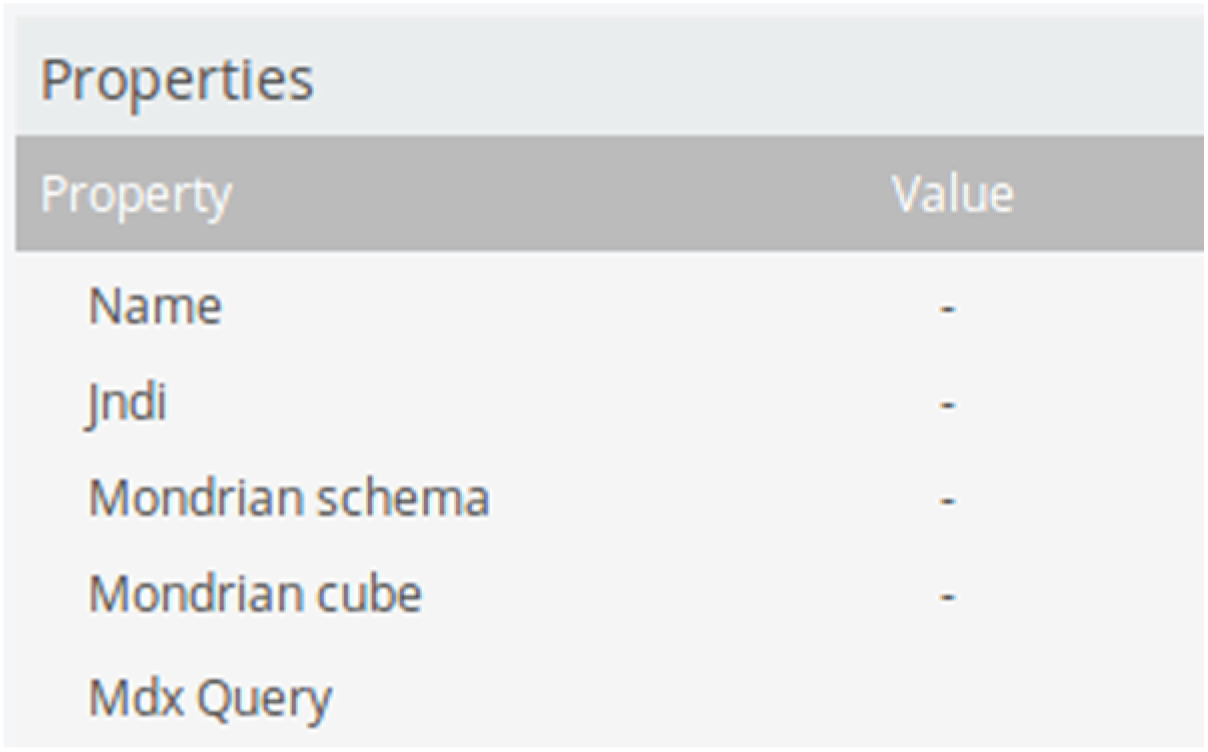


Properties

Property	Value
Name	-
Directory	-
Transformation	-
ImportStep	-

- OLAP MDX query

  This data source executes an MDX query when you provide the JNDI connection string, the Mondrian schema, Mondrian cube, and the MDX query itself.

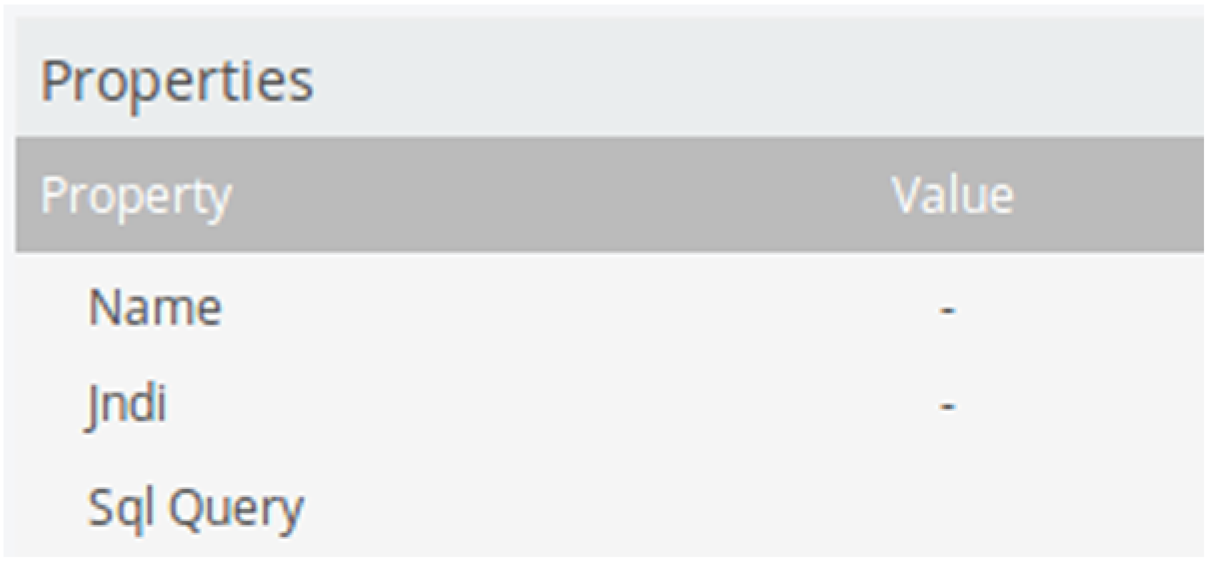


Properties

Property	Value
Name	-
Jndi	-
Mondrian schema	-
Mondrian cube	-
Mdx Query	

- SQL query

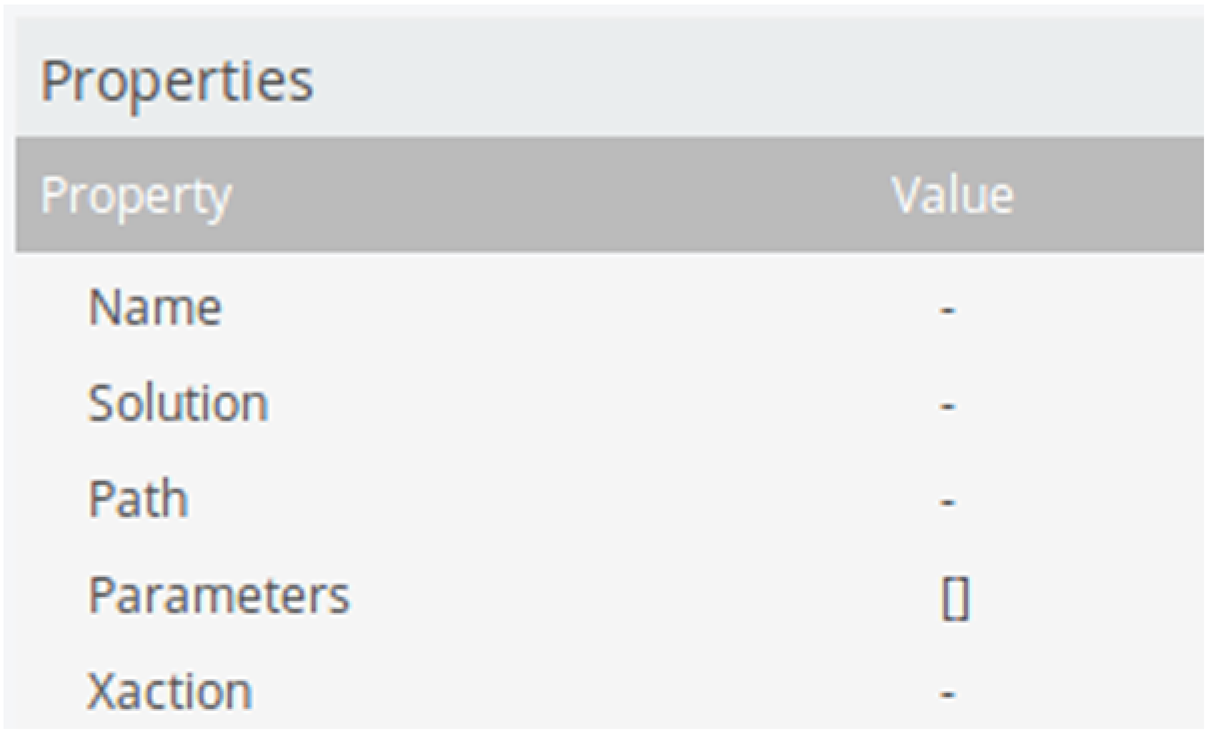
  This data source executes a SQL query when you provide the JNDI connection string and the SQL query.


Properties

Property	Value
Name	-
Jndi	-
Sql Query	

- XAction result set

  This data source retrieves a result set returned from an Xaction call to the Pentaho Server when you provide the location, path, parameters, and name of the Xaction you wish to execute.


Properties

Property	Value
Name	-
Solution	-
Path	-
Parameters	[]
Xaction	-

Parent Topic

- List of data sources

The Pentaho App Builder endpoints

Pentaho App Builder (previously known as SPARKL) is a Community Plugin Kickstarter (CPK) plugin which allows you to easily build other CPK plugins. Kettle transformations or jobs of a CPK plugin are automatically exposed as rest endpoints. While you can view these endpoints in CDE, they are internal to Pentaho App Builder and are not necessary when developing dashboards.

Parent Topic

- List of data sources

MDX Queries

You can fetch data from a Mondrian cube through an MDX query. To access the data through a Mondrian cube, provide the JNDI or JDBC connection properties, the name of the Mondrian schema file (XML), and the MDX query which will return the data. There are four types of MDX data sources:

- denormalizedMdx over mondrianJdbc
- denormalizedMdx over mondrianJndi
- mdx over mondrianJdbc
- mdx over mondrianJndi

MDX queries can be normalized or denormalized. The specifics of each type of query are detailed in the CDA documentation.

Parent Topic

- List of data sources

OLAP4J Queries

These data sources execute queries using the olap4j specification, which is an open Java API for accessing OLAP data. This type of data source can be denormalizedOlap4j over olap4j or olap4j over olap4j.

As with the MDX queries, OLAP4J queries can be normalized or denormalized.

Parent Topic

- List of data sources

Compound Queries

This type of query allows you to combine the result of two distinct queries. Compound queries can be one of two types, JOIN and UNION.

A JOIN compound query merges the result of two queries, using a specified set of keys. You can specify one of four join types: Inner, Left Outer, Right Outer, Full Outer. The result of this join will contain the columns of both queries if they are of the same type. Both the left and right side queries must be identified by an ID. You must also specify which keys (column IDs on the source queries) are used to join the data. This data source has the following properties:

Property	Description
Name	The name of the compound query.
Left	The first query.

Property	Description
Right	The second query.
Parameters	Lists the parameters' name, default value (i.e., the default value if the parameter value is not specified when the data access is called), and type (String, Integer, Numeric, Date, StringArray, IntegerArray, NumericArray, and DateArray) which are passed on to the compound query.
Calculated Columns	The columns to be calculated by a given formula. Each calculated column requires two properties: Name (the name that will be output by CDA), and Formula (the column's definition itself). Formulas are written in Open Formula format.
Columns	Names of the columns, in case you want to rename a particular column.
Left Keys	The ID or IDs of the columns from the first query which are common to the second query.
Output Columns	The IDs of the columns which will be the output from both queries in order, starting with the columns from the left query and then the columns from the right query.
Output Mode	The column's output mode, which will include or exclude the columns set above.
Right Keys	The ID or IDs of the columns from the second query which are common to the first query.
Join Type	The join type to be used, such as Inner, Left Outer, Right Outer, or Full Outer.

A UNION compound query takes the results of two queries with the same number of columns and returns the compounded result set from both queries. A union query data source has the following properties:

Property	Description
Name	The name of the compound query.
Top	The ID of the query which will stay on top.
Bottom	The ID of the query which will stay on the bottom.
Parameters	Lists the parameter's name, default value (i.e., the default value if the parameter value is not specified when the data access is called) and type (String, Integer, Numeric, Date, StringArray, IntegerArray, NumericArray, and DateArray) which are passed on to the compound query.
Calculated Columns	The columns to be calculated by a given formula. Each calculated column requires two properties: Name (the name that will be output by CDA), and Formula (the column's definition itself). Formulas are written in Open

Property	Description
	Formulaformat.
Columns	Names of the columns, in case you want to rename a particular column.

If the columns on both data sets have different names, the name of the column in the top result set will be used in the union's resulting data set.

Parent Topic

- List of data sources

SCRIPTING Queries

These data sources allow you to create ad hoc result sets, such as a small table, for prototyping purposes using Beanshell scripts. These result sets are useful during the dashboard development phase for generating data for a dashboard's components when real data is not yet available. This data source can be one of two types:

- scriptable over scripting

  Using the Beanshell scripting language, we can define a data structure and then create a result set based on this same structure to use in a component. You will need to define the column names, column types, and the result set rows.

```
import org.pentaho.reporting.engine.classic.core.util.TypedTableModel;
String[] columnNames = new String[]{
    "value","name2"
};
Class[] columnTypes = new Class[]{
    Integer.class,
    String.class
};
TypedTableModel model = new TypedTableModel(columnNames, columnTypes);
model.addRow(new Object[]{ new Integer("0"), new String("Name") });
return model;
```

- JSONscriptable over scripting

  This data source is similar to the scriptable data source in that it uses Beanshell script to generate a result set. However, rather than specifying the column names and column types, you just need to define the metadata and create the result set you want to use. This is simple and less prone to bugs than using the scriptable data source.

```
{
  "resultset":[
        ["Name", 0]
  ],
  "metadata":[
    {"colIndex":0,"colType":"String","colName":"value"},
    {"colIndex":1,"colType":"Integer","colName":"name2"}
  ]
}
```

Parent Topic

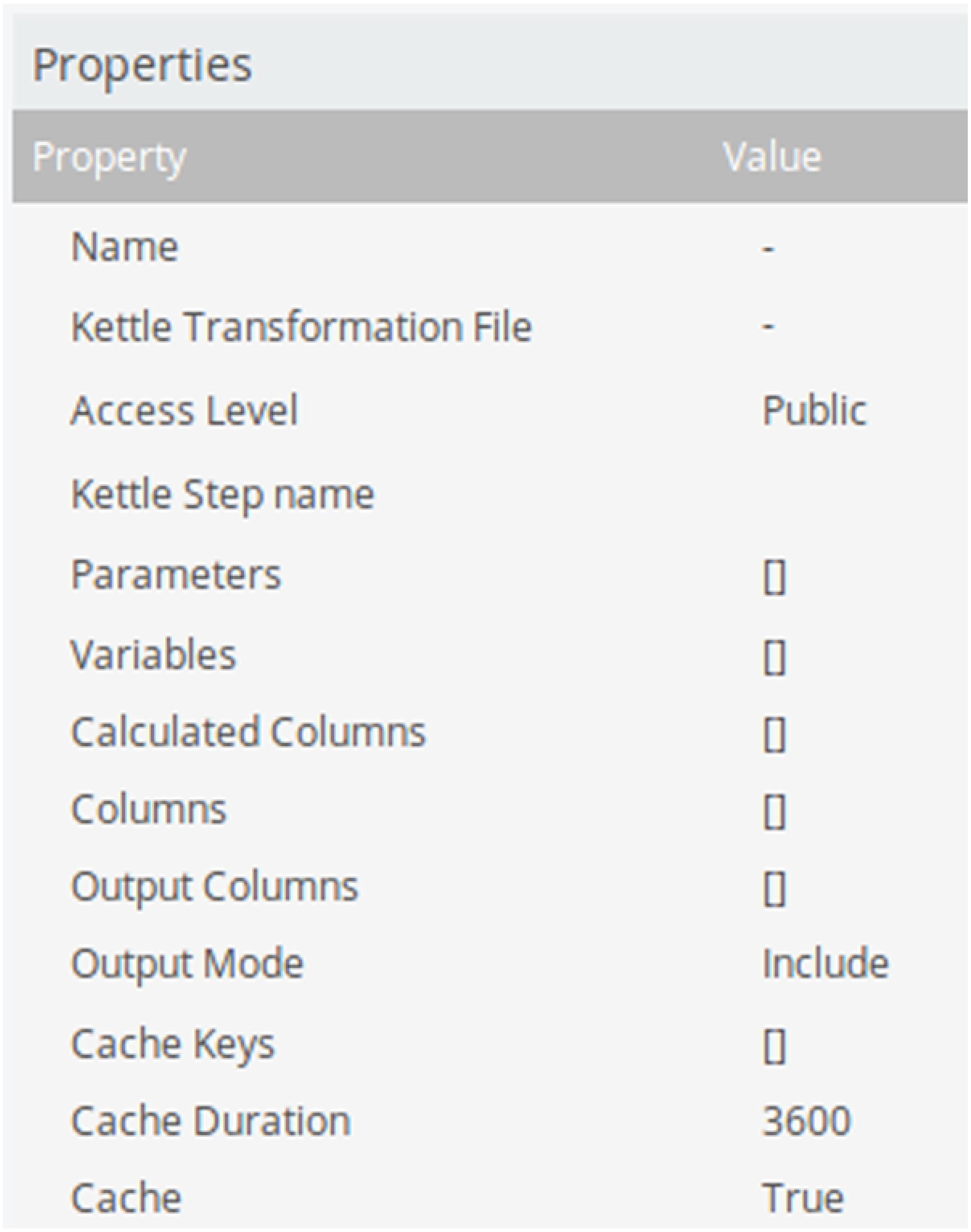
- List of data sources

KETTLE Queries

Using Pentaho Data Integration transformations, you can fetch data from virtually any data source such as plain text files, Excel spreadsheets, and web services.

- kettle over kettleTransFromFile

  To access data from Kettle, you will need to define the Kettle transformation file (KTR) you want to use and the name of the transformation step which will provide the data. You can also pass parameters and variables to the KTR transformation to filter the data.



Properties

Property	Value
Name	-
Kettle Transformation File	-
Access Level	Public
Kettle Step name	
Parameters	[]
Variables	[]
Calculated Columns	[]
Columns	[]
Output Columns	[]
Output Mode	Include
Cache Keys	[]
Cache Duration	3600
Cache	True

Parent Topic

- List of data sources

MQL Queries

Pentaho Metadata defines a business model and query implementation which makes it easy for business users to query data sources in Pentaho tools such as Report Designer and Ad Hoc Reporting. This metadata can be accessed through a MQL query. MQL is the syntax Pentaho Metadata uses for generating SQL queries based on metadata.

- mql over metadata

  To access the data, provide the name and location of the metadata domain file (XMI) and the domain where the data belongs.

Parent Topic

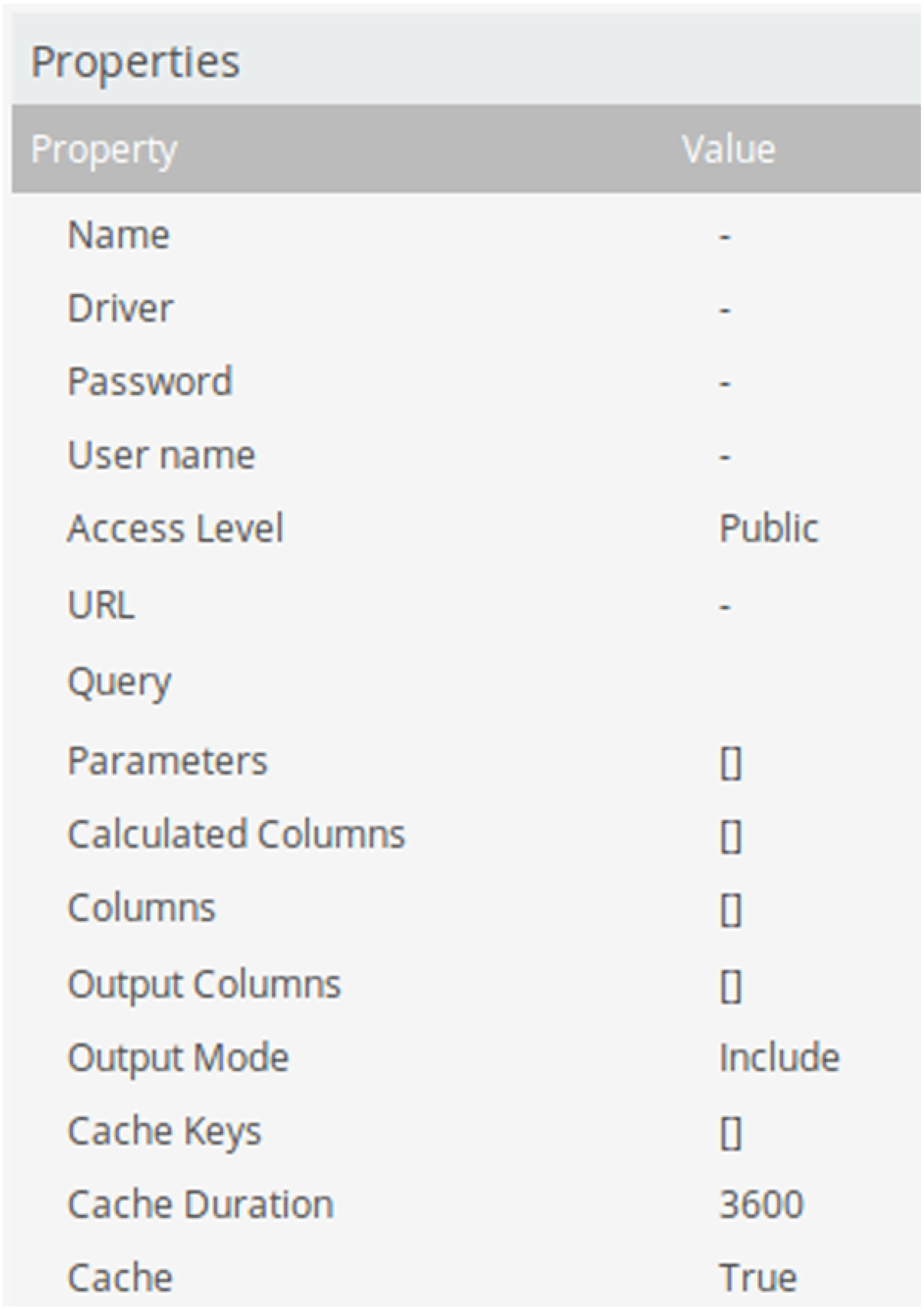
- List of data sources

SQL Queries

Use this type of data source to access data from SQL databases provided you have a JNDI connection or a JDBC driver setup. You can access a SQL database by defining the connection and providing the query to be executed.

- sql over sqlJdbc

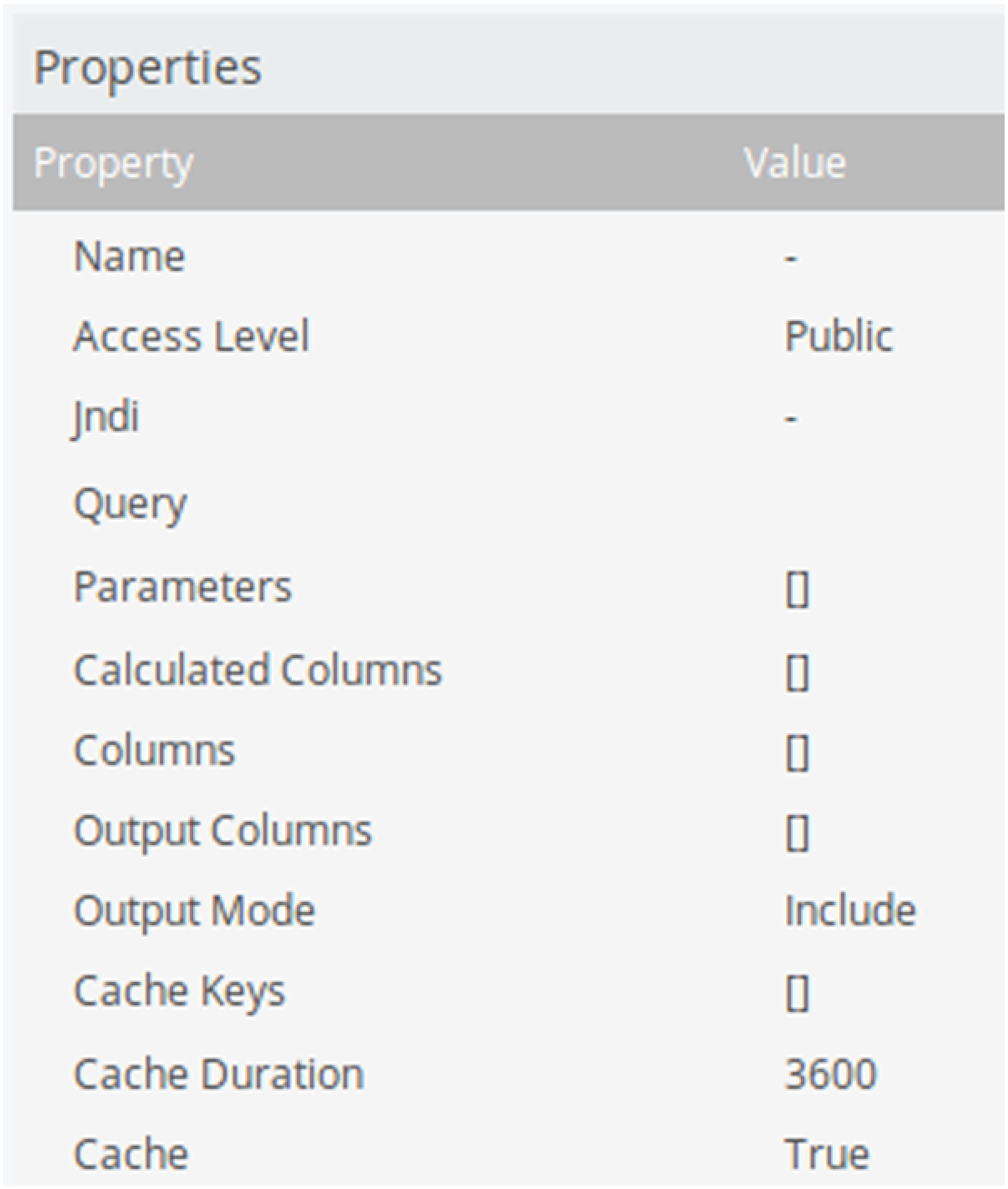
  Besides specifying the query to be used, you also need to specify the information needed to access the data such as the driver, user name and password of a user with access to the data.


Properties

Property	Value
Name	-
Driver	-
Password	-
User name	-
Access Level	Public
URL	-
Query	
Parameters	[]
Calculated Columns	[]
Columns	[]
Output Columns	[]
Output Mode	Include
Cache Keys	[]
Cache Duration	3600
Cache	True

- sql over sqlJndi

  This type of data source employs the Java Naming and Directory Interface (JNDI) which allows software clients to discover and look up data and object via a name, in this case a SQL database. To set up this type of data source you just need to specify the JNDI identifier and the query to be used.



Properties

Property	Value
Name	-
Access Level	Public
Jndi	-
Query	
Parameters	[]
Calculated Columns	[]
Columns	[]
Output Columns	[]
Output Mode	Include
Cache Keys	[]
Cache Duration	3600
Cache	True

Parent Topic

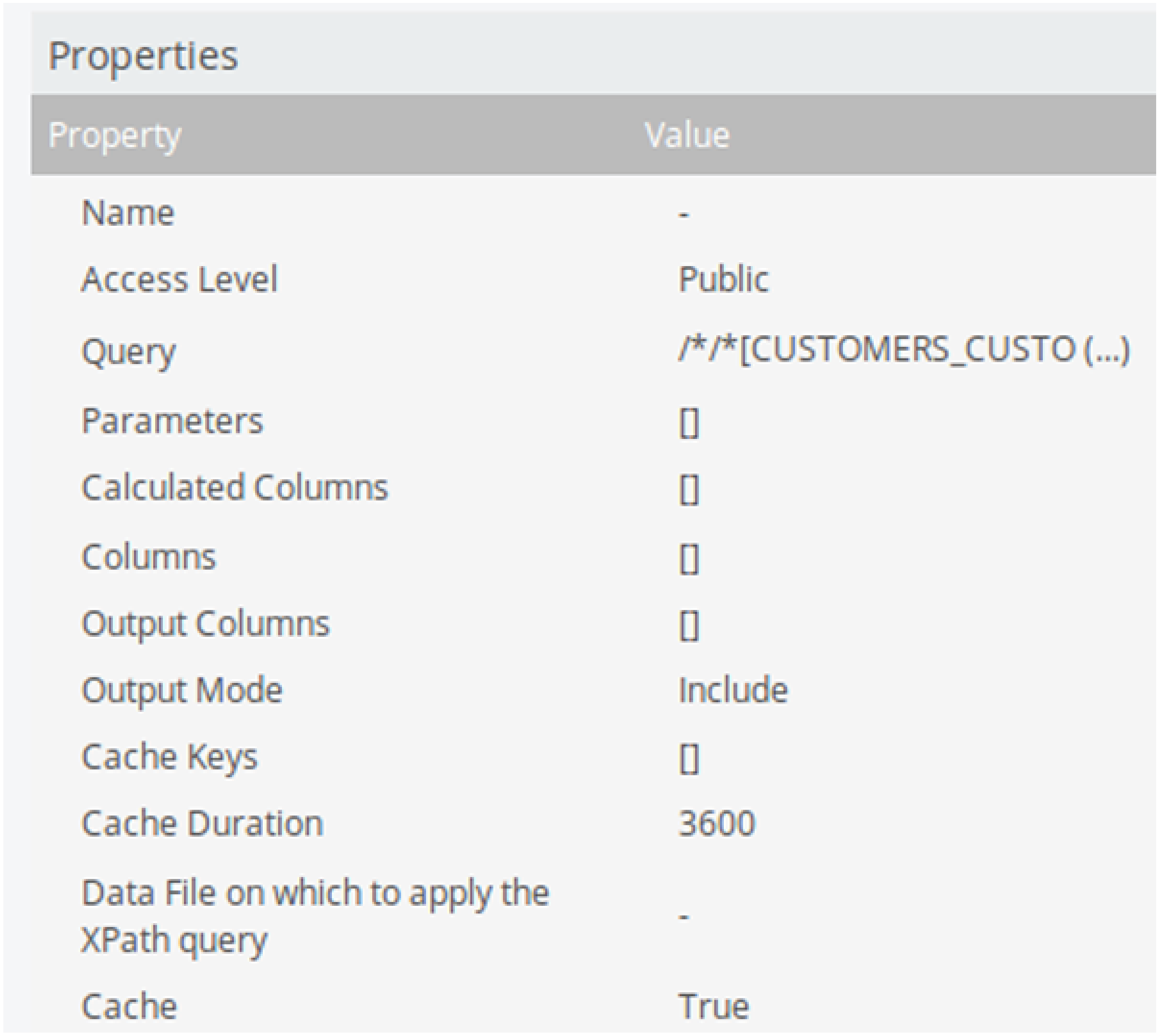
- List of data sources

XPATH Queries

This data source provides the ability to read data from any type of XML file using XPath specifications.

- xPath over xPath

  You need to provide a query as well as the path to the data file on which to apply the xPath query.



Properties

Property	Value
Name	-
Access Level	Public
Query	/*/*[CUSTOMERS_CUSTO (...)
Parameters	[]
Calculated Columns	[]
Columns	[]
Output Columns	[]
Output Mode	Include
Cache Keys	[]
Cache Duration	3600
Data File on which to apply the XPath query	-
Cache	True

Parent Topic

- List of data sources

Common properties

All the listed data sources share the following properties:

Property	Description
Name	The name of the data source.
Access Level	Public or private. Public data sources are available from outside calls, while private data sources can only be called from other data sources, e.g. Compound Queries. The default is Public.

Property	Description
Query	The query itself.
Parameters	Lists the parameters' name, default value (the default value if the parameter value is not specified when the data access is called) and type (String, Integer, Numeric, Date, StringArray, IntegerArray, NumericArray, and DateArray).
Output Columns	Specifies which columns (index) to output and in which order. If not specified, all columns from the query will be returned in the same order as defined in the query.
Output Mode	Set the output mode to include or exclude the columns set above. The default is Include.
Columns	Names of the columns, in case you want to rename a particular column.
Calculated Columns	The columns to be calculated by a given formula. Each calculated column requires two properties: Name (the name that will be output by CDA), and Formula (the column's definition itself). Formulas are written in Open Formula format.
Cache	(Optional) Set to cache the results. Set as either `True` or `False`. The default is True.
Cache Duration	Set the amount of time (in seconds) to keep the results in the cache. The default is 3600.

Parent Topic

- Data Sources perspective

  In the Data Sources perspective you can find various types of data sources which you can employ in a dashboard. These allow you to access the data that you want to use in your dashboard.

Queries with parameters

Parameterized queries allow for dynamic dashboards by translating user interactions via selectors. For example, you can set a parameterized query for the generation of result sets which reflect the visualization needs of the user. Query parameterization is done by enclosing the parameter name you want to pass to the query in curly brackets preceded by a dollar sign: *${parameterName}*. The following query is an example of a parameterized query. In this query, we want to use the status parameter.

```
select {[Measures].[Sales], [Measures].[Quantity]} ON COLUMNS,
NON EMPTY [Time].Children ON ROWS
from [SteelWheelsSales]
where ([Order Status].[${status}])
```

Parent Topic

- Data Sources perspective

  In the Data Sources perspective you can find various types of data sources which you can employ in a dashboard. These allow you to access the data that you want to use in your dashboard.

Final note

Community Data Access (CDA) is a stand-alone plugin tool which allows you to define data sources. CDE incorporates the functionality of CDA, so in most cases you can use CDE to meet all of your data access requirements.

Parent Topic

- Community Dashboard Editor (CDE) perspectives

  For the design of your Dashboard, CDE offers three perspectives. Click the link to learn more about each perspective.