STIBO SYSTEMS
MASTER DATA MANAGEMENT

USER GUIDE

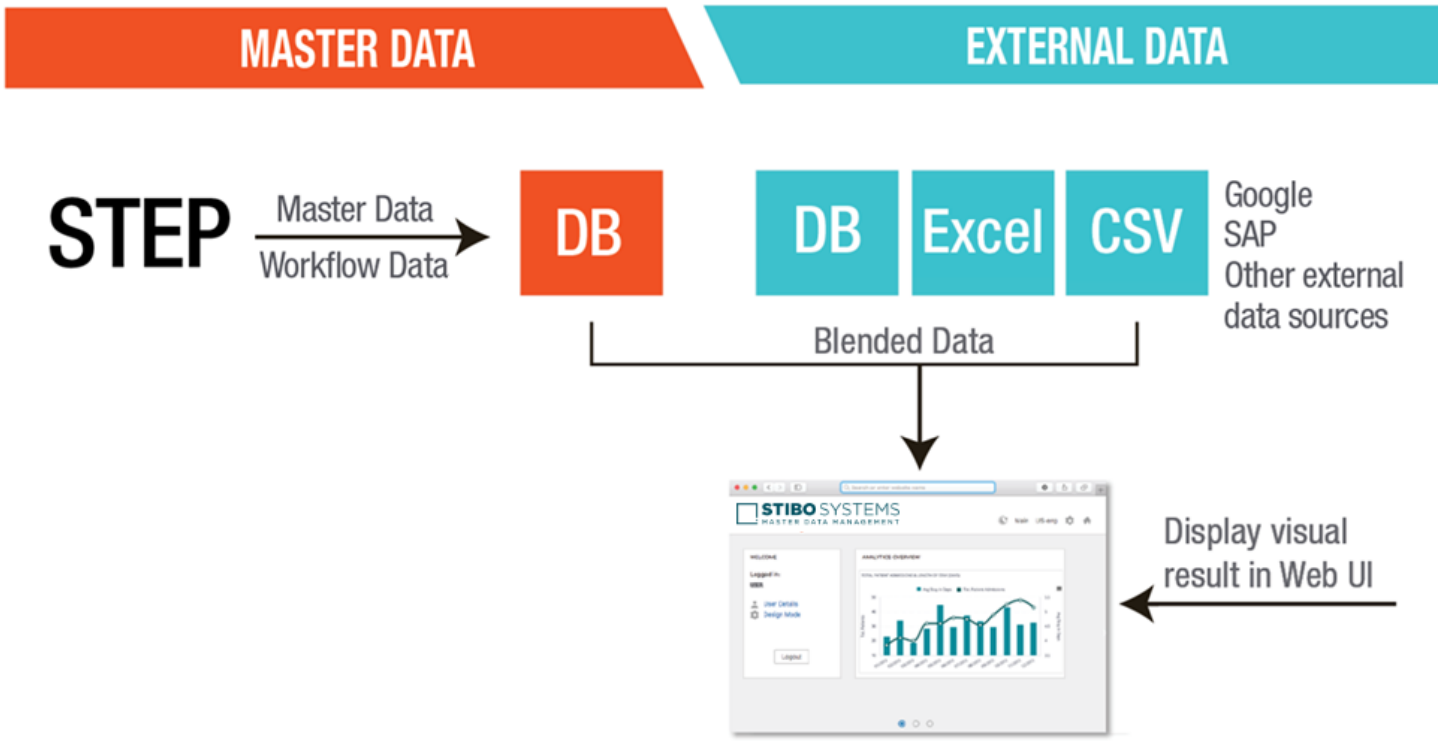
Analytics

Release 11.2 (2023.2)-MP2 – August 2023

Table of Contents

Analytics

Stibo Systems' multidomain approach to master data management gives organizations across a range of industries the ability to leverage data from multiple applications, systems, and domains to drive innovation and insights. The wide selection of analytics offerings from Stibo Systems allow businesses to take this advantage to another level by providing the ability to combine their master data with data from other business systems through seamless integrations between STEP and top-rated BI tools.



These analytics offerings can be used in conjunction with Stibo Systems' Product MDM and Customer MDM solutions, fulfilling a need to make data actionable, allowing users to respond quickly to market trends and improve collaboration through fast, unique insights.

Benefits for various business users of Stibo Systems' analytics offerings include:

- Vendors can examine the quality of submitted products in the Web UI and make adjustments based on presented insights.
- Onboarding managers can identify item onboarding times and potential bottlenecks in workflows, being able to reassign tasks and solve data issues in Web UI based on the presented information.
- Product managers can examine sales information and page views for their products in Web UI to identify market value and adjust the master data based on presented insights.

This guide covers the following analytics offerings from Stibo Systems:

Visual Analytics Integrations

The following analytics solution enables the capability to embed analytics data directly into the Web UI using widgets and screens. This tool allows business users to visualize their operational information, interact with it, and determine trends without having to leave the Web UI:

- **Visual Integration with External Analytics Tools:** The Web UI supports integration with the data analytics tools Tableau, Qlik, and Power BI. Through configuration of the Web UI, admin users can add both homepage widgets and Web UI screens to display visually compelling views of data from these tools in one seamless interface.
  - **Visual Integration with Tableau or Qlik**
  - **Visual Integration with Power BI**

Export of Analytics Data for BI Tools

For users who have existing BI tools that they want to integrate with, Stibo Systems offers more than one method of extracting data from STEP that can be sent to downstream BI tools:

- **Audit Message Framework**: The Audit Message Framework (AMF) is a powerful message delivery solution that sends configurable messages to an external system of the users' choice, allowing data in STEP to be extracted and exposed so it can be processed for statistical analysis.
- **JDBC Delivery Method:** The Java Database Connectivity (JDBC) delivery plugin feature can be configured to automatically (or manually) make STEP data available to an analytics platform, typically Tableau or Qlik. Refer to the **Analytics Using JDBC Example** topic in this guide.

Required Licenses and Components

Most of the analytics solutions discussed in this guide require either a license, an add-on component, or both.

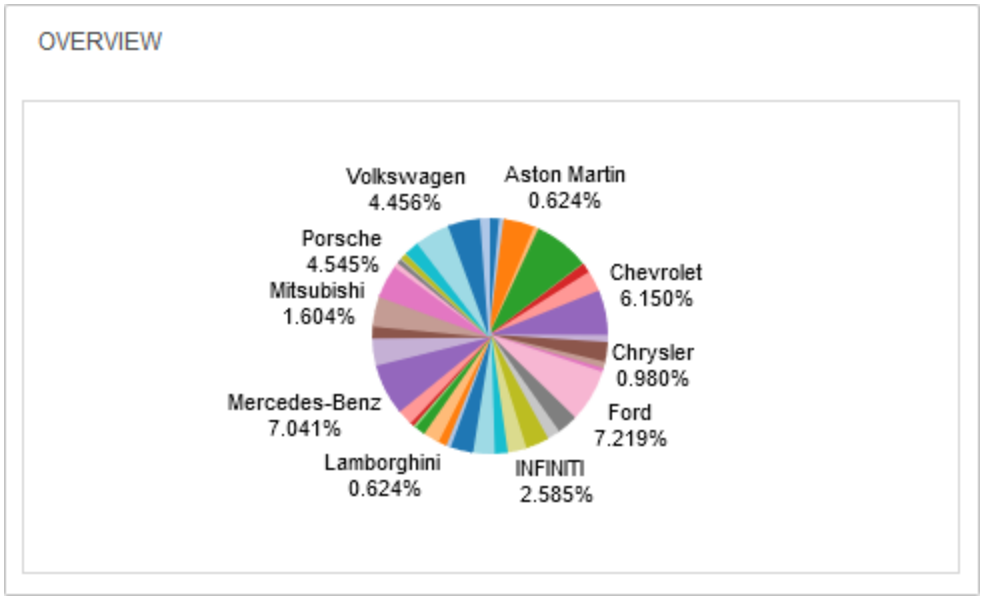
Contact your Stibo Systems account manager to enable licenses for your system. For on-premise systems, instructions for installing components can be found in the **SPOT Program** topic in the **System Administration Guide** found in **Downloadable Documentation**. For SaaS systems, contact your Stibo Systems account manager.

The required licenses and/or components are as follows:

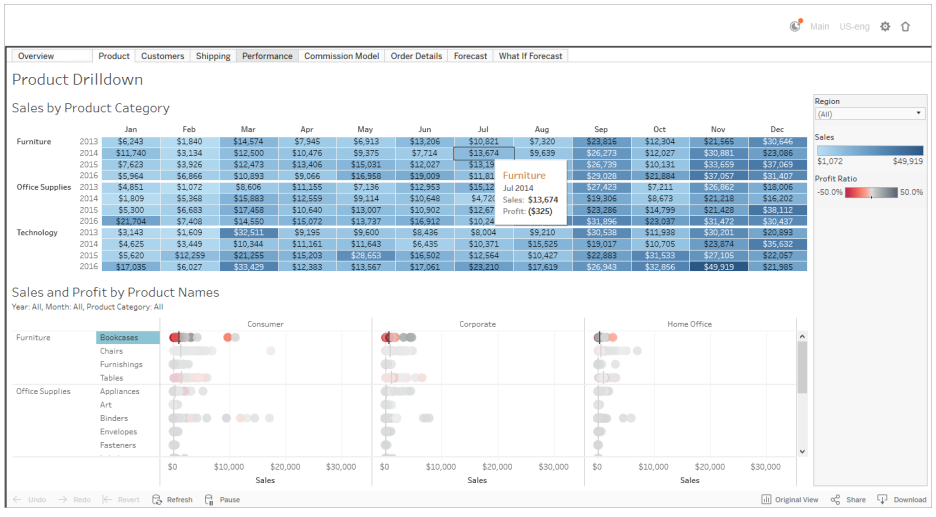
Analytics Offering	Required License	Required Add-on Component(s)
Visual Integration with Analytics Tools	X.WebUI.Analytics	N/A
Audit Message Framework	X.AuditMessaging	audit-messaging
JDBC Delivery Method	None; included in baseline	N/A

Visual Integration with External Analytics Tools

The Web UI supports visual integration with data analytics tools like Tableau Server, Qlik, and Power BI. Through configuration of the Web UI, admin users can add both homepage widgets (example shown below) and Web UI screens designed specifically to display visually compelling views of data (dashboards, reports, and/or tiles) from data analytics tools.



When working with Tableau or Qlik, the **Analytics Widget** Web UI component (shown above) is useful as a quick, simplified view of a dashboard, while the **Analytics Screen** Web UI component (shown below) offers a more expansive view of analytics data, allowing for more interaction.



Users have the ability to utilize analytics tools in a variety of ways. They may:

- Interact with dashboards composed of both master data and external data within the Web UI
- Apply filters to analytics widgets and screens that give users tailored views of data analytics dashboards, reports, and/or tiles
- Configure the Web UI to pass attribute value data into the analytics platform, thus making product-specific views of data from the analytics platform

Information related to configuration of data analytics tools for Tableau and Qlik in the Web UI can be found in the **Visual Integration with Tableau or Qlik** documentation.

To accommodate authentication with the Microsoft API, the Power BI integration uses its own components, the **Power BI Analytics Widget** and the **Power BI Analytics Screen**. These are detailed in more depth in the **Visual Integration with Power BI** documentation.

Prerequisites

To access the Web UI analytics components, the X.WebUI.Analytics license must be enabled on your system. Contact your account manager for more information and to enable licenses for your system.

Visual Integration with Tableau or Qlik

The Web UI supports integration with the data analytics tools Tableau Server and Qlik Server. Data analytics dashboards can be displayed in either a homepage widget or in a screen. Automatic authentication for Tableau and Qlik is also supported, which allows users seamless access to dashboards in the Web UI.

A successful integration that includes automatic authentication may require configuration in the Web UI, the data analytics tool, and the application server. This guide describes only those actions that can be taken in the Web UI and the application server. Configuration actions that must be taken in the relevant data analytics tools are detailed in the appropriate vendor documentation.

This functionality supports the sharing of attribute values with the analytics environment, thus creating a dynamic dashboard that is filtered based on the selected node (e.g., product).

Configuring the Analytics Screen

The **Analytics Screen** is a Web UI screen that can display a much larger analytics dashboard. The screen is typically added as a tabbed page to any mapped object where the specific data analytics dashboard is most pertinent.

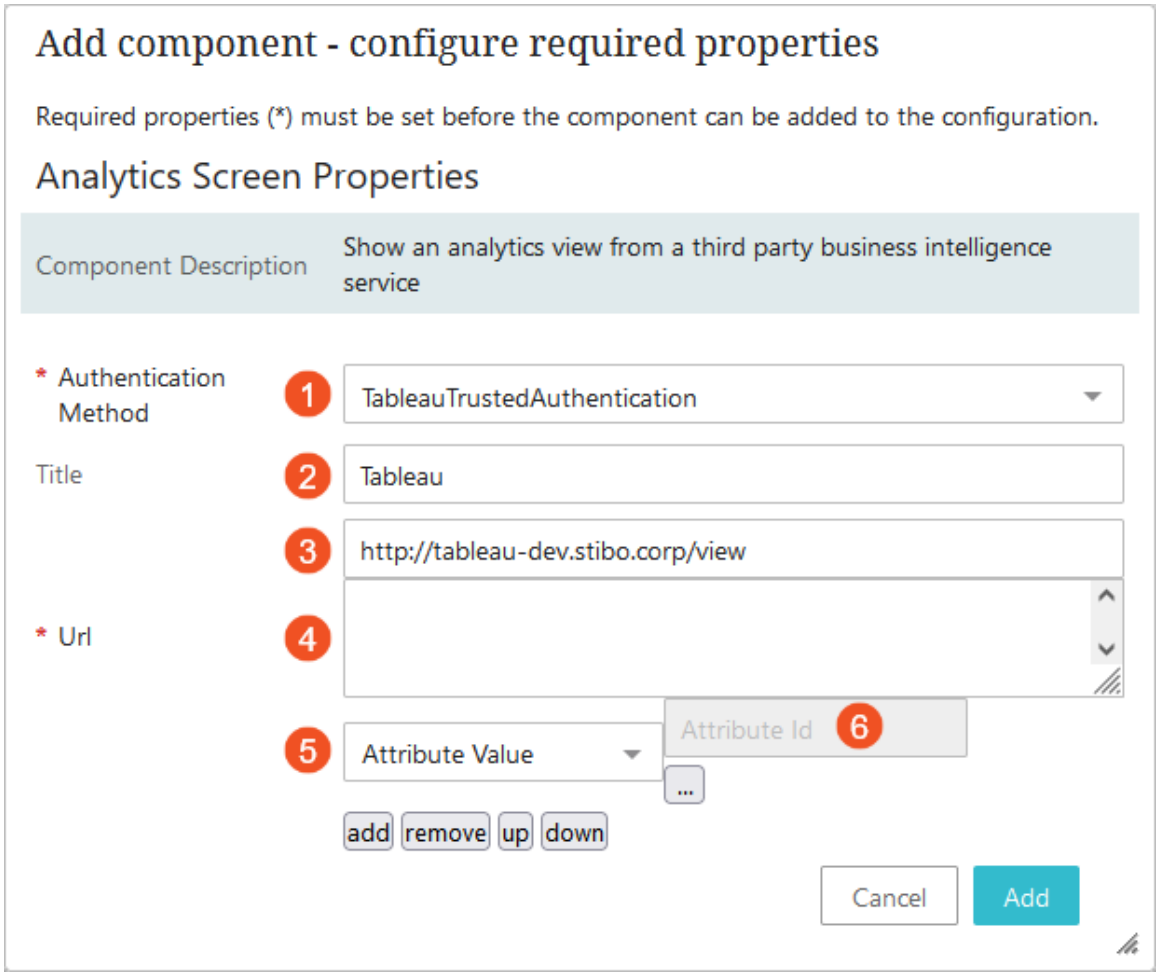
Both the analytics widget and the analytics screen are configured in almost identical ways. The configuration steps described below can be applied to configuring either the widget or the screen, except where expressly noted.

Adding the Analytics Screen

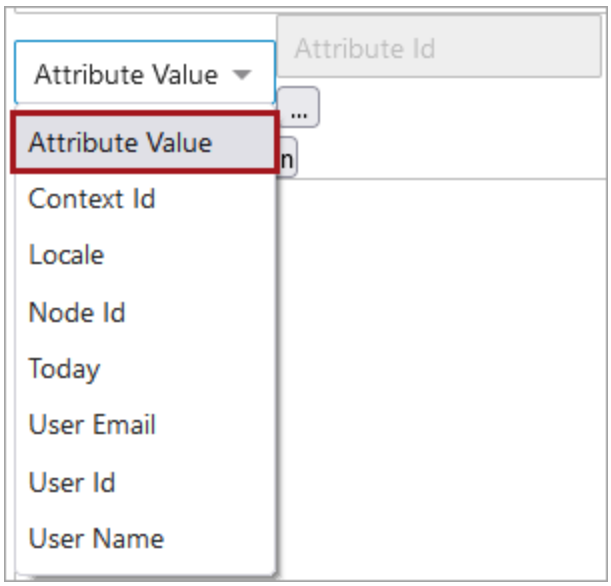
The analytics screen must be added before it can be configured as described below. The method for adding screens in the Web UI is detailed in the **Creating a New Screen** section of the **Design Mode Basics** topic.

Setting up the Analytics Screen

Using the analytics screen, admin users are able to create a view from the Web UI into a published data analytics dashboard. Below is a screenshot of the Properties window for the Analytics Screen in the Web UI designer. The parameters that appear in the screenshot are described in detail directly beneath the screenshot.



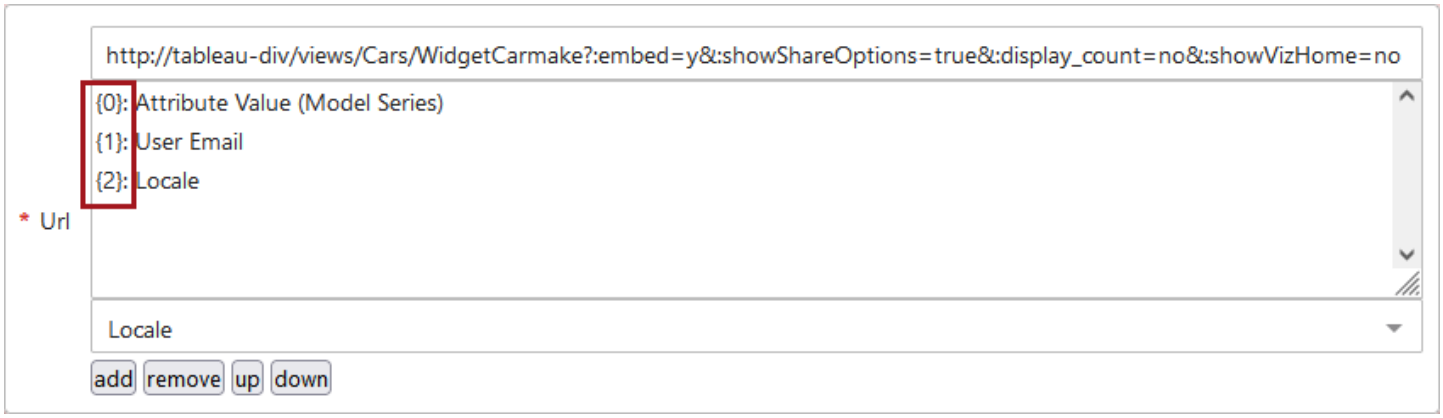
1. **Authentication Method:** This parameter enables users to configure automatic access to Tableau or Qlik dashboards through the Web UI analytics component (screen or widget). For more information on setting up authentication in the Web UI, refer to the **Configuring Authentication** section at the end of this topic.
2. **Title:** Content added to this field appears at the top of the Web UI analytics component (screen or widget).
3. **Url:** This field relates to the analytics server dashboard URL that will be viewed in the Web UI. The URL can be static (showing a dashboard whose display is not dependent on which object is currently selected) or dynamic (showing a dashboard that is dependent on which object is currently selected). A static URL contains no mapping to STEP attributes. A dynamic URL does.
4. The fourth field displays the attributes selected in fields five and six with their corresponding placeholder numbers (in braces).
5. The fifth field is a dropdown menu from which users can select one of eight attribute types to create a dynamic dashboard in the analytics screen. The attributes selected, if matched precisely in the analytics tool, can give users a filtered view of a dashboard based on which object is selected. For instance, if a dashboard has been configured to display a pie chart representing data from ten different car manufacturers, but the user wishes only to examine data for one of those manufacturers, keying on the attribute value 'manufacturer' (which must be present in both the dashboard and the Web UI analytics screen), the Web UI screen can show the same pie chart with only the data for that one manufacturer displaying. The available attribute types are as follows:



- **Attribute Value** returns, or filters the dashboard view for, a STEP attribute value
- **ContextID** returns the current context ID
- **Locale** returns the selected locale, which is used for the Web UI session
- **NodeID** returns the STEP ID of the current object
- **Today** returns the current date. For other date formats, this functionality uses the rules for SimpleDateFormatter in Java
- **User Email** returns the email address of the logged-in user (if specified)
- **User Id** returns the user ID of the logged-in user
- **User Name** returns the name of the logged-in user

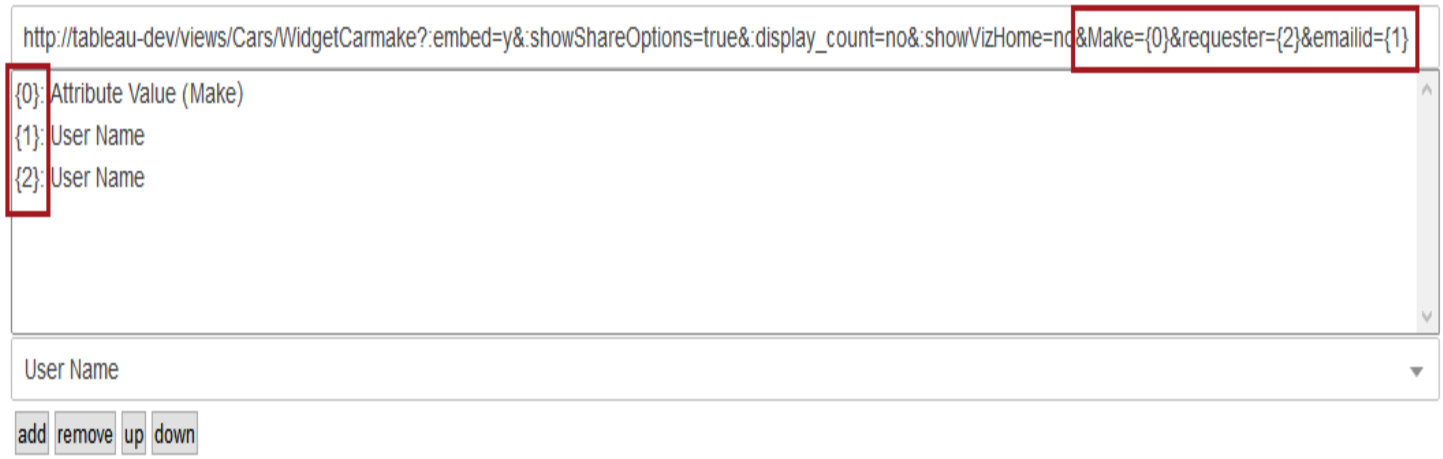
6. The sixth field will display when either 'Attribute Value' or 'Today' is selected in the fifth field. For instance, if the user selects 'Attribute Value,' the sixth field displays with a node picker button (...) beside it. Once clicked, the user may then select the appropriate attribute value in the node picker window. If a user selects 'Today', then the sixth field will display allowing users to add the date in the proper format.

After the attributes needed to implement a dynamic dashboard view have been added, but before the URL has been manually edited to include the placeholders, the URL might look something like this:

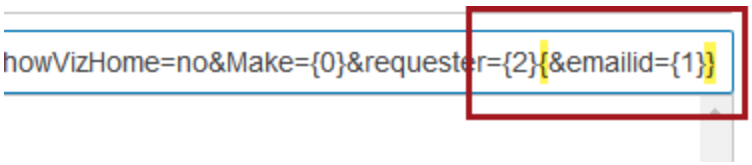


As shown in the screenshot above, the selected attributes are listed along with a braced number (ex. '{0}'). These braced numbers act as placeholders that can be manually added into the URL. Adding these braced numbers into the URL actuates a dynamic view of the dashboard.

In the example below, the Tableau field 'Make' is mapped to placeholder {0}, Tableau field 'requester' is mapped to placeholder {2}, and Tableau field 'emailed' is mapped to placeholder {1}.



Braces can be encapsulated within braces to ensure that a URL is always valid (e.g., to avoid 404 errors), especially if one of the placeholders does not return any content. If such a placeholder within the encapsulated section does not return content, the entire section within braces will be removed from the URL parameter. An example of this is shown in the screenshot below.



For more information on how to structure the URL, users should reference the 'Using Field and Filter Values in URLs' topic in the documentation available for the relevant analytics tool.

Configuring the Analytics Widget

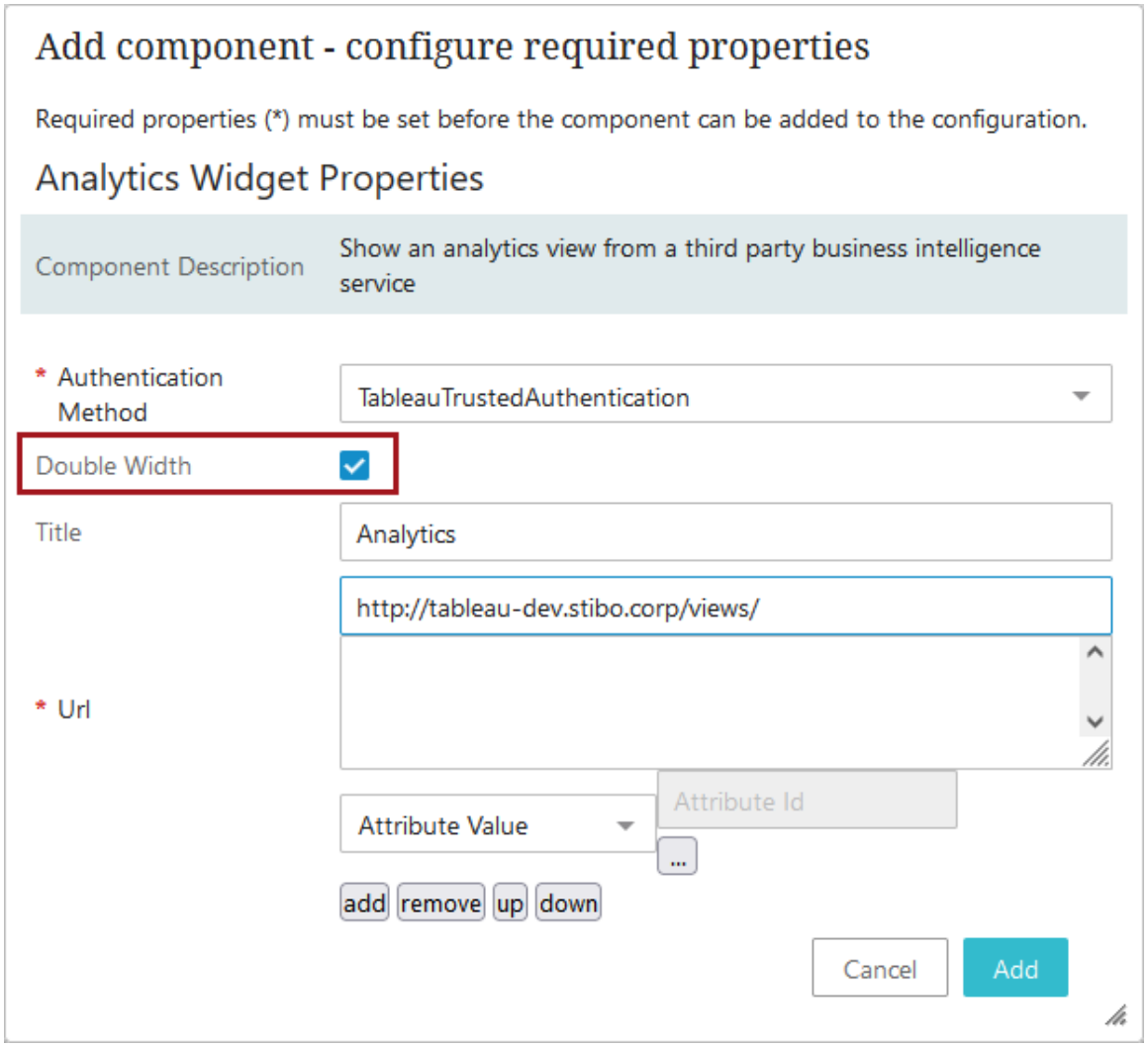
The **Analytics Widget** provides users with a quick and simple view of an analytics dashboard on the Web UI's homepage. The appearance and behavior of the analytics widget is, in large part, determined by how the dashboard is configured in the analytics tool itself. These dashboards can vary greatly based on the specific business need. Analytics widgets can be single- or double-width, and multiple homepage widgets can be configured to display multiple views of analytics dashboards.

Adding the Analytics Widget

The Analytics Widget must be added to the homepage prior to configuration. Details on how to do this can be found in the **Adding Widgets to a Homepage** topic in the **Web UI Getting Started** section of the **Web User Interfaces** documentation.

Setting up the Analytics Widget

The Analytics Widget Properties window in which users configure the widget is shown below. The only parameter for the analytics widget that is different from those used to configure the analytics screen is the 'Double Width' parameter (indicated by a red box in the screenshot below). This parameter determines the width of the widget. The size of the dashboard display is configured in Tableau, but it is recommended that this box is checked to create a widget in the Web UI that is double-width. This allows the most information to be displayed.



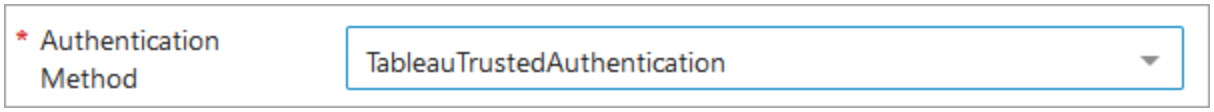
Additionally, using the filtering functionality for a homepage widget would not conform to the recommended practices for this Web UI component. Optimally, the homepage widget should be neutral with respect to which objects are selected in the hierarchy.

For information on the purpose of the other parameters, refer to the description in the Configuring the Analytics Screen section of this topic.

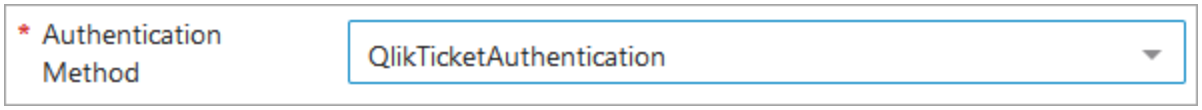
Configuring Authentication

Automatic authentication refers to the capability of enabling seamless access to the data analytics tool upon logging in to the Web UI. The analytics functionality available in STEP is built to accommodate many analytics tools, but with respect to automatic authentication of users, STEP supports Tableau and Qlik. Once authentication is configured correctly, the user will be granted access to the analytics server restricted by the licensing and user privileges set on the analytics environment. A suitably privileged user will have a seamless display of a Tableau or Qlik dashboard in their Web UI.

- To enable automatic sign-on for access to Tableau dashboards, users must select the 'TableauTrustedAuthentication' option from the dropdown.



- To enable automatic sign-on for access to Qlik dashboards, users must select the 'QlikTicketAuthentication' option.



Tableau

To enable automated Tableau authentication, admin users must configure Tableau to add the relevant STEP application server as a trusted host. Further, the username used in Tableau must match the username used in the Web UI to enable the authentication (the passwords, however, can be different). Refer to the 'Trusted Authentication' documentation on the official Tableau website for details on this functionality.

Qlik

Successful configuration of automatic authentication for Qlik in the Web UI requires that a number of conditions be met:

- Access must be established to the REST API on port 4243
- a certificate, created and configured on the Qlik server, must be placed on the user's app server
- the Web UI user name and the user name used to access Qlik must be the same (the passwords can be different)
- three new properties must be added to the sharedconfig.properties file

Adding the certificate to the app server

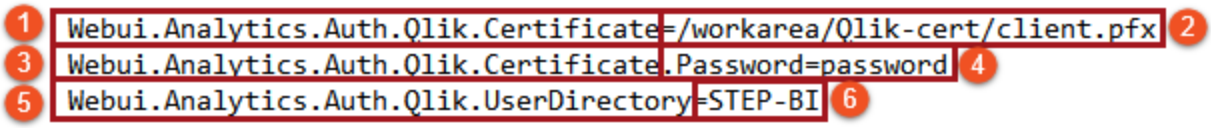
For information on how to add the certificate to the app server, refer to the 'Certificate Trust' section of the 'Security Overview' documentation on Qlik's official website.

Adding the properties to the configuration file

Below is an example of how the analytics properties should be written:

```
Webui.Analytics.Auth.Qlik.Certificate=/workarea/Qlik-cert/client.pfx
Webui.Analytics.Auth.Qlik.Certificate.Password=password
Webui.Analytics.Auth.Qlik.UserDirectory=STEP-BI
```

Below are the three sample properties divided into the components required to enable placement of the certificate.



1. Required text to enable the certificate location property
2. The application server path to where the certificate is posted ('/workarea/Qlik-cert/'), followed by the file name of the certificate ('client.pfx')
3. Required text to enable the certificate password property
4. The password assigned to the certificate when it was created in Qlik
5. Required text to enable the user directory property
6. The name of the user directory in Qlik in which the allowable user names are contained

Once the properties are properly added to the sharedconfig.properties file, the user should save the file and then execute a stop / start of the relevant STEP system. Following this (allow for a few minutes) the user with a Qlik analytics widget or screen will have automated sign-in enabled, which means that access to the Qlik server will automatically occur immediately following a successful sign-in to the Web UI.

Useful Hints Regarding Configuration of Tableau

Listed below are some helpful steps users may take to facilitate proper configuration of a Web UI integration with Tableau:

- Enable navigation from embedded Tableau Web UI dashboards to other standard Web UI screens using Tableau 'Actions'. To do this, users must create calculated Tableau fields to generate STEP Web UI URLs using STEP IDs and screen IDs. Be sure to have STEP IDs in your underlying Tableau data to identify the target object in the generated URL. This method can be extended to access any STEP REST API functionality. For example, it is possible to initiate a workflow from within an embedded Web UI Tableau screen.
- To inform users when the dashboard was last updated, users can configure Tableau to display a timestamp. This can be done by adding the tag “Data Update Time” in the dashboard title. This is particularly useful when you have automated update processes.
- To transform STEP multi-value attributes into a Tableau list of values, use parameters and calculated fields. Users must define a Tableau parameter with the list of user-selectable values. Then, a Tableau calculated field with a Boolean condition (e.g., Tableau’s ‘CONTAINS’ function) must be defined that compares the multi-value attribute in the dashboard source data with the parameter list of values. Finally, users must add the calculated field as a filter with the option 'True' selected.
- Consider using data source filters to optimize the performance of Tableau. Data source filters exclude unnecessary data from your dashboard, which has the effect of improving dashboard performance. Standard filters only hide unnecessary data, which means the data is still processed, leading to potentially diminished dashboard performance.

- Prevent the embedded Web UI Tableau dashboard from creating new windows or tabs when following links to other screens in the Web UI by including the tag ":linktarget=_self" in the Tableau URL that is added into the STEP Web UI designer.
- To configure a Tableau dashboard to fit into a double-size widget, the screen should be set to 485px wide by 260px high.

Visual Integration with Power BI

The visual analytics integration with Microsoft Power BI allows users to view and interact with Power BI reports, dashboards, and tiles in the Web UI, providing an opportunity to analyze and take action on data from both STEP (the MDM platform) and external systems in one seamless interface. By embedding Power BI data using Web UI screens and widgets, business users can visualize their operational information and determine trends without having to leave the Web UI.

Prerequisites

- Customers must already have a licensed Power BI account. There are no preconfigured dashboards or reports delivered with the visual integration with Power BI; it is up to users to pre-create these in their existing Power BI instance before they can be viewed in the Web UI.
- To access the Power BI visual analytics integration, the X.WebUI.Analytics license must be enabled on your system. Additional setup tasks and system configurations must also be performed by Stibo Systems' Technical Services team upon initial setup. Contact your account manager for additional information and to enable licenses for your system.

Topics in this Guide

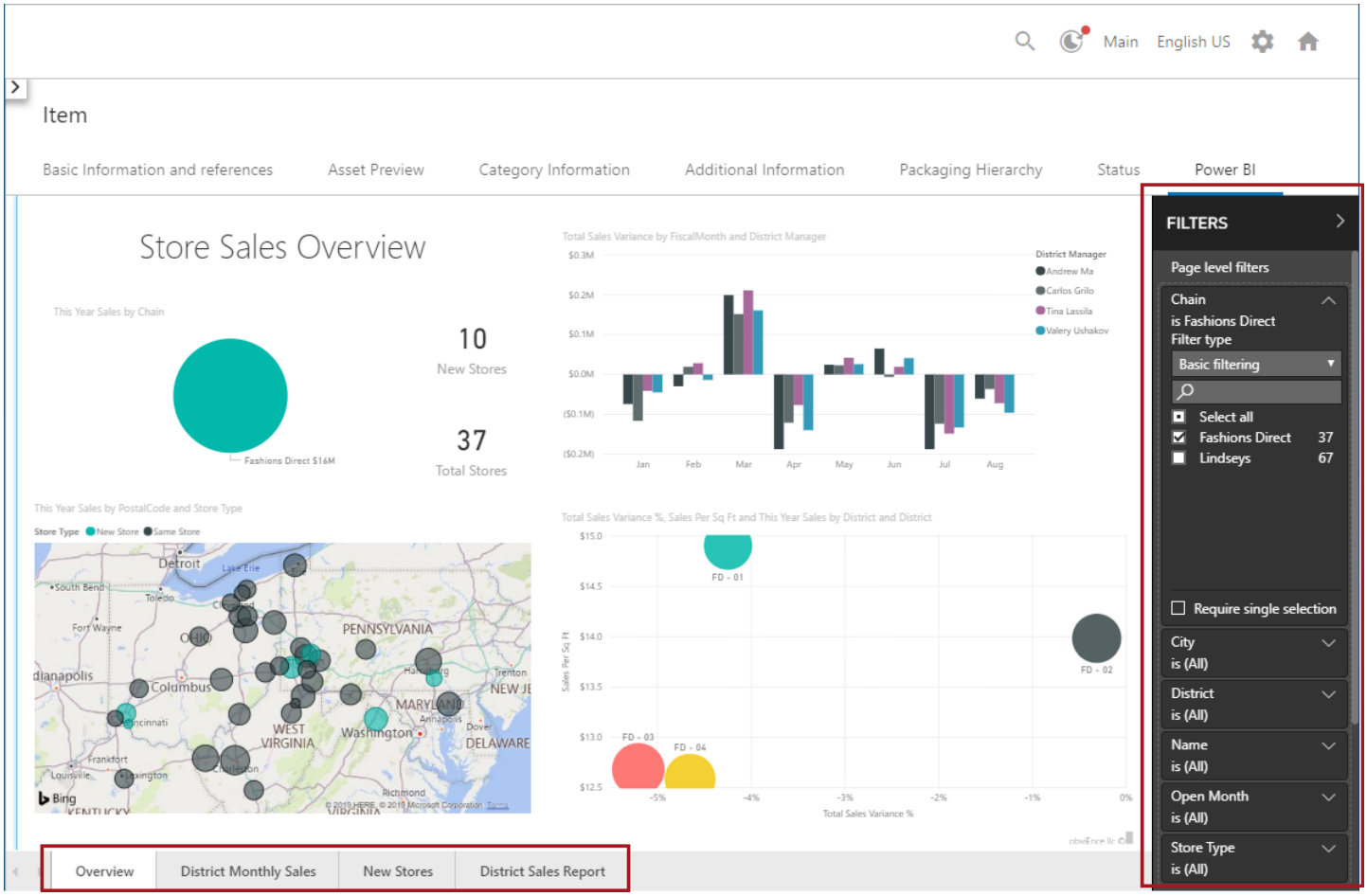
This guide covers the following topics related to the visual integration with Power BI:

- **Power BI Web UI Screen**: Includes example Power BI screens and instructions for configuring the Power BI Screen
- **Power BI Web UI Widget:** Includes example Power BI widgets and instructions for configuring the Power BI Widget
- **Power BI Flyout Panel:** Includes information for configuring and using the Power BI flyout panel
- **Locating Power BI IDs:** Explains how to locate the Power BI IDs that are required to configure the Power BI Web UI
- **Example Power BI Report Filters**: Provides several example JSON-formatted report filters that can be used when configuring the Power BI Web UI to filter the results of displayed data
- **Power BI Row-Level Security**: Provides an overview of how to configure row-level security (RLS) for Power BI dashboards and reports based on users and user groups in STEP and their corresponding roles and rules in Power BI
- **Power BI Authentication Configuration**: Provides a brief overview of how to configure user authentication between STEP and Power BI as well as the sharedconfig.properties required on the STEP application server to enable this authentication

Power BI Web UI Screen

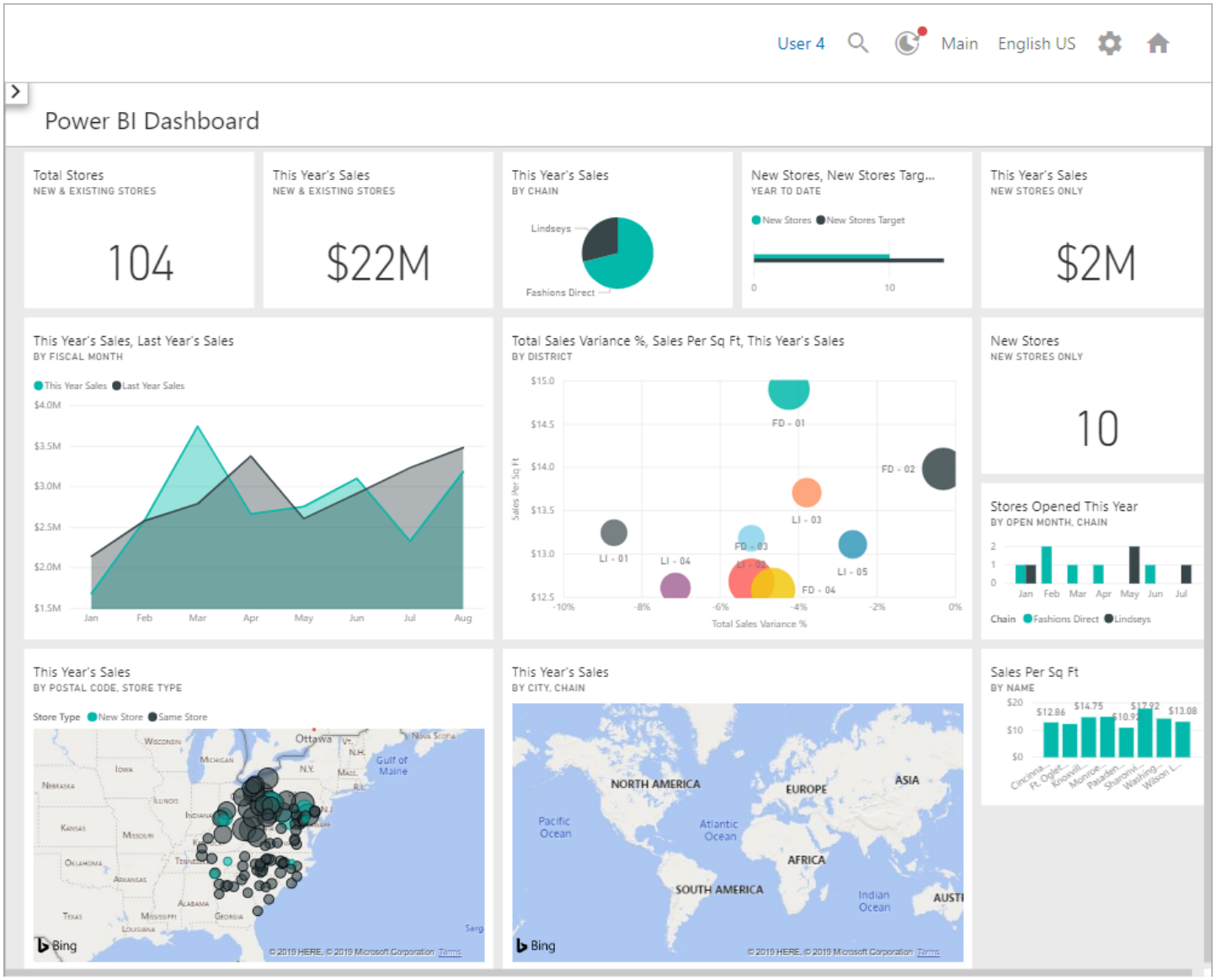
The visual integration with Power BI uses two components to display analytics data in the Web UI: the Power BI Widget and the Power BI Screen. This topic details the configuration of the Power BI Screen. The Power BI Screen is used to configure full-screen Power BI dashboards and Power BI reports.

The following screenshot shows a sample Power BI **report** on a Power BI screen in the Web UI. An optional **filter panel** is shown on the right side of the screen, and an optional **page navigation panel** appears on the bottom.



The following screenshot shows a sample Power BI screen in the Web UI displaying several Power BI tiles as part of a Power BI **dashboard**.

Note: Filters are only available for reports — not for dashboards.



Configuring the Power BI Screen

A Power BI screen can be configured to display Power BI **reports** or Power BI **dashboards**. For example, it can be added as a Sub Screen Tab Page on a Node Details screen if it contains product-specific reports or dashboards. If the screen is intended to display a general report or dashboard that needs no product selection, it can be accessed through a screen navigation link (e.g., from a quick link from the Web UI homepage).

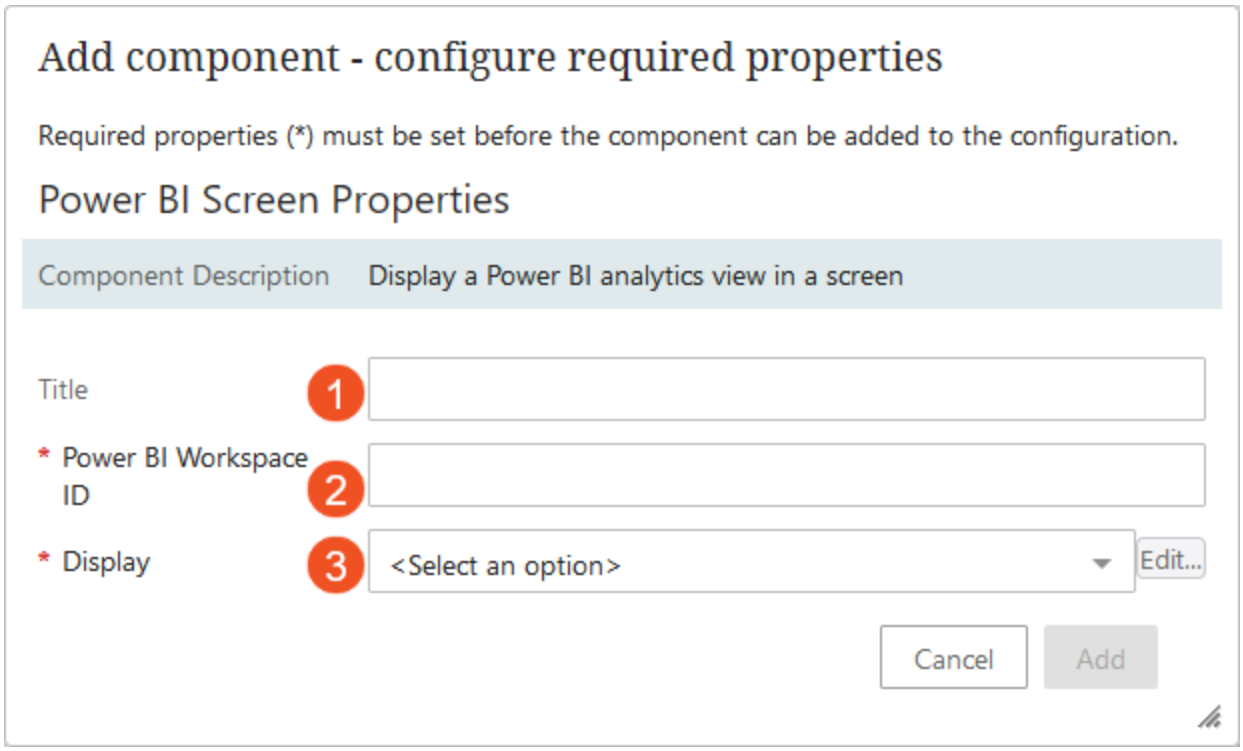
Adding the Power BI Screen

The Power BI screen must be added before it can be configured as described below. The method for adding screens in the Web UI is detailed in the **Design Mode Basics** topic in the **Web UI Getting Started** section of the **Web User Interfaces** documentation.

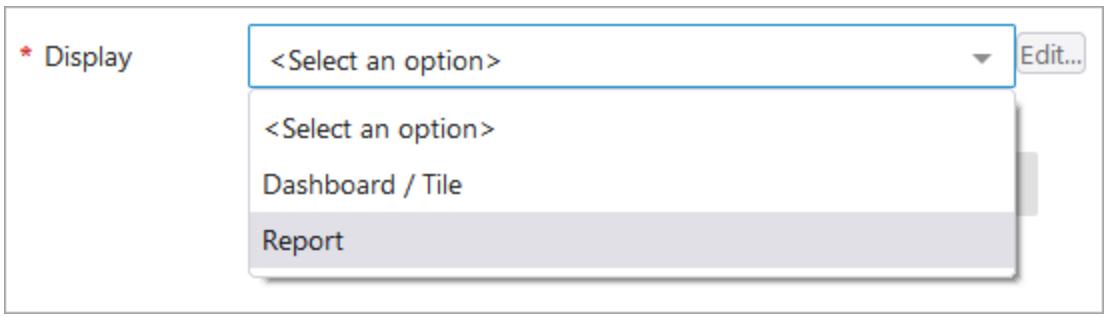
Setting up the Power BI Screen

Using the Power BI screen, admin users are able to create a view from the Web UI into a published Power BI report or dashboard. The Properties window for the Power BI Screen in the Web UI designer is shown below. The parameters that appear in the screenshot are described in detail directly beneath the image. Mandatory selections are marked with an asterisk.

The Power BI screen and Power BI widget are configured in almost identical ways, with some minor exceptions. The configuration steps described below can be applied to configuring either the widget or the screen, except where expressly noted.

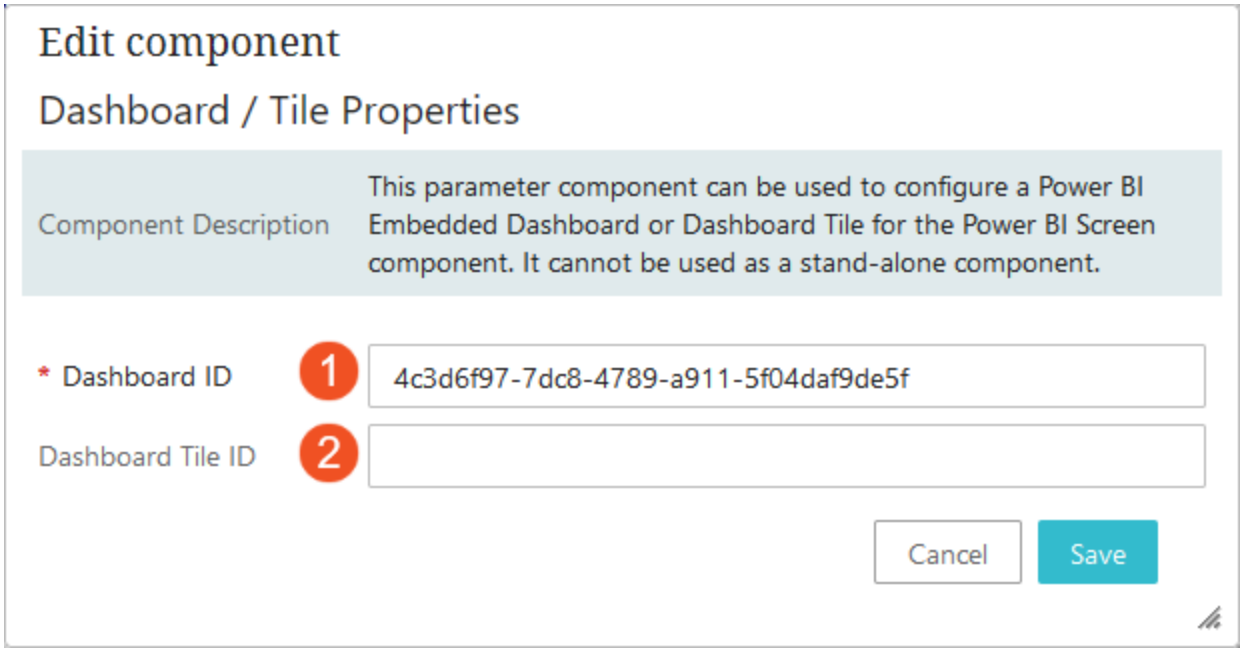


1. **Title:** Content added to this field appears at the top of the Power BI (screen or homepage widget).
2. **Power BI Workspace ID:** Enter the ID of the relevant Power BI Workspace. For example, b556ac84-9d9a-4af6-b569-6a770f3bbe11. For information on how to obtain this value, refer to the **Locating Power BI IDs** topic.
3. **Display:** Select either 'Dashboard / Tile' or 'Report.'



Dashboard / Tile Configuration

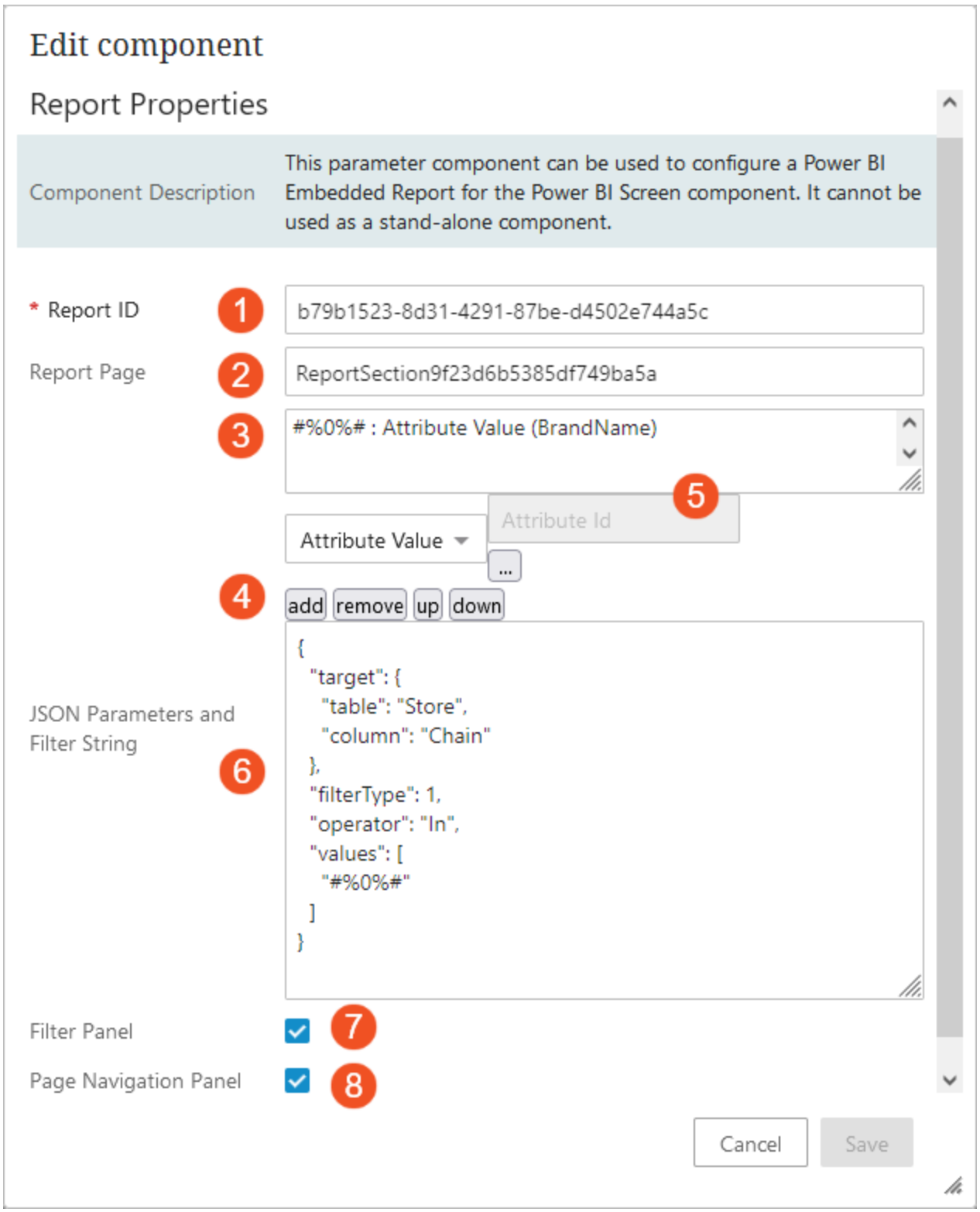
If you are configuring your Power BI screen to display a **dashboard**, select Dashboard / Tile from the Display dropdown, then click the 'Edit...' button to the right of the Display field to open the **Dashboard / Tile Properties** designer window.



1. **Dashboard ID:** Enter the ID of the relevant Power BI dashboard. For example, 4c3d6f97-7dc8-4789-a911-5f04daf9de5f.
2. **Dashboard Tile ID:** Enter the ID of the relevant Power BI dashboard tile. This option will cause the Web UI screen to be composed of a single tile.

Report Configuration

If you are configuring your Power BI screen to display a **report**, select Report from the Display dropdown, then click the 'Edit...' button to the right of the Display field to open the **Report Properties** designer window.



1. **Report ID:** Enter the ID of the relevant Power BI report. For information on how to obtain this value, refer to the **Locating Power BI IDs** topic.

2. **Report Page:** Enter the ID of the Power BI report page that you want to use as the first page to display when you open the report. The ID that needs to be entered can be found in the dashboard URL once the report is published.

   The ID begins with the element 'ReportSection' (refer to URL example below). In the case where you want the first page of the report to display, 'ReportSection' should be entered:

   ```
   https://app.powerbi.com/groups/xx/reports/xx/ReportSection?redirectedFromSignup=1
   ```

   If you want to display another page, the ID format includes IDs after the 'ReportSection' element as displayed below and in the previous image:

   ```
   https://app.powerbi.com/groups/xx/reports/xx/ReportSection9f23d6b5385df749ba5a?redirectedFromSignup=1
   ```

Filter Configuration Options: JSON Parameters and Filter String

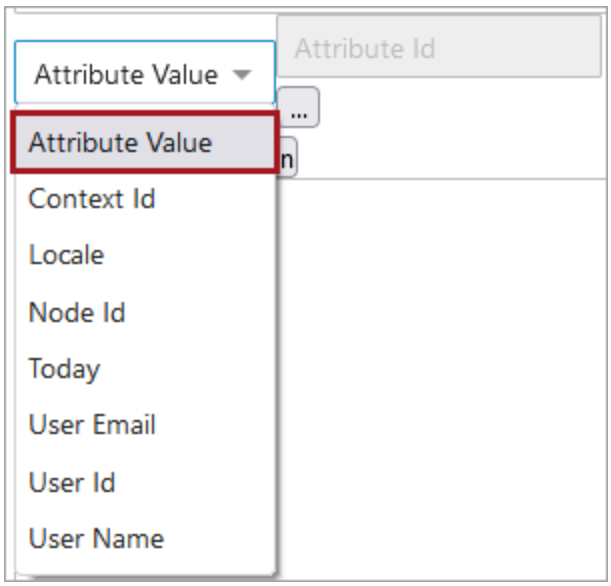
This section contains two fields. The top field (3) is for JSON filter parameters. The bottom field (6) is where the JSON filter string is entered. Use these options to further limit the data that users can refer to, and/or to configure the filters that display in the optional Right Filter Panel, which is addressed in more detail later in this topic. For more information on using JSON strings to filter Power BI reports, refer to the **Example Power BI Report Filters** topic in this guide.

> **Note:** To filter data based on user permissions, e.g., to ensure that the logged-in user only accesses data that is relevant to them, you must use row-level security (RLS). For more information, refer to the **Power BI Row-Level Security** topic in this guide.

3. **Filter parameters** *(field not individually labeled)* – In this field, enter the relevant filter parameters, e.g., 'Attribute Value.' Each selected parameter will be assigned a placeholder string that will be replaced with a context-specific value to be referenced in the JSON query string. These placeholders also contain an embedded integer that identifies the sequence in which the parameter will appear in the filter string, e.g., #%0%# will appear first, #%1%# will appear second, etc. Moving a parameter up or down with the 'up' or 'down' button will automatically renumber the placeholder.

   > **Note:** It is recommended to generate the placeholders for the filter parameters before entering the JSON filter string in the bottom field.

4. The dropdown menu below the filter parameters field allows the selection of various options to create the filter parameters. The available parameters include:



- **Attribute Value:** Returns / filters the view for a STEP attribute value
- **Context Id:** Returns the current context ID
- **Locale**: Returns the selected locale, which is used for the Web UI session
- **Node Id:** Returns the STEP ID of the current object *(valid for Power BI Screen only)*
- **Today:** Returns the current date
- **User Email:** Returns the email address of the logged-in user (if specified)
- **User Id:** Returns the user ID of the logged-in user
- **User Name:** Returns the name of the logged-in user

5. The field directly to the right of the dropdown menu will display when either 'Attribute Value' or 'Today' is selected. If **Attribute Value** is selected, the ellipsis button displays (...). Clicking this button opens the 'Select Node(s)' window, where the relevant attribute is chosen. If **Today** is selected, a 'Date Format' window displays, in which the chosen date format is entered, e.g., yyyy-MM-dd. The date format uses the rules for SimpleDateFormat in Java.
6. **JSON filter string** *(field not individually labeled)* – In this field, enter the desired JSON filter string. It is recommended that the string contain placeholders that correspond with the numbered dynamic values selected for the JSON filter parameters, e.g., #%0%#. #%1%#, etc.
7. **Filter Panel:** Check this box to display a Power BI filter panel on the right side of the Web UI screen. An example is shown in the first screenshot in this topic. The filter panel is optional.

   Note: Any configured filters will still apply to the Power BI report even if no filter panel is present.

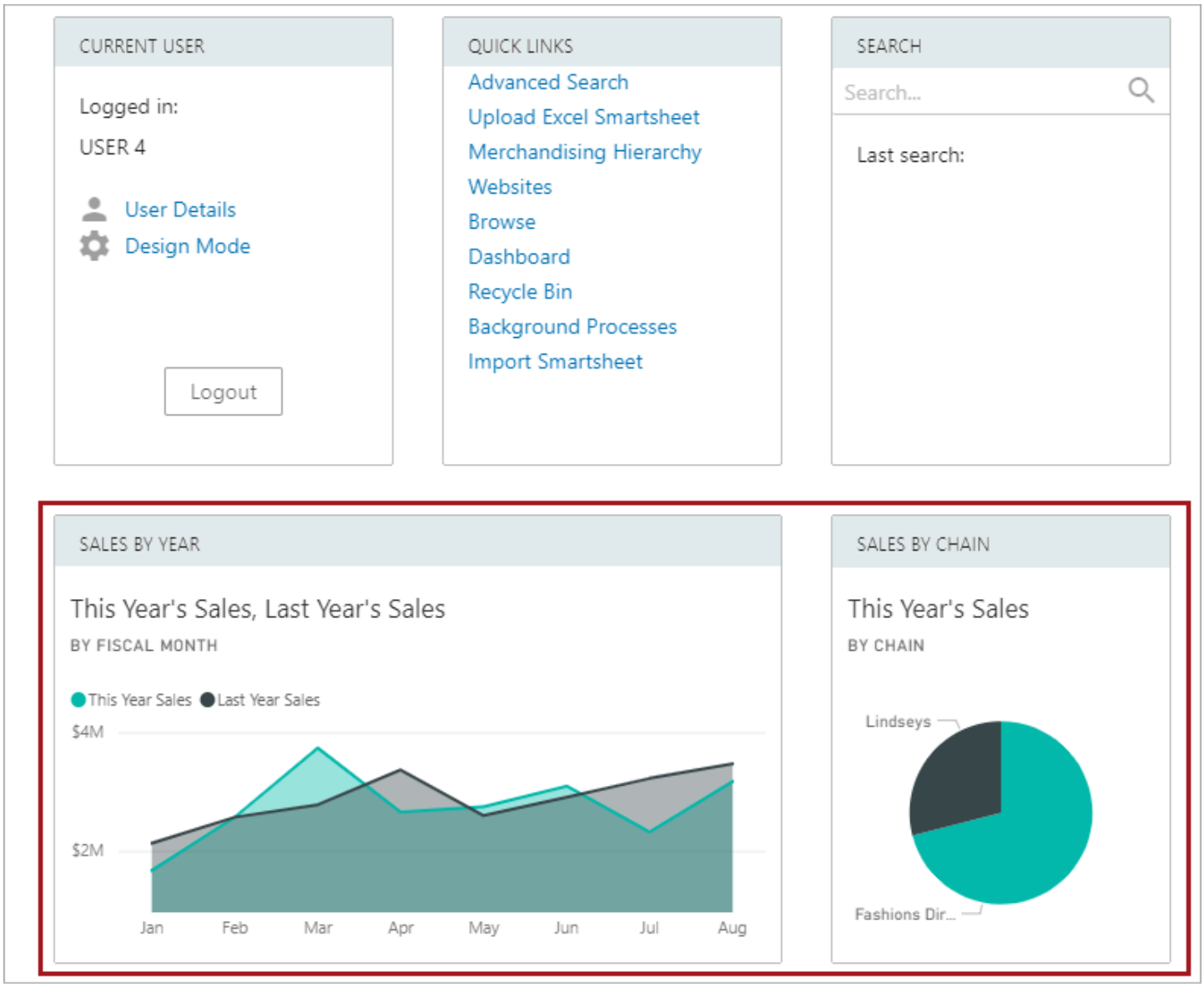
8. **Page Navigation Panel:** Check this box to display a page navigation panel on the bottom of the Web UI screen. This allows you to easily navigate between different pages of the Power BI report. An example is shown in the first screenshot in this topic.

Power BI Web UI Widget

The visual integration with Power BI uses two components to display analytics data in the Web UI: the Power BI Widget and the Power BI Screen. This topic details the configuration of the Power BI Widget.

The Power BI Widget is used to provide high-level summary information of Power BI analytics information made available to users as widgets on the Web UI homepage. Most commonly, Power BI dashboard tiles are embedded into widgets, though Power BI reports can also be embedded into Power BI widgets.

The below screenshot shows a Web UI homepage with two sample Power BI Widgets at the bottom, each of which is configured to show a Power BI dashboard **tile**. The Power BI Widget is very similar to the Power BI Screen except it is designed to display smaller amounts of high-level summary data.



Configuring the Power BI Widget

The Power BI Widget provides users with a quick and simple view of specified analytics data on the Web UI's homepage. The appearance and behavior of the widget is determined by the Power BI dashboard tile or report that is embedded within it, which is preconfigured in Power BI. Analytics homepage widgets can be single-width or double-width, and multiple homepage widgets can be configured to display multiple views of analytics dashboards.

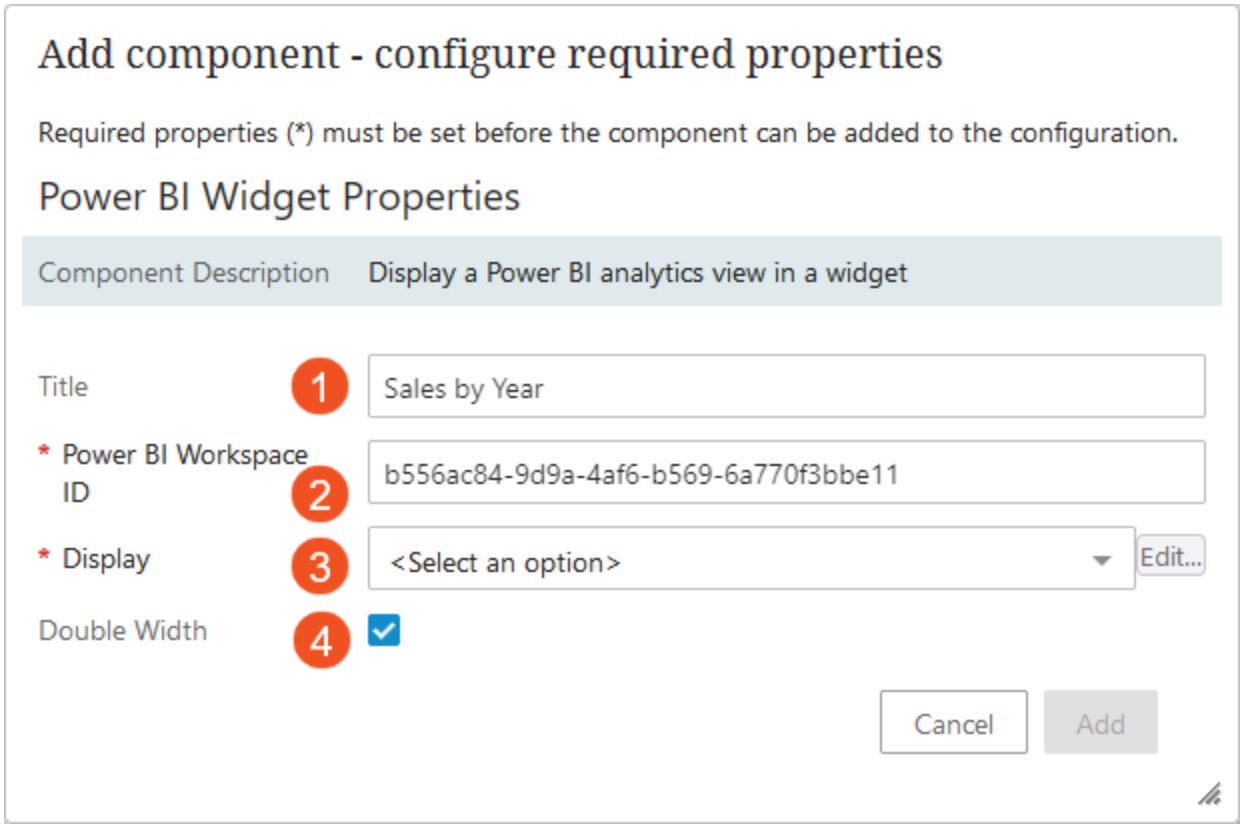
Adding the Power BI Widget

The Power BI Widget must be added to the homepage prior to configuration. Details on how to do this can be found in the **Adding Widgets to a Homepage** topic in the **Web User Interfaces** documentation.

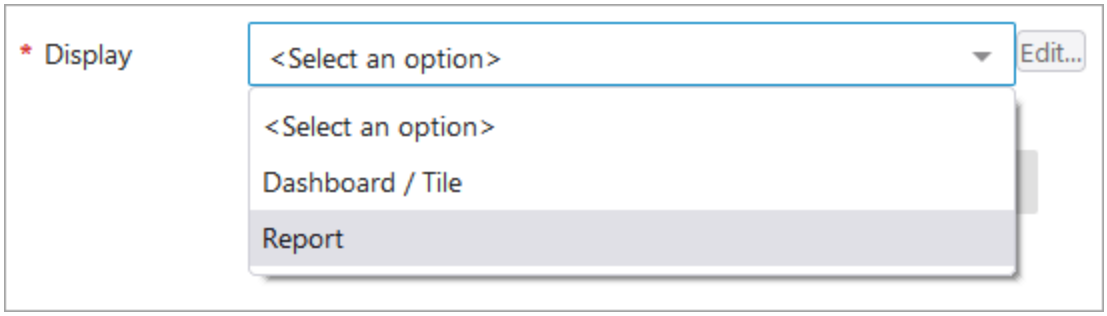
Setting up the Power BI Widget

The Properties window for the Power BI Widget in the Web UI designer is shown below. The parameters that appear in the screenshot are described in detail directly beneath the screenshot.

The Power BI Screen and Power BI Widget are configured in almost identical ways, with some minor exceptions. The configuration steps described below can be applied to configuring either the widget or the screen, except where expressly noted.



1. **Title:** Content added to this field appears in the top left corner of the widget. If no title is set, the default title of 'Analytics' will display.
2. **Power BI Workspace ID:** Enter the ID of the relevant Power BI Workspace. For example, b556ac84-9d9a-4af6-b569-6a770f3bbe11. For information on how to obtain this value, refer to the **Locating Power BI IDs** topic.
3. **Display:** Select either 'Dashboard / Tile' or 'Report.' The most common selection for the Power BI Widget will be Dashboard / Tile, though reports can also be embedded in a Power BI widget if desired.

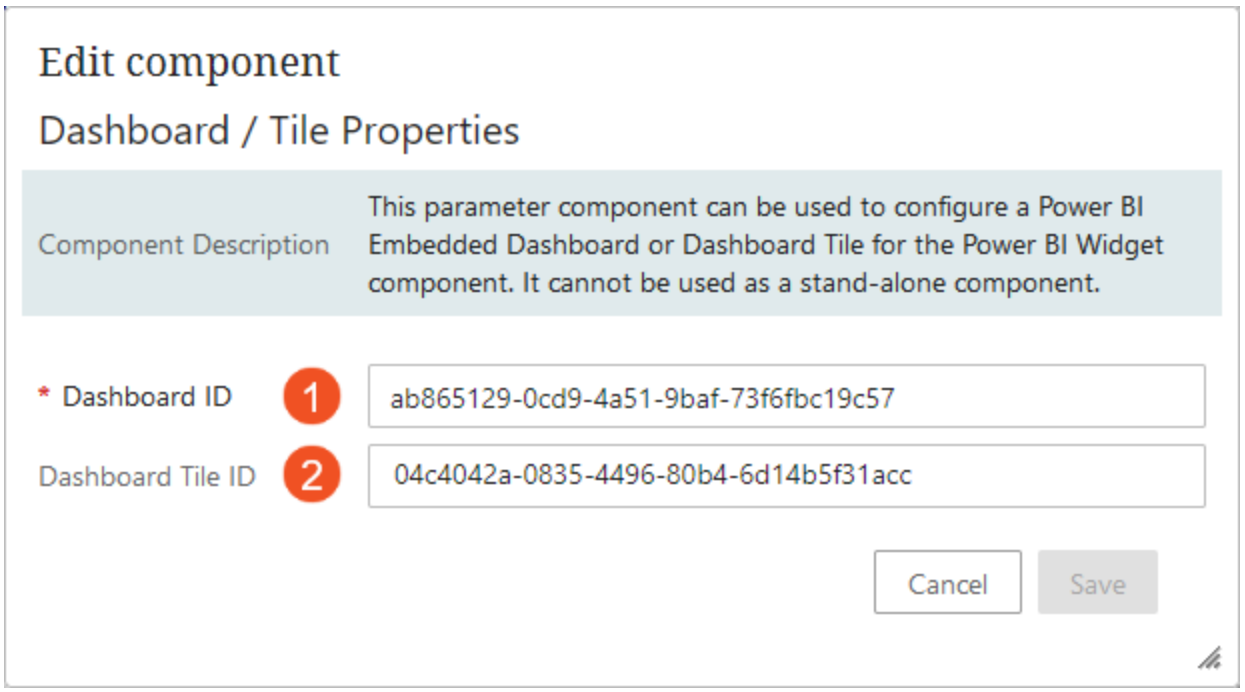


4. **Double Width:** If this parameter is checked, the widget is doubled in width from the standard widget's single-width size.

Dashboard / Tile Configuration

If you are configuring your Power BI widget to display a **tile**, select Dashboard / Tile from the Display dropdown, then click the 'Edit...' button to the right of the Display field to open the **Dashboard / Tile Properties** designer window.

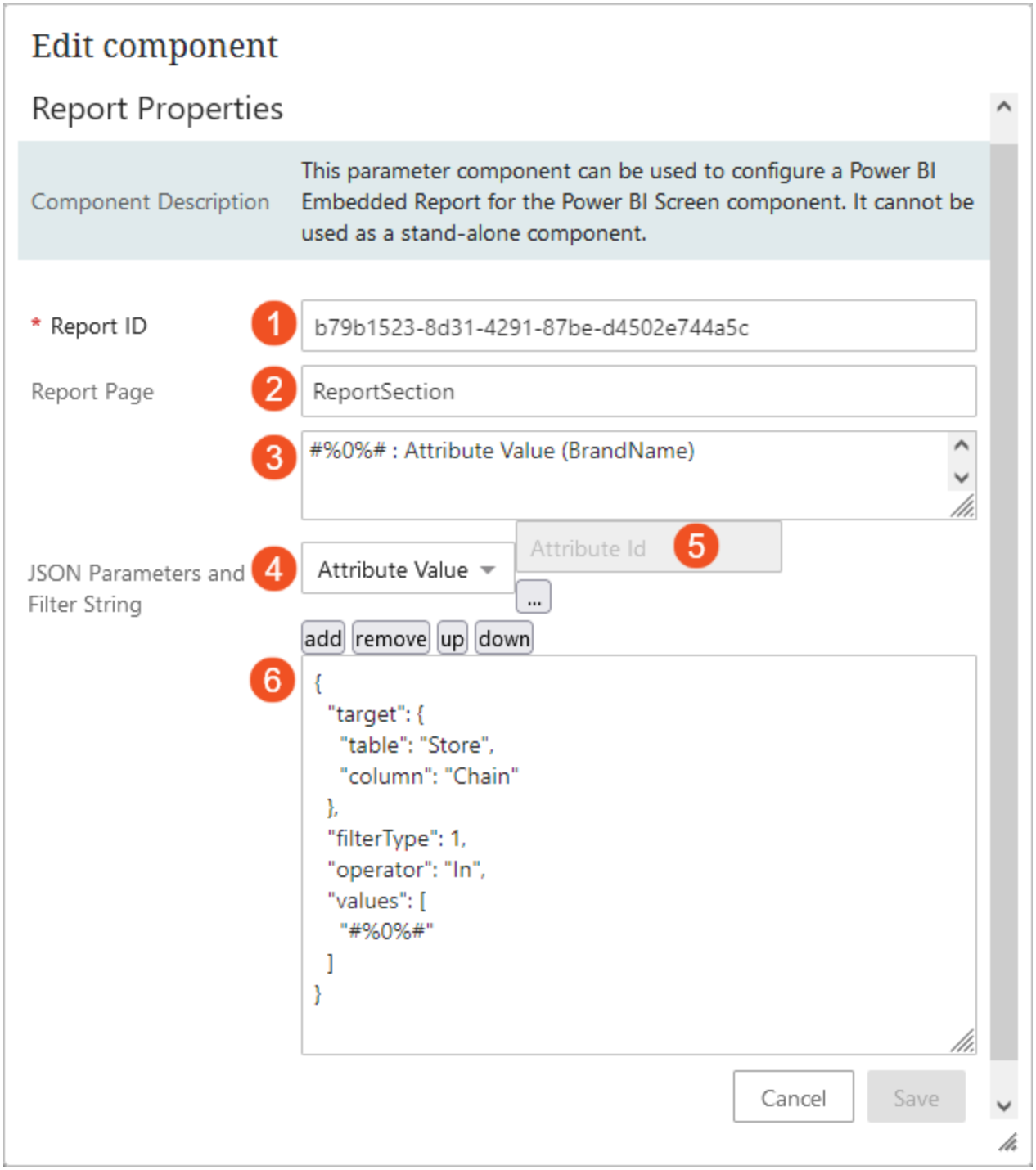
Note: It is not recommended to embed dashboards into Power BI widgets due to the limited space available in the widget.



1. **Dashboard ID:** Enter the ID of the relevant Power BI dashboard. For example, 4c3d6f97-7dc8-4789-a911-5f04daf9de5f.
2. **Dashboard Tile ID:** Enter the ID of the relevant Power BI dashboard tile. This should always be entered for Power BI widgets, since tiles are what widgets are intended to display.

Report Configuration

If you are configuring your Power BI widget to display a **report**, select Report from the Display dropdown, then click the 'Edit...' button to the right of the Display field to open the **Report Properties** designer window. Reports in Power BI widgets are configured in a near-identical fashion to those for Power BI screens, except widgets cannot have a Filter Panel or Page Navigation Panel. Filtering is available through JSON filter strings and/or by row level security, but end-users cannot further filter the information that displays.



1. **Report ID:** Enter the ID of the relevant Power BI report. For information on how to obtain this value, refer to the **Locating Power BI IDs** topic.
2. **Report Page:** Enter the name of the Power BI report page that you want to display when the report loads in the widget.

Filter Configuration Options: JSON Parameters and Filter String

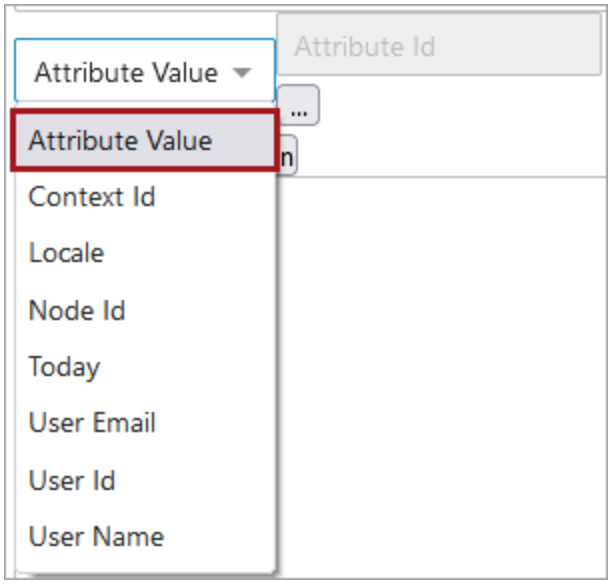
This section contains two fields. The top field (3) is for JSON filter parameters. The bottom field (6) is where the JSON filter string is entered. Use these options to pre-configure the information that displays in the report. Though filter panels are not available for widgets, the JSON strings applied in the Web UI designer can be used to refine data that is available to the logged-in user. For more information on using JSON strings to filter Power BI reports, refer to the **Example Power BI Report Filters** topic in this guide.

Note: To filter data based on user permissions, e.g., to ensure that the logged-in user only accesses data that is relevant to them, you must use row-level security (RLS). For more information, refer to the **Power BI Row-Level Security** topic in this guide.

3. **Filter parameters** *(field not individually labeled)* – In this field, enter the relevant filter parameters, e.g., 'Attribute Value.' Each selected parameter will be assigned a placeholder string that will be replaced with a context-specific value to be referenced in the JSON query string. These placeholders also contain an embedded integer that identifies the sequence in which the parameter will appear in the JSON filter string, e.g., #%0%# will appear first, #%1%# will appear second, etc. Moving a parameter up or down with the 'up' or 'down' button will automatically renumber the placeholder.

   Note: It is recommended to generate the placeholders for the filter parameters before entering the JSON filter string in the bottom field.

4. The dropdown menu below the filter parameters field allows the selection of various options to create the filter parameters. The available parameters are as follows:



- **Attribute Value:** Returns / filters the view for a STEP attribute value. *Valid for the Power BI Screen only.*
- **Context Id:** Returns the current context ID
- **Locale**: Returns the selected locale, which is used for the Web UI session

- **Node Id:** Returns the STEP ID of the current object *(valid for Power BI Screen only)*
- **Today:** Returns the current date
- **User Email:** Returns the email address of the logged-in user (if specified)
- **User Id:** Returns the user ID of the logged-in user
- **User Name:** Returns the name of the logged-in user

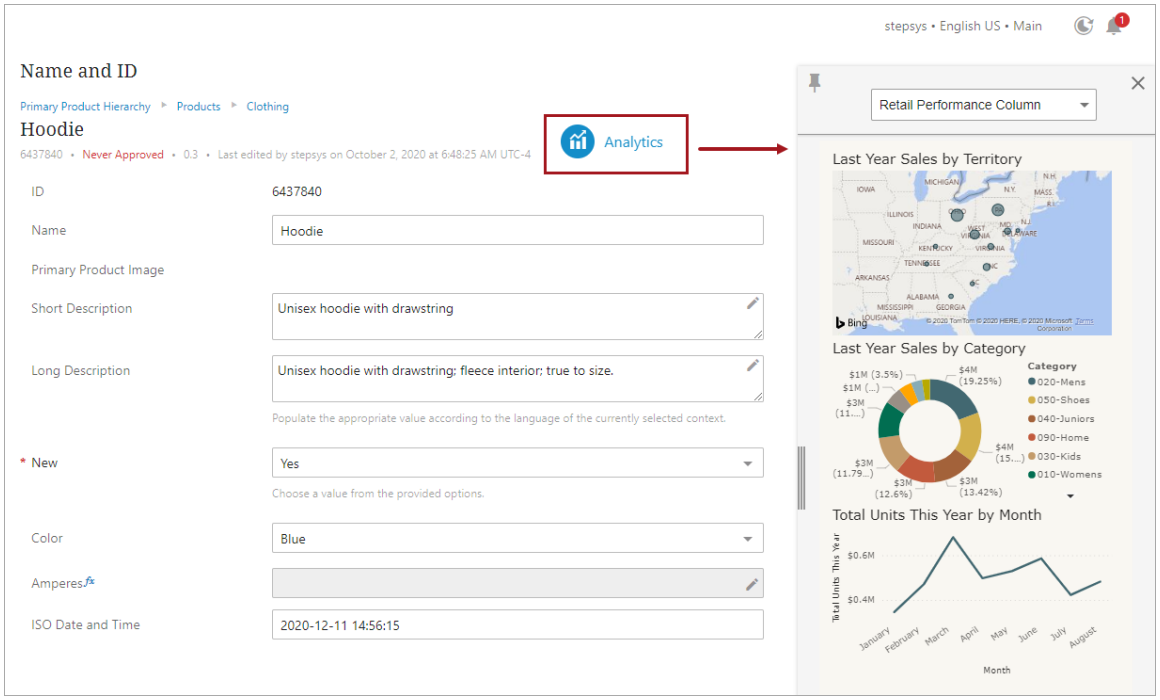
5. The field directly to the right of the dropdown menu will display when either 'Attribute Value' or 'Today' is selected. If **Attribute Value** is selected, the ellipsis button displays (...). Clicking this button opens the 'Select Node(s)' window, where the relevant attribute is chosen. If **Today** is selected, a 'Date Format' window displays, in which the chosen date format is entered, e.g., yyyy-MM-dd. The date format uses the rules for SimpleDateFormat in Java.
6. **JSON filter string** *(field not individually labeled)* – In this field, enter the desired JSON filter string. It is recommended that the string contain placeholders that correspond with the numbered dynamic values selected for the JSON filter parameters, e.g., #%0%#. #%1%#, etc.

Power BI Flyout Panel

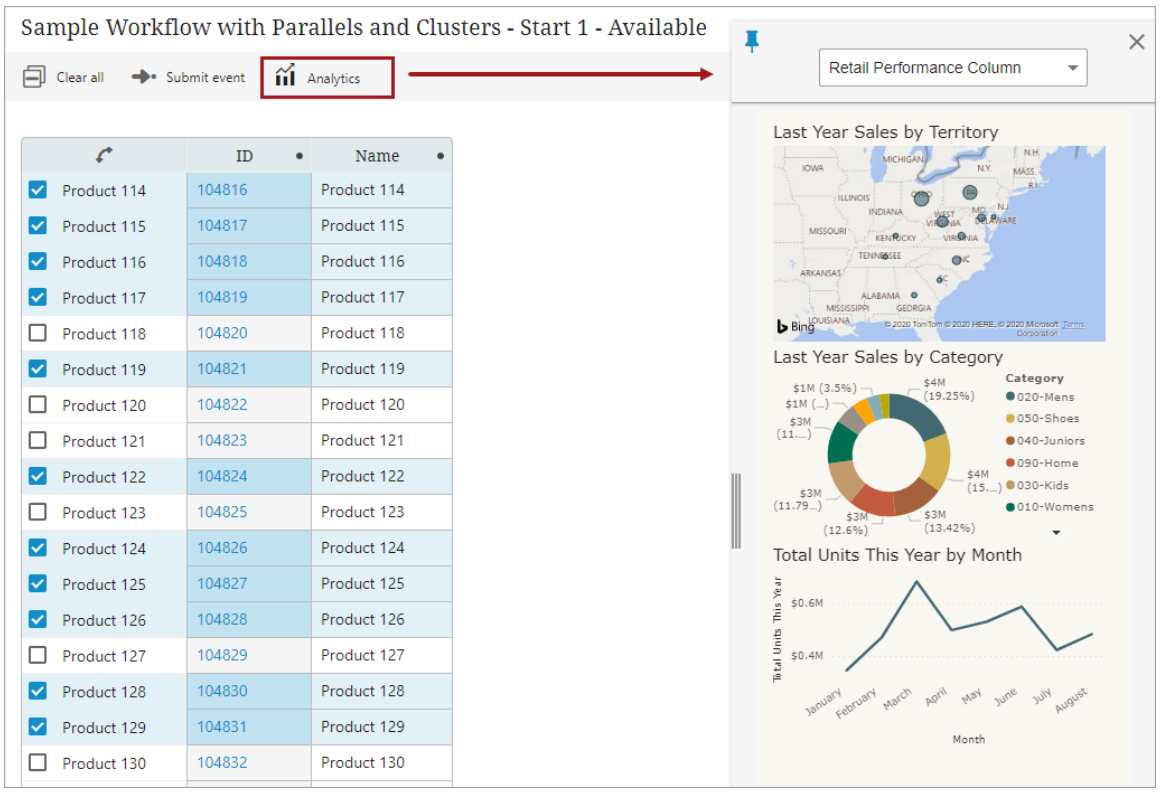
The flyout panel is available to be added to the Node Details screen and Node List components in the Web UI, giving you the ability to view analytics information in a conceptualized manner for both single nodes and multiple nodes. The flyout panel displays Power BI reports and provides the ability to toggle between multiple reports. By presenting analytics information in context with the content being analyzed, the Power BI flyout panel provides users a seamless, contextual interface to help in their decision-making processes.

Displaying and Using the Flyout Panel

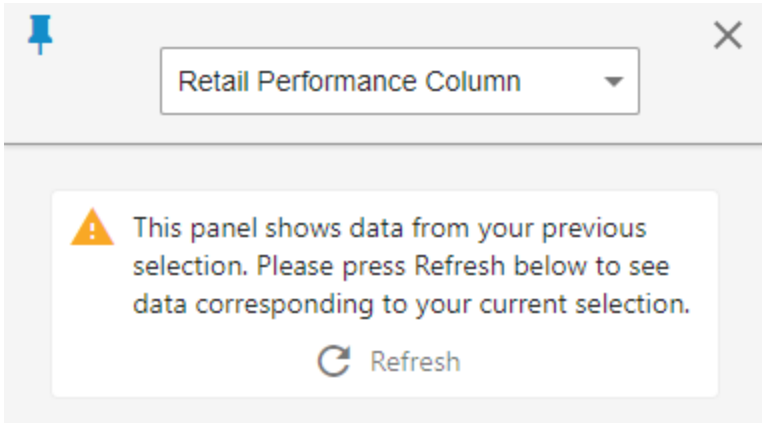
Node Details: When the flyout panel is configured on a Node Detail screen, navigate to an object and click the Analytics button in the 'Below Title' component (top area) of the screen. The panel will display on the right.



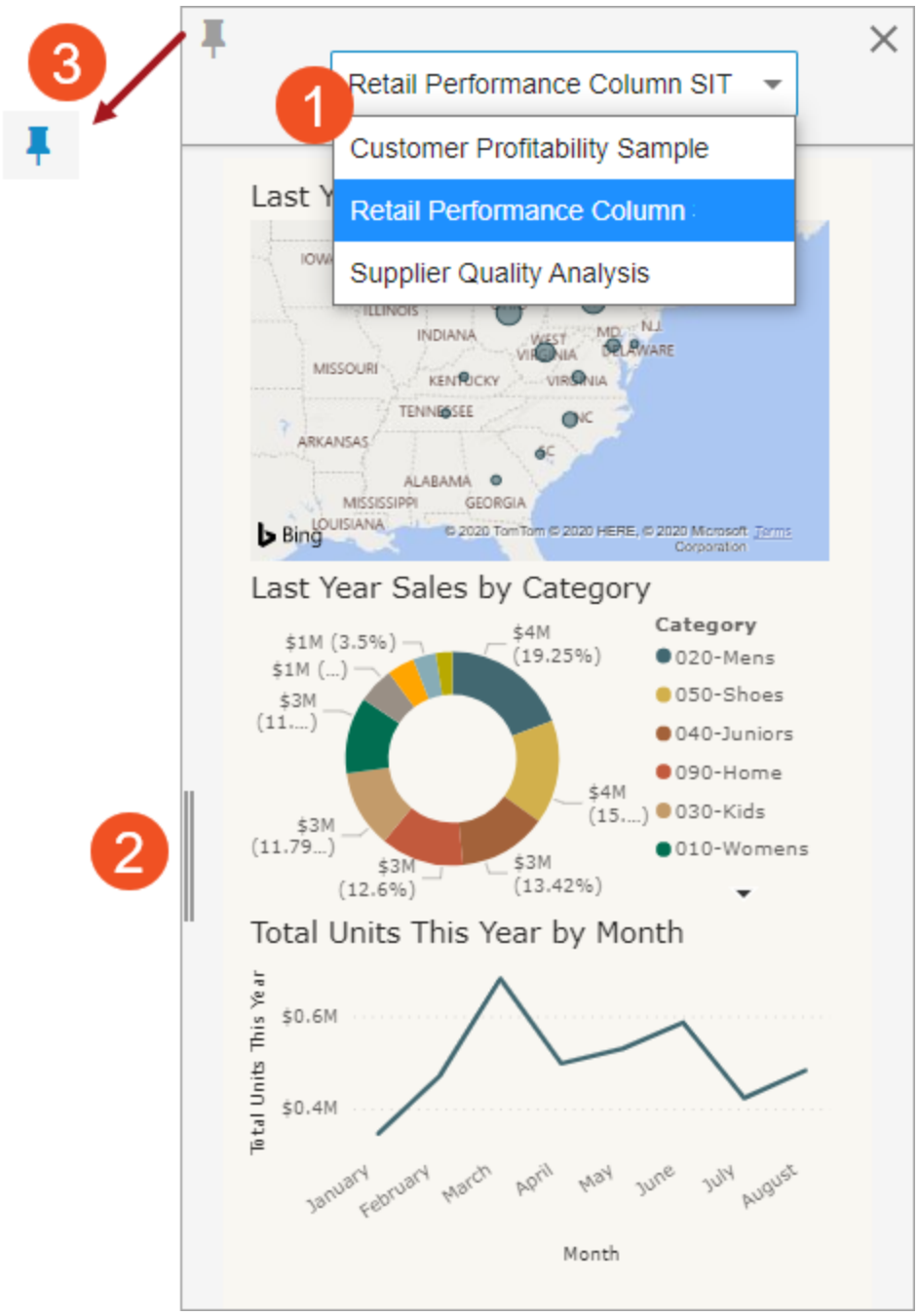
Node List: If you access the flyout panel from a Node List, where the Analytics action has been configured to display in the toolbar, you simply make a single or multiple object selection and then click the Analytics button.



If you have the flyout panel pinned and select another object from the Node List, the panel will display a message giving you the option to refresh.



Features of the flyout panel include:



1. A dropdown list displays at the top of the panel, which allows you to toggle between different reports.
2. The panel is resizeable; you can drag the handle on the left side of the panel and drag it to the left across the screen to view larger widgets. The maximum height is 400 px; widgets start with a height of 255 px. Most reports grow larger / automatically resize as you expand the panel; however, all do not, such as prices and indicator widgets.
3. If you click outside the flyout panel and the panel is not pinned, it will retract. If you want the panel to remain open as you update the object, click the pin icon and the panel will remain open. The pin is gray when unpinned, and it turns blue when pinned.

The Web UI will remember your preferences if you navigate away from the page then come back and when you exit your Web UI instance and log back in. The report you were last on will display each visit until you change it, and the system will remember if you pin the flyout panel and/or if you resized it.

If you click another 'global' flyout panel, such as alerts or BGP, the analytics panel will be replaced by the other flyout until the Analytics button is clicked again, even if the Power BI flyout panel is pinned.

Configuring the Flyout Panel

Prerequisites

It is expected that anyone configuring the component is familiar with the Web UI Design Mode as basic concepts for working with the designer are not covered in this section. In addition, the user must have appropriate privileges to access the designer. Additional information can be found in the **Designer Access** section of the **Web User Interfaces** documentation.

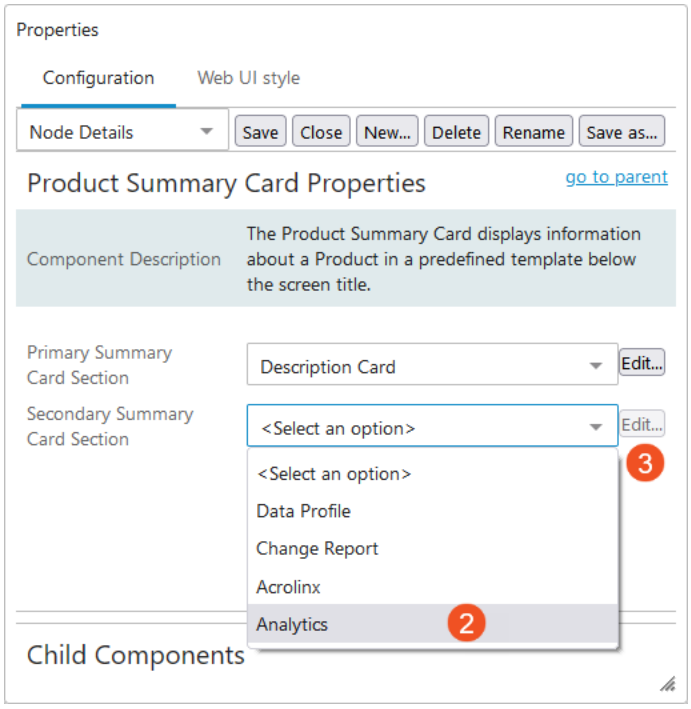
Adding to a Node Details screen

The panel supports product, classification, and entity objects and is added to Node Details screens in Web UI as a 'Below Title' summary card child component (either as a Product Summary Card, Classification Summary Card, or Entity Summary Card).

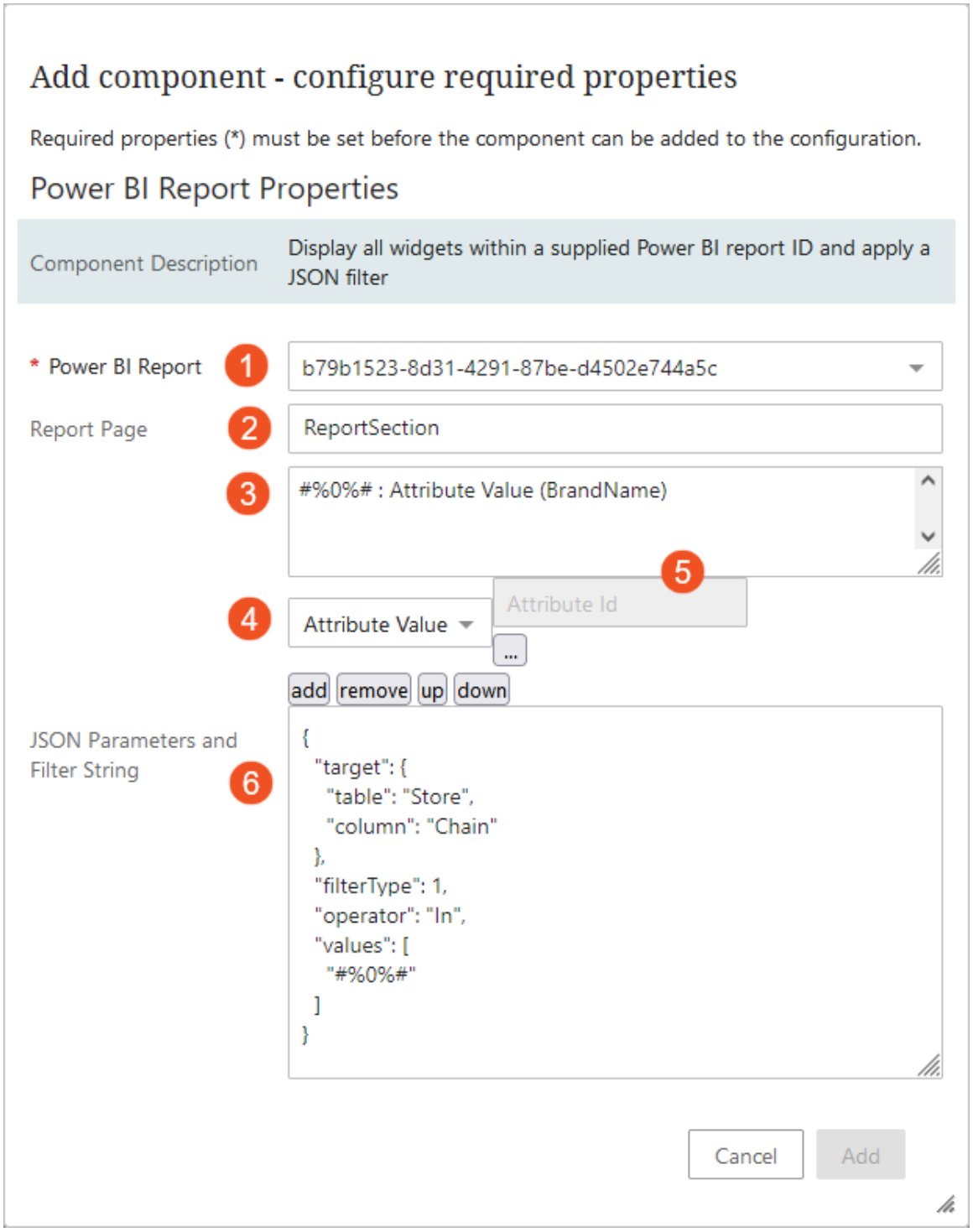
To configure this panel, in the Node Details component, add the applicable Summary Card component as a child component to the Below Title component. For general configuration details about these Summary Cards, refer to the **Below Title Component** section in the **Node Details Screen** in the **Web User Interfaces** documentation.

For the steps below, the Product Summary Card component will be used to outline the Power BI flyout panel configuration. The steps will be similar for the Entity Summary Card and Classification Summary Card.

1. Within Node Details Properties, add the 'Product Summary Card' as a child component to the Below Title component, then click 'go to component.'
2. On Product Summary Card Properties, select **Analytics** from the Secondary Summary Card Section dropdown list. (This option will only display if the Analytics commercial license is activated for your system,)



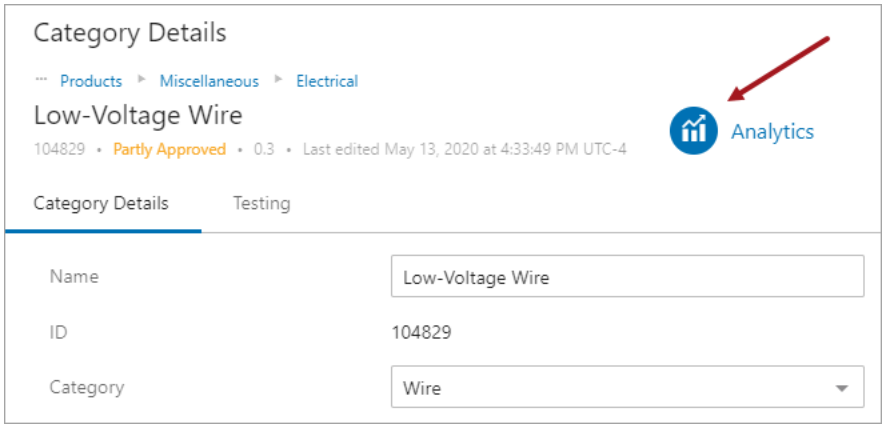
3. The Analytics Properties will open automatically with new setups. If making changes to an existing setup, click Edit... to open the Analytics Properties.
4. In Analytics Properties, add a Power BI Report ID.
5. For Report Page, enter the name of the Power BI report page that you want to display when the report loads in the flyout panel.
6. Select / configure the JSON Parameters and Filter String parameters. For example:



For more information on using JSON strings to filter Power BI reports, refer to the **Example Power BI Report Filters** topic. You can also refer to the 'Filter Configuration Options: JSON Parameters and Filter String' section in either the **Power BI Web UI Screen** topic or **Power BI Web UI Widget** topic.

7. Click **Save** to complete the configuration.
8. Repeat steps 4 through 7 to add additional reports.
9. Click **Save** in Analytics Properties to save the settings.
10. Click **Save** and then **Close** to exit Web UI Design Mode.

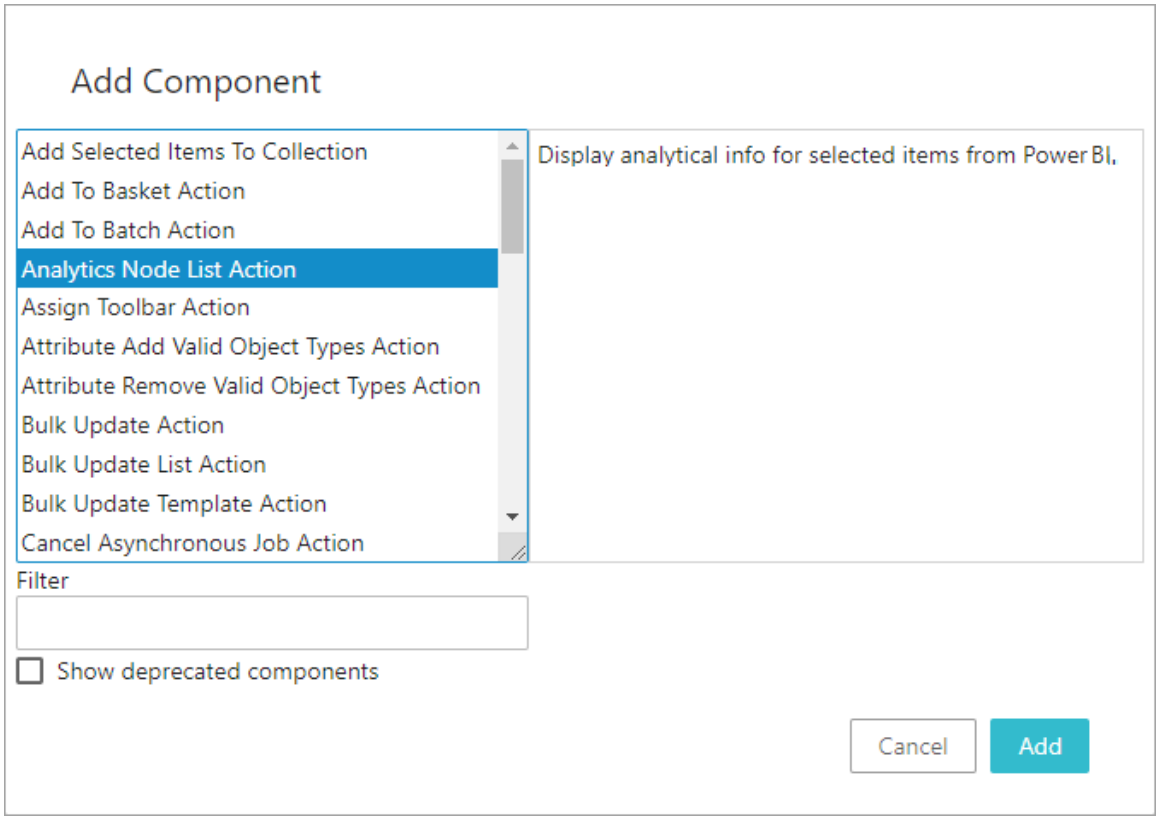
A clickable Analytics button now displays below the title.



Adding to a Node List component

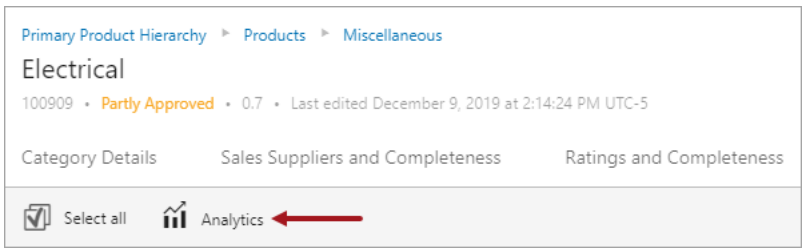
The Analytics button can also be added to the Node List component as a child Analytics Node List Action. Adding this action will display the action button in the toolbar. Users can then click the button to view a Power BI flyout panel for the selected node(s).

1. Within Node List Properties, go to Child Components. Click **Add** under the Actions field.



2. In Analytics Node List Action Properties, add a Report ID.
3. Enter a Report Page.
4. Select / configure the JSON Parameters and Filter String parameters. Refer to the section above for an example, and refer to the **Example Power BI Report Filters** topic.
5. Click **Save** to complete the configuration.
6. Repeat steps 4 through 7 to add additional reports.
7. Click **Save** in Analytics Node List Action Properties to save the settings.
8. Click **Save** and then **Close** to exit Web UI Design Mode.

A clickable Analytics action button now displays in the toolbar.



Locating Power BI IDs

Power BI IDs are required when configuring the Power BI Screen and Power BI Widget components in the Web UI. These IDs are needed to link to the relevant reports, dashboards, and tiles in Power BI. Five types of IDs are available:

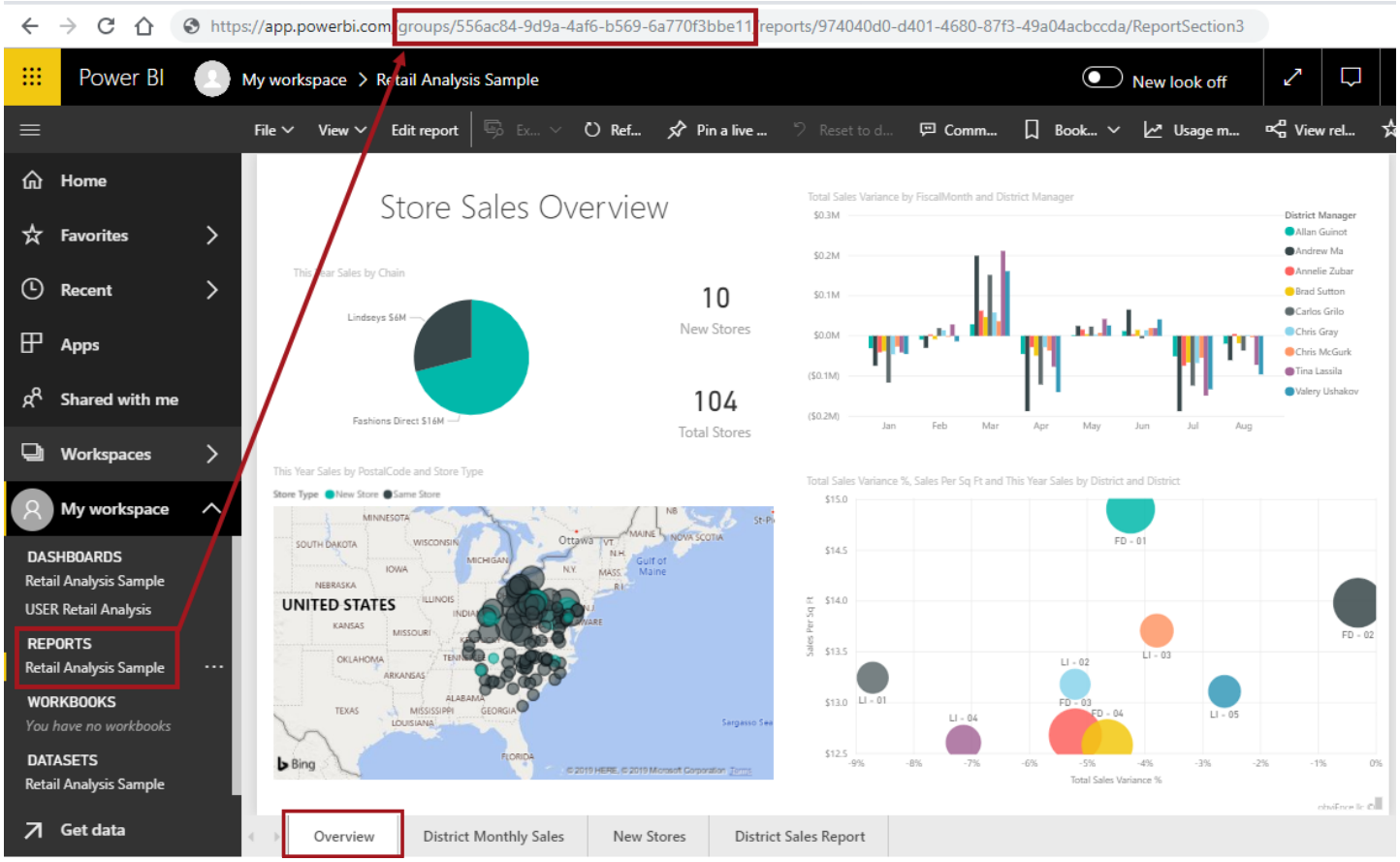
- Workspace ID
- Report ID
- Report Page ID
- Dashboard ID
- Dashboard Tile ID

This topic explains how to locate these IDs in the URLs that are accessed from within the Power BI web application. You must be logged into Power BI (https://app.powerbi.com) to access these URLs.

Report and Report Page IDs

To configure either the Power BI Screen or Power BI Widget, you need the ID of the report, dashboard, or dashboard tile to embed.

- The **Workspace ID** can be copied from the URL of a report or dashboard when viewed from within Power BI. The Workspace ID is defined in the **groups** section of the URL. For example, https://app.powerbi.com/groups/**556ac84-9d9a-4af6-b569-6a770f3bbe11**/reports/974040d0-d401-4680-87f3-49a04acbccda/ReportSection3.

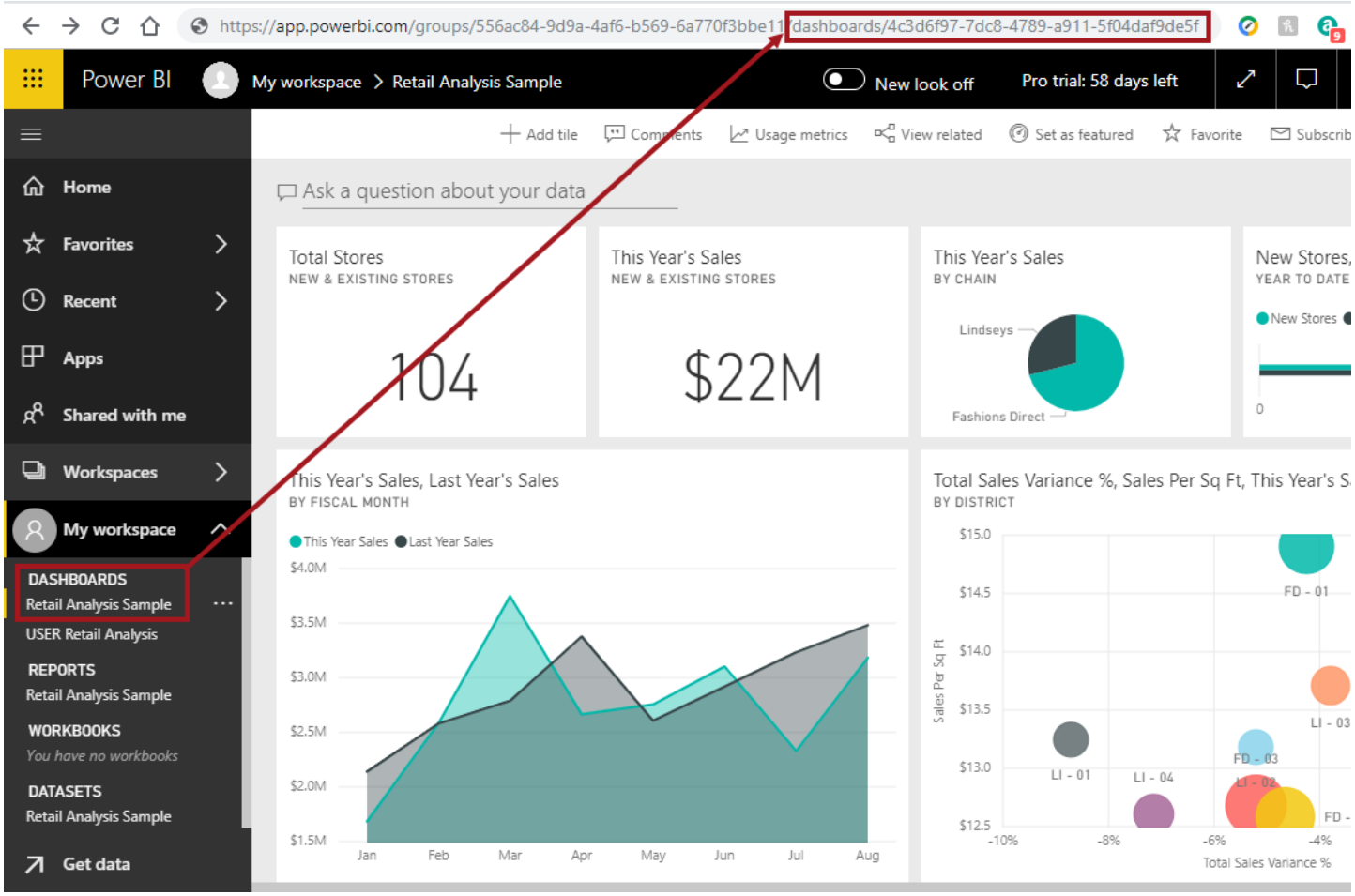


- The **Report ID** is available by selecting the relevant report when viewed in Power BI and inspecting the URL. For example, in the above URL, the report ID is https://app.powerbi.com/groups/556ac84-9d9a-4af6-b569-6a770f3bbe11/reports/**974040d0-d401-4680-87f3-49a04acbccda**/ReportSection3.
- The **Report Page** is the page of the report that will show when the report is first loaded. For example, https://app.powerbi.com/groups/556ac84-9d9a-4af6-b569-6a770f3bbe11/reports/974040d0-d401-4680-87f3-49a04acbccda/**ReportSection3**

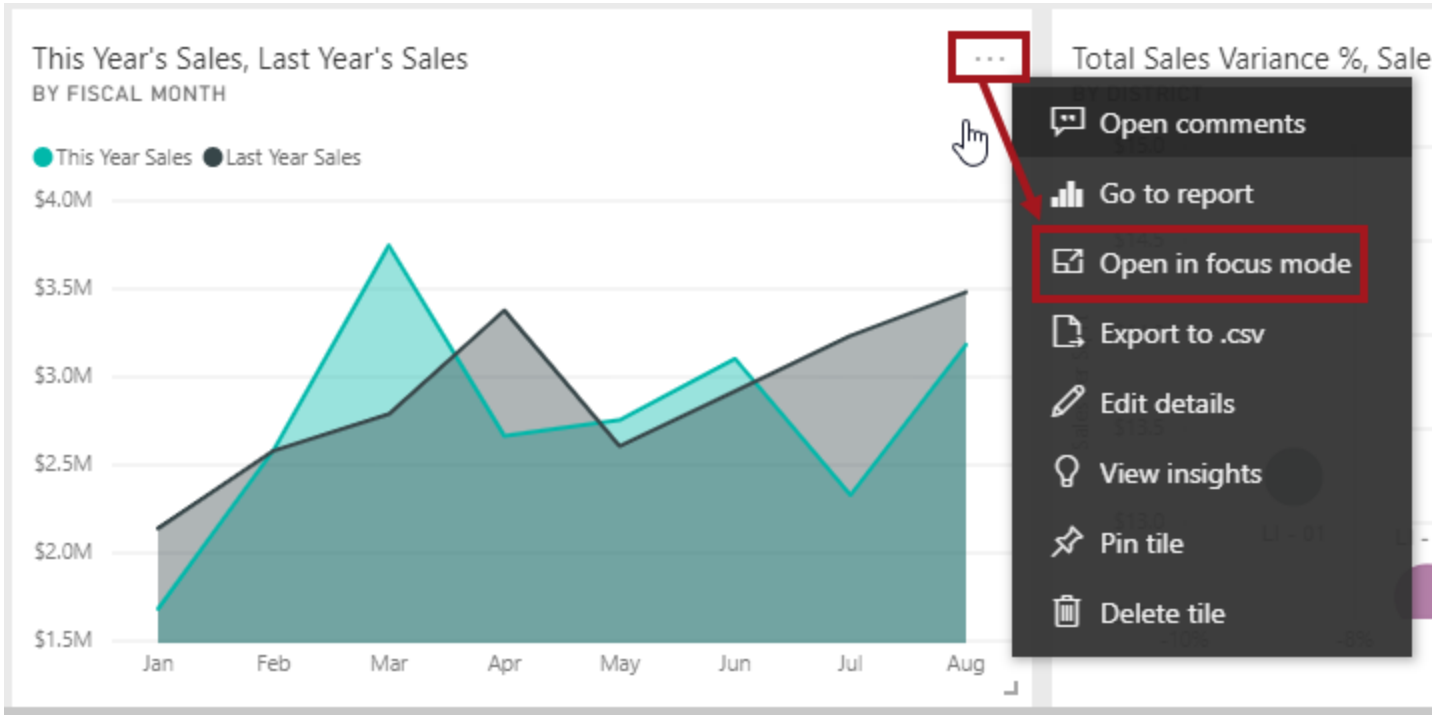
  In the above screenshot, the Overview tab is the report page. Other report pages can be viewed by clicking on another tab (e.g., District Monthly Sales or New Stores). When another page is selected, the URL will then show the page ID of the selected page.

Dashboard and Dashboard Tile IDs

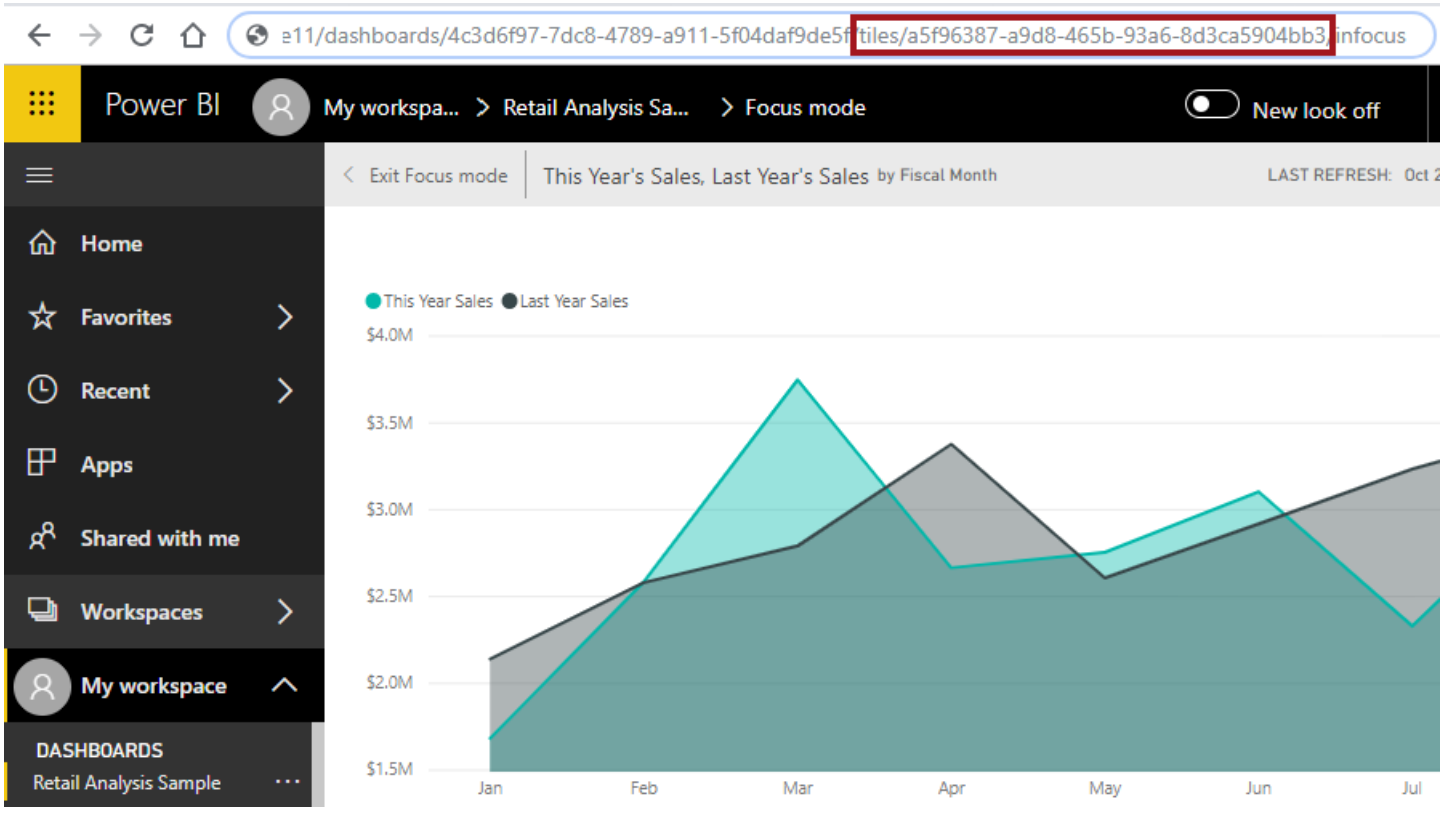
- The **Dashboard ID** is available by selecting the relevant dashboard when viewed in Power BI and inspecting the URL. For example: https://app.powerbi.com/groups/556ac84-9d9a-4af6-b569-6a770f3bbe11/dashboards/**4c3d6f97-7dc8-4789-a911-5f04daf9de5f**.



- The **Dashboard Tile ID** is available by selecting the relevant dashboard when viewed in Power BI, then clicking the ellipsis (...) that appears when hovering over a dashboard tile and selecting 'Open in Focus Mode.'



The tile opens in Focus mode, and the ID of the tile is available in the URL. For example, https://app.powerbi.com/groups/556ac84-9d9a-4af6-b569-6a770f3bbe11/dashboards/4c3d6f97-7dc8-4789-a911-5f04daf9de5f/tiles/**a5f96387-a9d8-465b-93a6-8d3ca5904bb3**/infocus



Example Power BI Report Filters

Power BI supports three different levels of filters that can be applied to reports: report level, page level, and visual level. To simplify the Web UI configuration, only **report level** filters are supported, which are filters that apply to the entire report. Power BI filters information on a dataset, and the filters that are passed from STEP to Power BI at report load time are in **JSON** format.

Note: Filter panels are only applicable to **reports**; dashboards and tiles do not support filter panels.

On Power BI screens and widgets that display Power BI reports, JSON filters allow you to specify additional filters beyond the standard filters configured in Power BI. On **screens** that show reports, the filters can be shown in the optional Power BI 'Filters' pane on the right side of the page.

Note: To filter data based on user permissions, e.g., to ensure that the logged-in user only accesses data that is relevant to them, you must use row-level security (RLS). For more information, refer to the **Power BI Row-Level Security** topic in this guide.

This topic describes several examples of JSON-formatted filter strings that can be placed into the 'JSON parameters and Filter String' field in the Web UI designer when configuring a report display for a Power BI screen. For more in-depth details on configuring report filters for Power BI, refer to the following Microsoft help documentation: https://github.com/Microsoft/PowerBI-JavaScript/wiki/Filters.

Example Filters

Report-level filters support the following types: Basic Filter, Advanced Filter, and Relative Date Filter. The exact format for filters is defined by Microsoft. For more information, refer to: https://microsoft.github.io/powerbi-models/interfaces/_models_.ifilter.html.

When **no** JSON filter is specified in the Web UI screen configuration, the Power BI filter pane appears as in the following screenshot. The filter parameters shown in the filter pane are only those that are configured on the Report within the Power BI application.

When JSON filter panes **are** specified in the Web UI screen configuration, additional Power BI report-level filter parameters are added to the filter panel with the relevant value(s) preselected.

Note: Users can deselect the values and show the unfiltered results.

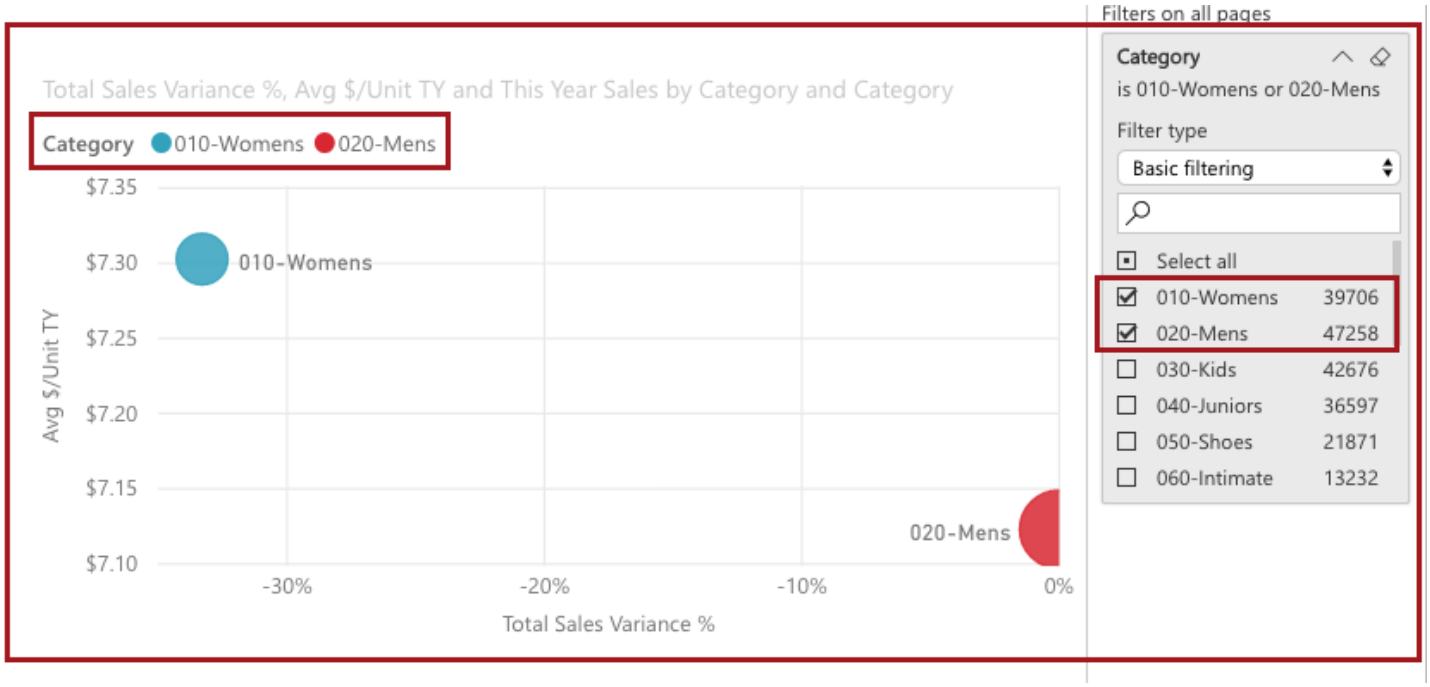
Basic Filter

The following string creates a simple Power BI filter that only includes data if the value in the 'Item' table / 'Category' column is either 010-Womens or 020-Mens.

```
{
  "target": {
    "table": "Item",
    "column": "Category"
  },
  "filterType": 1,
  "operator": "In",
  "values": [
    "010-Womens", "020-Mens"
```

```
  ]
}
```

As shown in the following screenshot, the string has created a 'Category' filter pane with basic filtering that automatically filters data on the 010-Womens and 020-Mens categories.



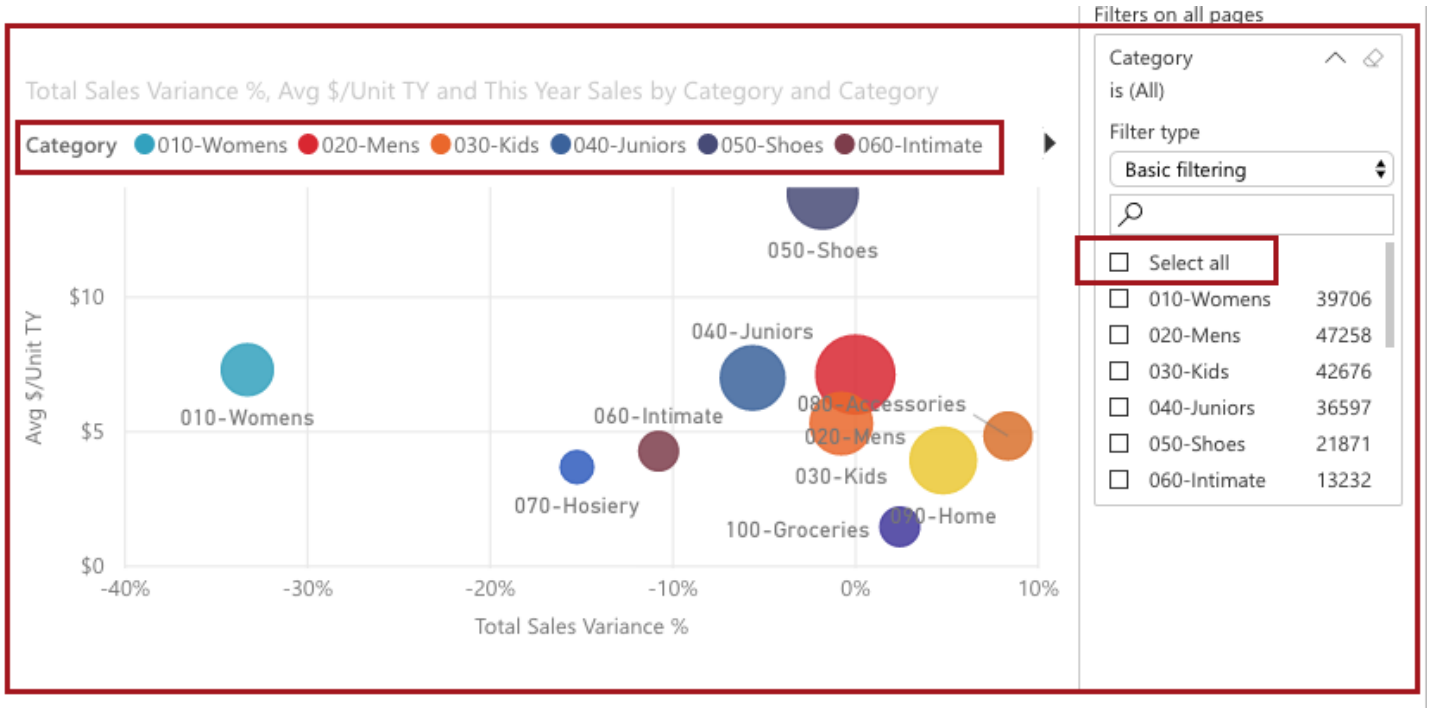
Basic Filter - Select All

The following basic filter string uses the 'All' operator, which will cause all entries in the 'Item' table / 'Category' column to contribute data to the report. It is effectively a 'Select All' filter.

```
{
  "target": {
    "table": "Item",
    "column": "Category"
  },
 "filterType": 1,
 "operator": "All",
 "values": []
}
```

In the filter panel, data from all entries are displaying.

Note: In Power BI, the 'Select All' button is not activated when using this filter.

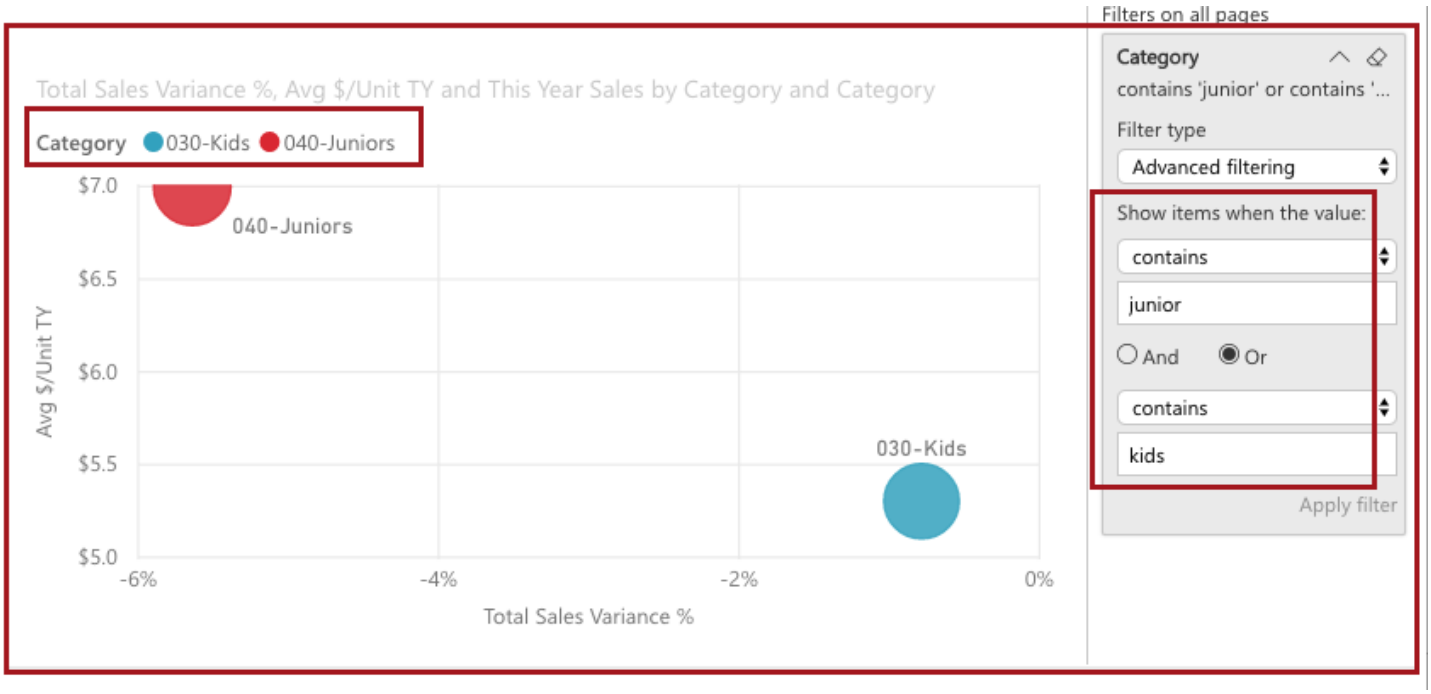


Advanced Filter

An advanced filter can contain operators and conditions. The below string creates a Power BI filter pane with advanced filtering that only includes data if the value in the 'Item' table / 'Category' column contains either 'junior' or 'kids.'

```
{
  "target": {
    "table": "Item",
    "column": "Category"
  },
  "logicalOperator": "Or",
  "conditions": [
    {
      "operator": "Contains",
      "value": "junior"
    },
    {
      "operator": "Contains",
      "value": "kids"
    }
  ],
  "filterType": 0
}
```

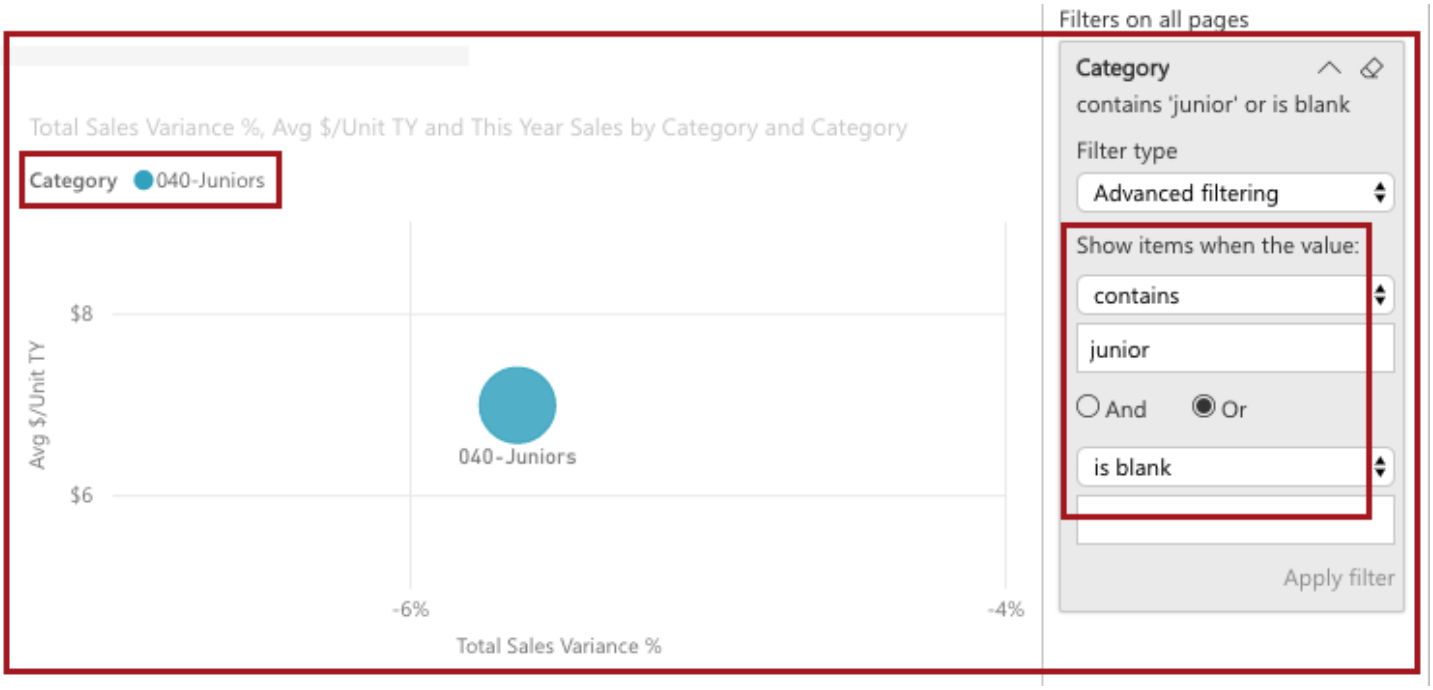
In the report sample shown below, this will match the values 030-Kids and 040-Juniors.



Advanced Filter - Is Blank

The following string creates a similar Power BI filter as the previous example, except the filter only includes data if the value in the 'Item' table / 'Category' column contains either 'junior' or is blank.

```
{
  "target": {
    "table": "Item",
    "column": "Category"
  },
  "logicalOperator": "Or",
  "conditions": [
    {
      "operator": "Contains",
      "value": "junior"
    },
    {
      "operator": "IsBlank",
      "value": ""
    }
  ],
  "filterType": 0
}
```



Relative Date Filter

A relative date range is a period of time that is relative to the current date, e.g., last week, last month, or last year. The following string creates a Power BI filter pane that includes data if the value in the 'Store' table / 'OpenDate' column contains a value within the last 6 years.

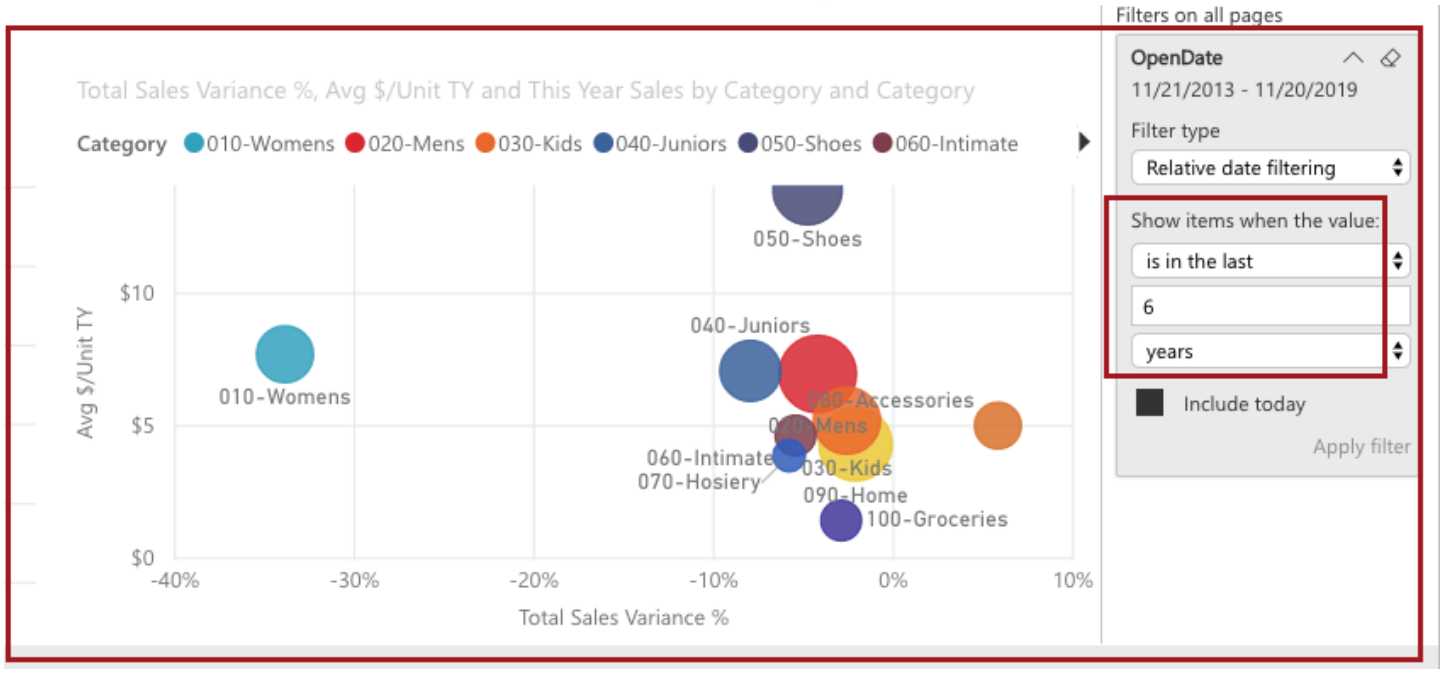
```
{
  "target": {
    "table": "Store",
    "column": "OpenDate"
  },
  "operator": 0,
  "timeUnitsCount": 6,
  "timeUnitType": 5,
  "includeToday": true,
  "filterType": 4
}
```

Explanation of values:

- The **operator** field is set to **0**, which is RelativeDateOperators.InLast (refer to https://microsoft.github.io/PowerBI-JavaScript/enums/_node_modules_powerbi_models_dist_models_d_.relativedateoperators.html for more information)
- The **timeUnitType** of **5** is RelativeDateFilterTimeUnit.Years (refer to https://microsoft.github.io/PowerBI-JavaScript/enums/_node_modules_powerbi_models_dist_models_d_.relativedatefiltertimeunit.html for more information)

- The **filterType** of **4** is FilterType.RelativeDate (refer to https://microsoft.github.io/PowerBI-JavaScript/enums/_node_modules_powerbi_models_dist_models_d_.filtertype.html for more information)

The filter displays as follows in the Web UI:



Example Web UI Report Filters

JSON filter strings like those described in this topic are placed into the 'JSON parameters and Filter String' field in the Web UI designer when configuring a Power BI screen or widget. One or more filters can be specified at the same time within this field. Multiple filters are defined as an **array** of filters. For additional information on configuring the Power BI Screen in the Web UI designer, refer to the **Power BI Web UI Configuration** topic.

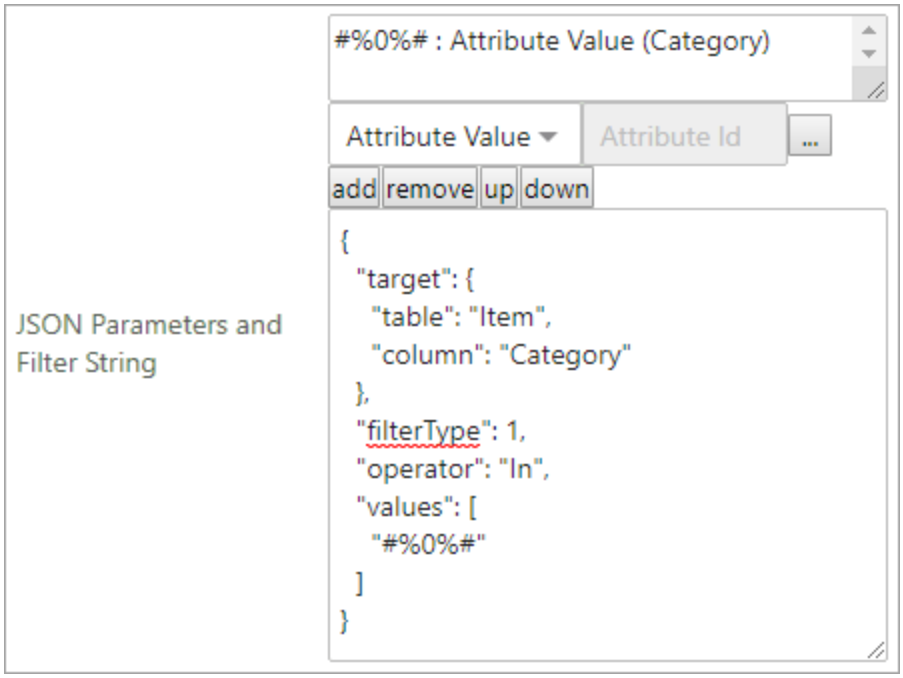
Single Filter

This sample shows a basic JSON filter string that is configured to have a **dynamic value replacement** in the form of a placeholder to fill in the 'values' field. The placeholder string **#%0%#** will be replaced with the value(s) for the attribute 'Category' for whichever node is currently selected in the Web UI. A screenshot of how the filter displays in the Web UI designer appears below the string.

Note: The placeholder string (e.g., #%0%#) is specific to STEP functionality and is not part of the schema as defined by Microsoft.

```
{
  "target": {
    "table": "Item",
    "column": "Category"
  },
```

```
  "filterType": 1,
  "operator": "In",
  "values": [
    "#%0%#"
  ]
}
```



Note: If you specify a basic filter with the operator set to either 'In' or 'NotIn', and the 'values' field contains no values or a single empty value, you are effectively setting a 'Select All' filter. To set a filter that only matches empty values, you will need to use an Advanced Filter set to 'is blank' instead.

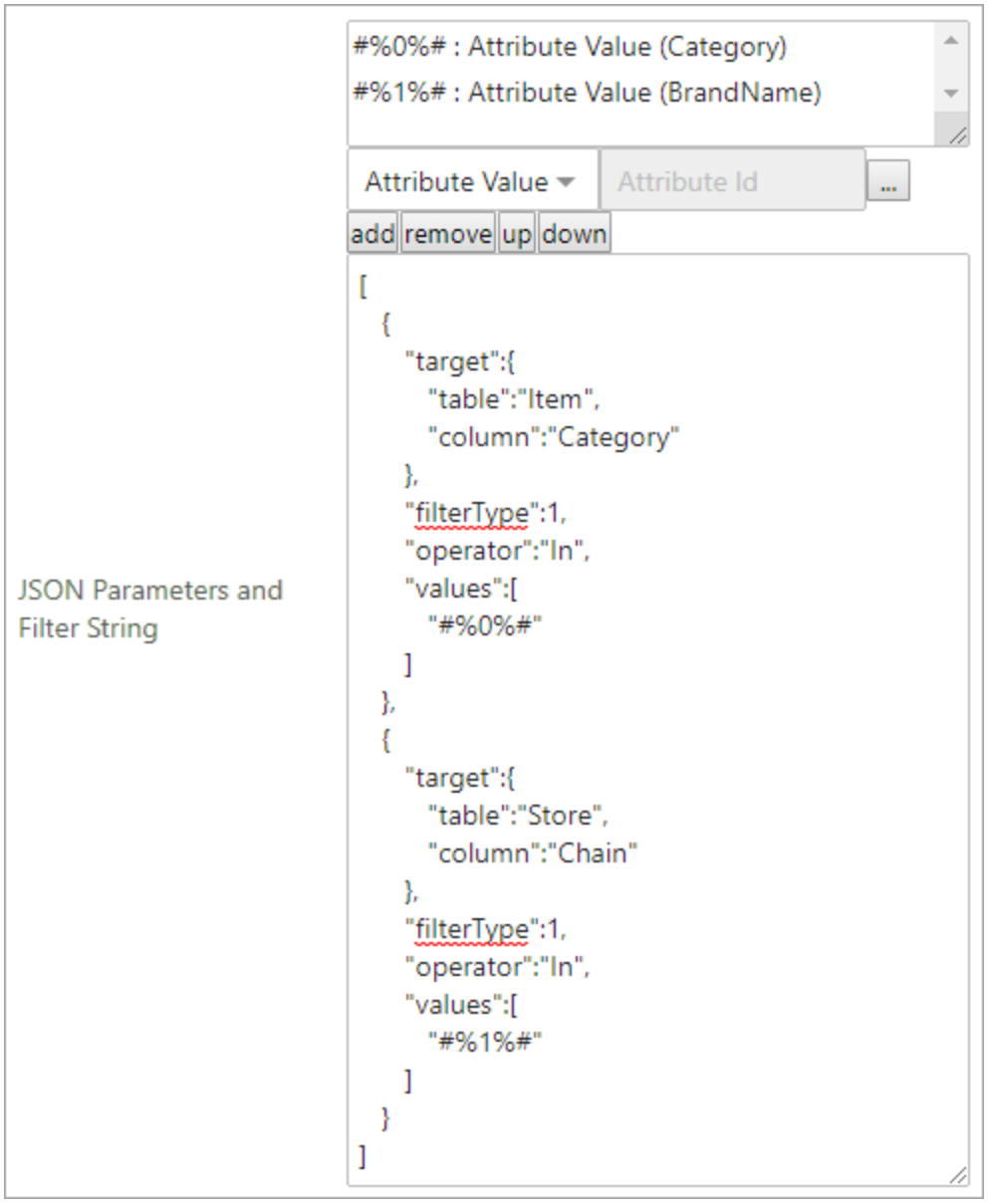
When the relevant product folder is selected in the Web UI on a Node Details screen, the attribute value for 'Category' is passed to Power BI to use in a filter on the 'Item' table / 'Category' column. In this instance, the value of 'Category' on the selected 'Mens' product node is '020-Mens.' This results in a filtered report with '020-Mens' preselected, as shown below:

Multiple Filters Defined in an Array - Hierarchical Example

Multiple filters are defined as an **array** of filters, as shown in the following example. Arrays of filters display within **square** brackets ([ ]) and are separated by commas. This example shows two basic filters in an array. The placeholders '#%0%#' and '#%1%#' will create filter parameters for the attributes Category and Brand Name. This array has two filters because the filtering is being done on two separate attributes.

```
[
   {
      "target":{
         "table":"Item",
         "column":"Category"
      },
      "filterType":1,
      "operator":"In",
      "values":[
         "#%0%#"
      ]
   },
   {
      "target":{
         "table":"Store",
         "column":"Chain"
      },
      "filterType":1,
      "operator":"In",
      "values":[
         "#%1%#"
      ]
```

```
    }
]
```



The example Web UI scenario for the above string is as follows:

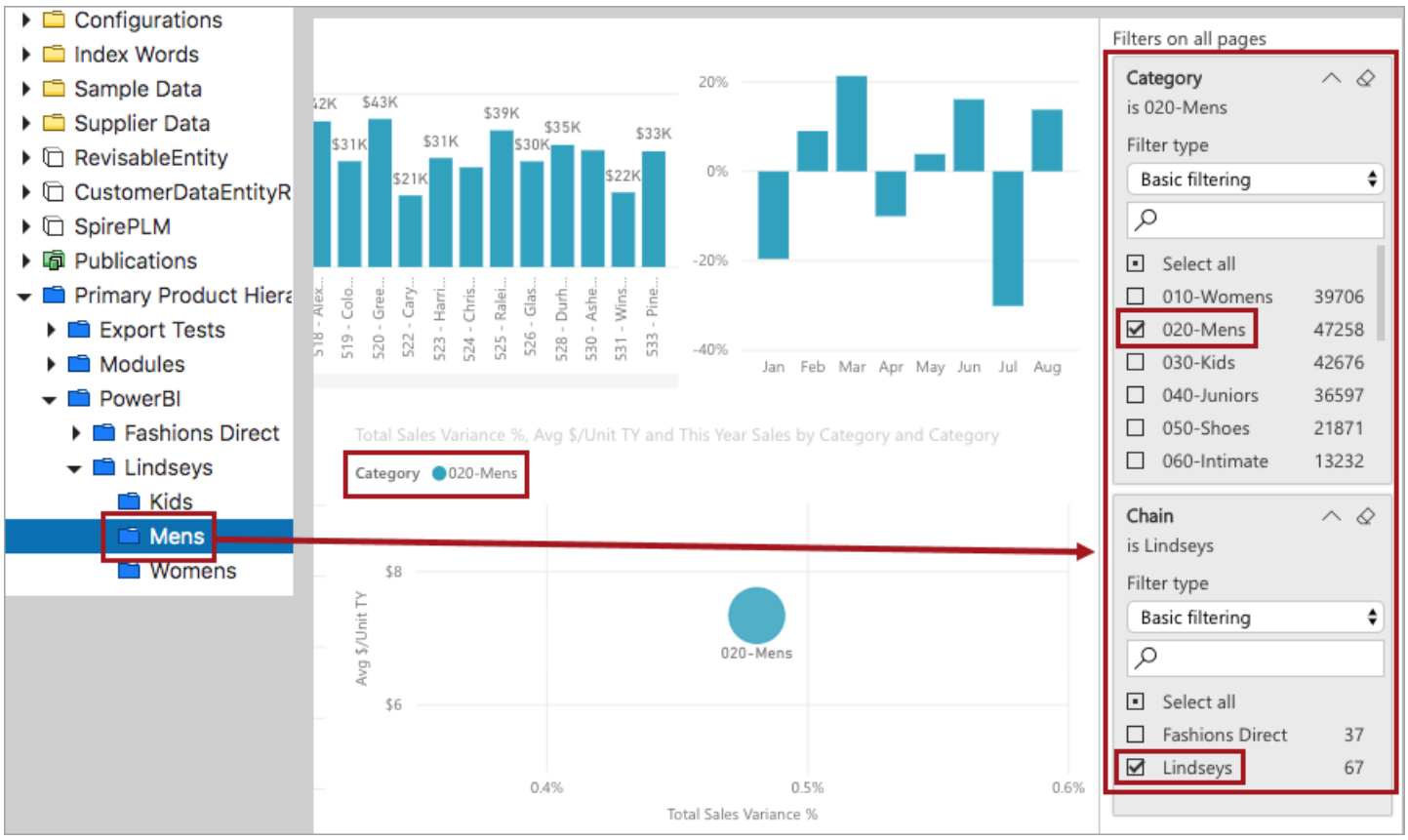
For a hierarchy of products, the report data at each level in the Tree should be further filtered to be more specific to the selected node. This is possible using basic filtering and attribute value inheritance. In this example, there is a folder hierarchy named 'Power BI' that contains two child folders: Fashions Direct and Lindseys. The Lindseys folder contains further category folders—Kids, Mens and Womens. When the Power BI folder is selected in the Web UI, it will display a report that includes all data—both Lindseys and Fashions Direct and all the category folders below them.

The **Power BI product folder** has no values for either the Brand Name or the Category attribute. When specifying a basic filter, if there are no values for the specified attributes, then the filter defaults to 'Select all.'

Note: In Power BI, the 'Select all' checkbox is not always selected when a filter is set to 'Select all.'

The **Lindseys product folder** has a value for the Brand Name but does not have a value for the Category attribute. This sets the filter to 'Select all' for Category and 'Lindseys' for Chain.

The **Mens product folder** has an inherited value for Brand Name (inherited from Lindseys) and also has a value for the Category attribute. When you select the Power BI > Lindseys > Mens folder, this sets the filter to 'Mens' for Category and 'Lindseys' for Chain.


Configurations
Index Words
Sample Data
Supplier Data
RevisableEntity
CustomerDataEntityR
SpirePLM
Publications
Primary Product Hiera
Export Tests
Modules
PowerBI
Fashions Direct
Lindseys
Kids
Mens
Womens
$43K
$31K
$21K
$31K
$39K
$30K
$35K
$22K
$33K
20%
0%
-20%
-40%
Jan Feb Mar Apr May Jun Jul Aug
Total Sales Variance %, Avg $/Unit TY and This Year Sales by Category and Category
Category 020-Mens
$8
$6
Avg $/Unit TY
020-Mens
0.4%
0.5%
0.6%
Total Sales Variance %
Filters on all pages
Category
is 020-Mens
Filter type
Basic filtering
Select all
010-Womens 39706
020-Mens 47258
030-Kids 42676
040-Juniors 36597
050-Shoes 21871
060-Intimate 13232
Chain
is Lindseys
Filter type
Basic filtering
Select all
Fashions Direct 37
Lindseys 67

Power BI Row-Level Security

The visual integration with Power BI supports row-level security (RLS) as a way to restrict data access for specified users. Like in STEP, users can be given 'super user' privileges in Power BI that allow them to view all information in reports, dashboards, and tiles, while other users can be set up with more limited permissions that allow them to view only specified subsets of information.

This topic provides an example setup of row-level security applied to data from two fictional retail chains (Lindseys and Fashions Direct), how to set up users and permissions in Power BI through the Power BI Desktop application, and how to set up corresponding users and groups in the workbench.

The following **considerations** apply to configuring row-level security for Power BI:

- To configure row-level security, **roles** and **rules** must be defined within the **Power BI Desktop** application; they cannot be defined from within the Web UI or by logging into Power BI (via https://app.powerbi.com).
- The Power BI Desktop application is only available for the **Windows** platform. For more information on Power BI Desktop, including download access, refer to https://powerbi.microsoft.com/desktop.

For more detailed information from Microsoft on how to configure row-level security for Power BI, refer to the following topics:

- Row-level security with Power BI Embedded: https://docs.microsoft.com/en-us/power-bi/developer/embedded-row-level-security
- Row-Level security (RLS) with Power BI: https://docs.microsoft.com/en-us/power-bi/service-admin-rls

Overview of Row-Level Security in Power BI

Row-level security in Power BI is enforced by filters that are applied in Power BI to restrict data. Three main concepts are used to configure row-level security in Power BI: **users**, **roles**, and **rules**.

- **Users:** The end **users** that view the artifact (dashboard, tile, or report). After configuring Power BI authentication, users will be automatically authenticated with Power BI when viewing Power BI data in the Web UI. For more information, refer to the **Power BI Authentication Configuration** topic.
- **Roles:** Users belong to **roles**. A role is a container for rules and can be named something like Sales Representative or Sales Manager.
- **Rules:** Row-level security is defined within the roles. Roles have **rules**, and these rules are the actual filters that are applied to the data. The rules can be as simple as 'Country = Canada' or something more complex.

From a STEP perspective, when row-level security is applied, the Power BI **user** for which STEP requests an access token will be the STEP **user ID** of the currently logged in user. The Power BI **roles** that the user belongs to will correspond to the STEP **user group IDs** that the user is a member of.

Example Row-Level Security Setup

The following example uses a 'Retail Analysis Sample' data model to outline the steps for setting up row-level security in Power BI. These steps include:

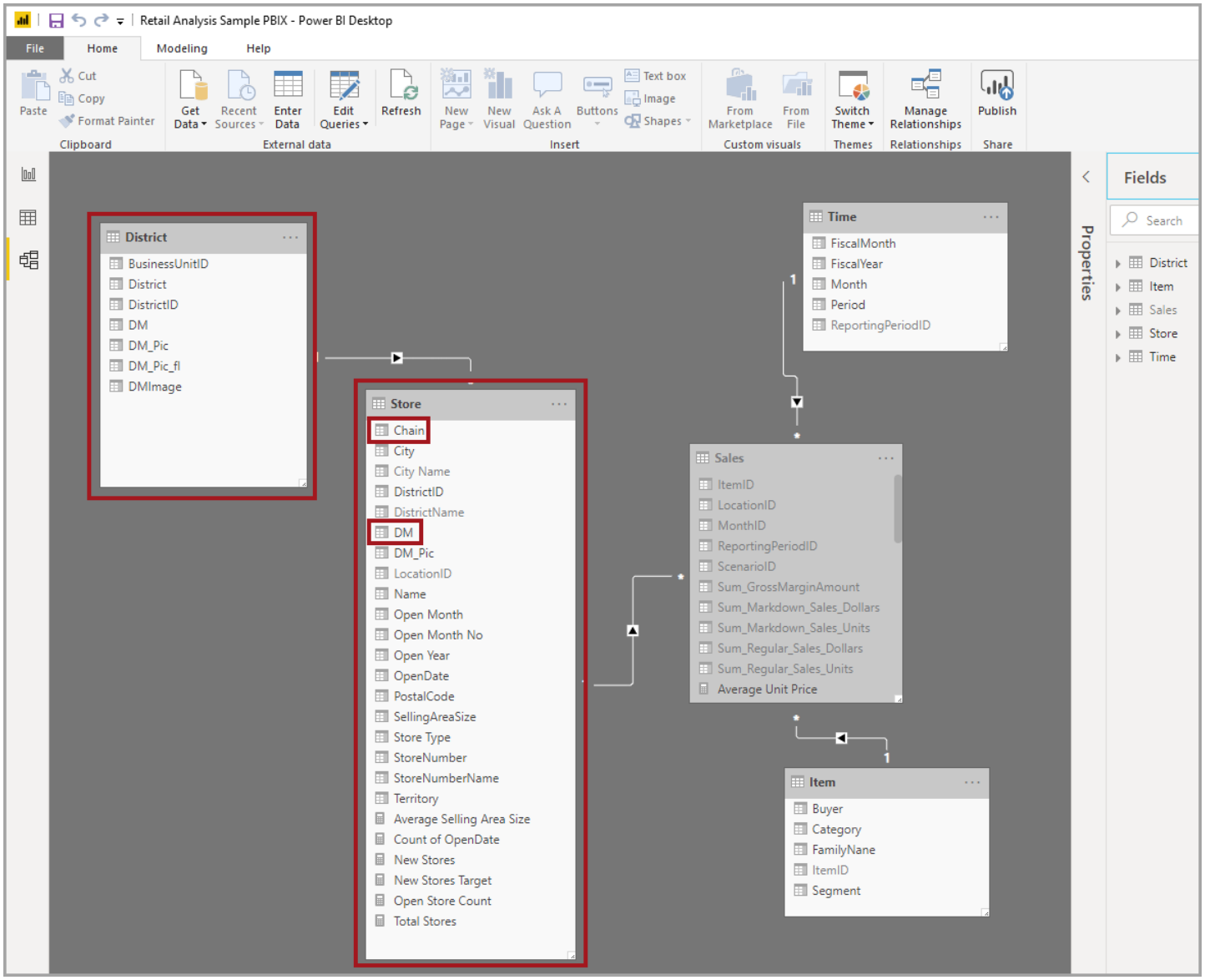
- Create roles and rules in Power BI Desktop
- Publish the data to Power BI
- Set up users and groups in workbench
- Confirm filtering in Web UI

Note: These instructions describe a high-level overview of a sample setup that includes fictional retailers and users. For more information on how to navigate the Power BI Desktop interface itself, as well as how to configure a solution that is a fit for your specific business needs, refer to the 'Row-level security (RLS) with Power BI' topic in the Power BI help documentation: https://docs.microsoft.com/en-us/power-bi/service-admin-rls.

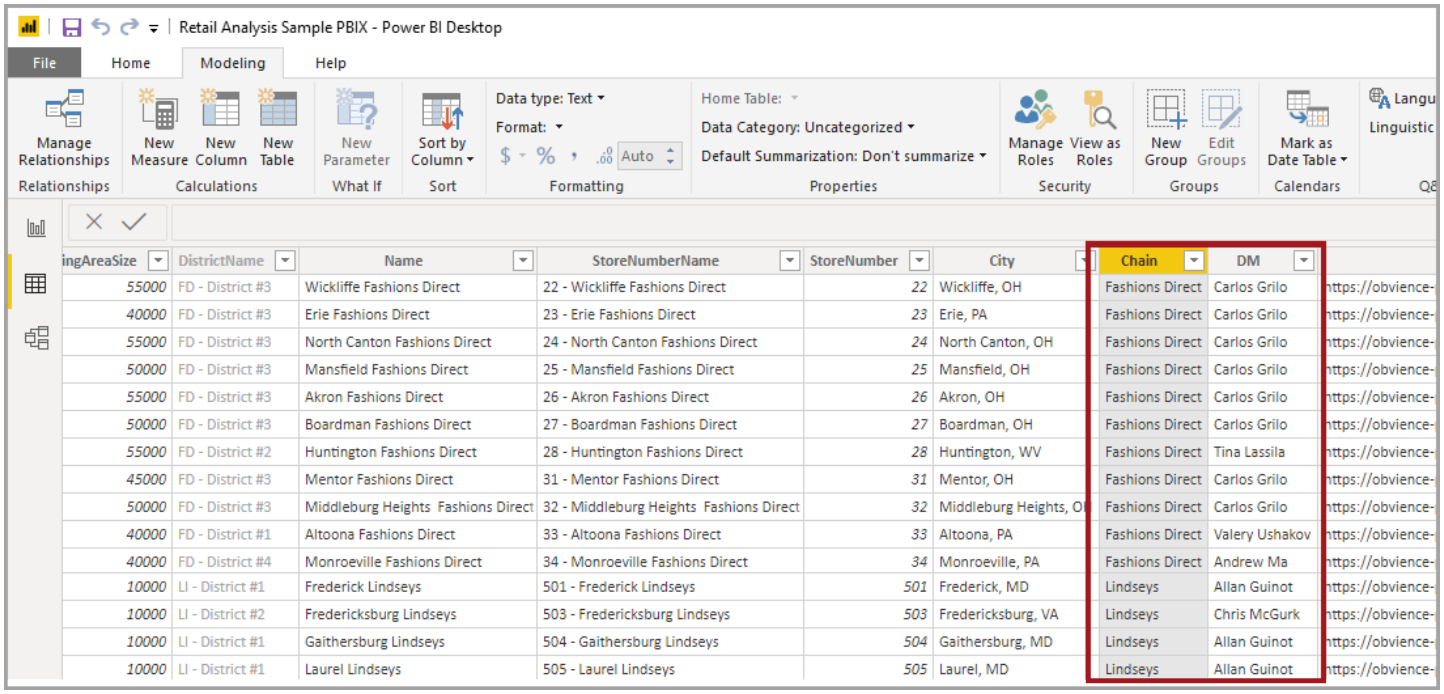
Create Roles and Rules in Power BI Desktop

Before these steps can be carried out, you must first download the **Power BI Desktop** application, which is only available for the Windows platform. For more information, refer to https://powerbi.microsoft.com/desktop.

1. The below screenshot shows the Power BI Desktop interface with five database tables displayed. This example uses information from the 'District' and 'Store' tables, with a focus on the 'Chain' and 'DM' (district manager) columns from the Store table.



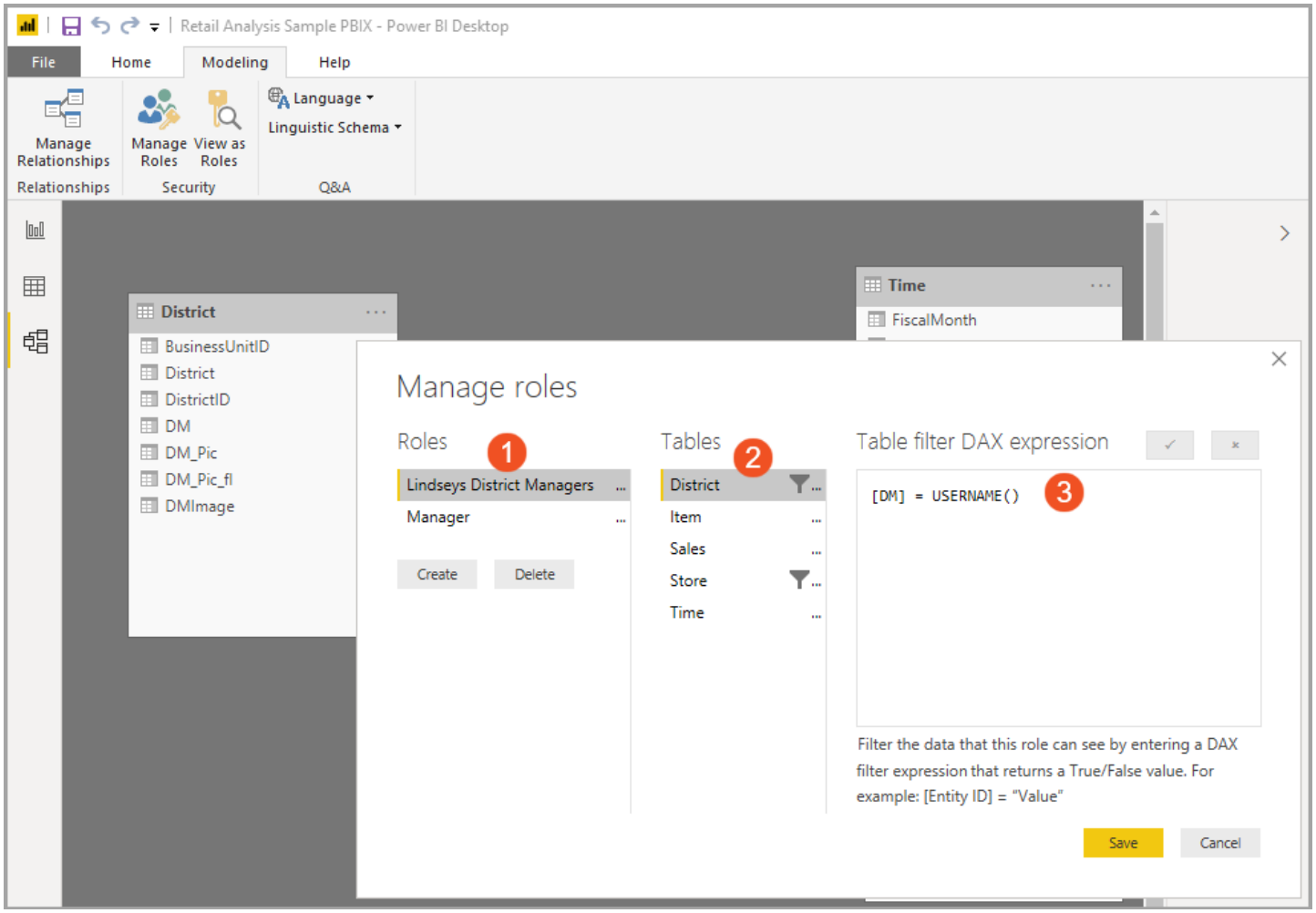
The next screenshot shows more details about the Store table, with the Chain and DM columns highlighted. The Chain column contains the names of the two fictional retailers (Fashions Direct and Lindseys), and the DM column contains the names of the fictional district managers.



ingAreaSize	DistrictName	Name	StoreNumberName	StoreNumber	City	Chain	DM	
55000	FD - District #3	Wickliffe Fashions Direct	22 - Wickliffe Fashions Direct	22	Wickliffe, OH	Fashions Direct	Carlos Grilo	https://obvience-
40000	FD - District #3	Erie Fashions Direct	23 - Erie Fashions Direct	23	Erie, PA	Fashions Direct	Carlos Grilo	https://obvience-
55000	FD - District #3	North Canton Fashions Direct	24 - North Canton Fashions Direct	24	North Canton, OH	Fashions Direct	Carlos Grilo	https://obvience-
50000	FD - District #3	Mansfield Fashions Direct	25 - Mansfield Fashions Direct	25	Mansfield, OH	Fashions Direct	Carlos Grilo	https://obvience-
55000	FD - District #3	Akron Fashions Direct	26 - Akron Fashions Direct	26	Akron, OH	Fashions Direct	Carlos Grilo	https://obvience-
50000	FD - District #3	Boardman Fashions Direct	27 - Boardman Fashions Direct	27	Boardman, OH	Fashions Direct	Carlos Grilo	https://obvience-
55000	FD - District #2	Huntington Fashions Direct	28 - Huntington Fashions Direct	28	Huntington, WV	Fashions Direct	Tina Lassila	https://obvience-
45000	FD - District #3	Mentor Fashions Direct	31 - Mentor Fashions Direct	31	Mentor, OH	Fashions Direct	Carlos Grilo	https://obvience-
50000	FD - District #3	Middleburg Heights Fashions Direct	32 - Middleburg Heights Fashions Direct	32	Middleburg Heights, OH	Fashions Direct	Carlos Grilo	https://obvience-
40000	FD - District #1	Altoona Fashions Direct	33 - Altoona Fashions Direct	33	Altoona, PA	Fashions Direct	Valery Ushakov	https://obvience-
40000	FD - District #4	Monroeville Fashions Direct	34 - Monroeville Fashions Direct	34	Monroeville, PA	Fashions Direct	Andrew Ma	https://obvience-
10000	LI - District #1	Frederick Lindseys	501 - Frederick Lindseys	501	Frederick, MD	Lindseys	Allan Guinot	https://obvience-
10000	LI - District #2	Fredericksburg Lindseys	503 - Fredericksburg Lindseys	503	Fredericksburg, VA	Lindseys	Chris McGurk	https://obvience-
10000	LI - District #1	Gaithersburg Lindseys	504 - Gaithersburg Lindseys	504	Gaithersburg, MD	Lindseys	Allan Guinot	https://obvience-
10000	LI - District #1	Laurel Lindseys	505 - Laurel Lindseys	505	Laurel, MD	Lindseys	Allan Guinot	https://obvience-

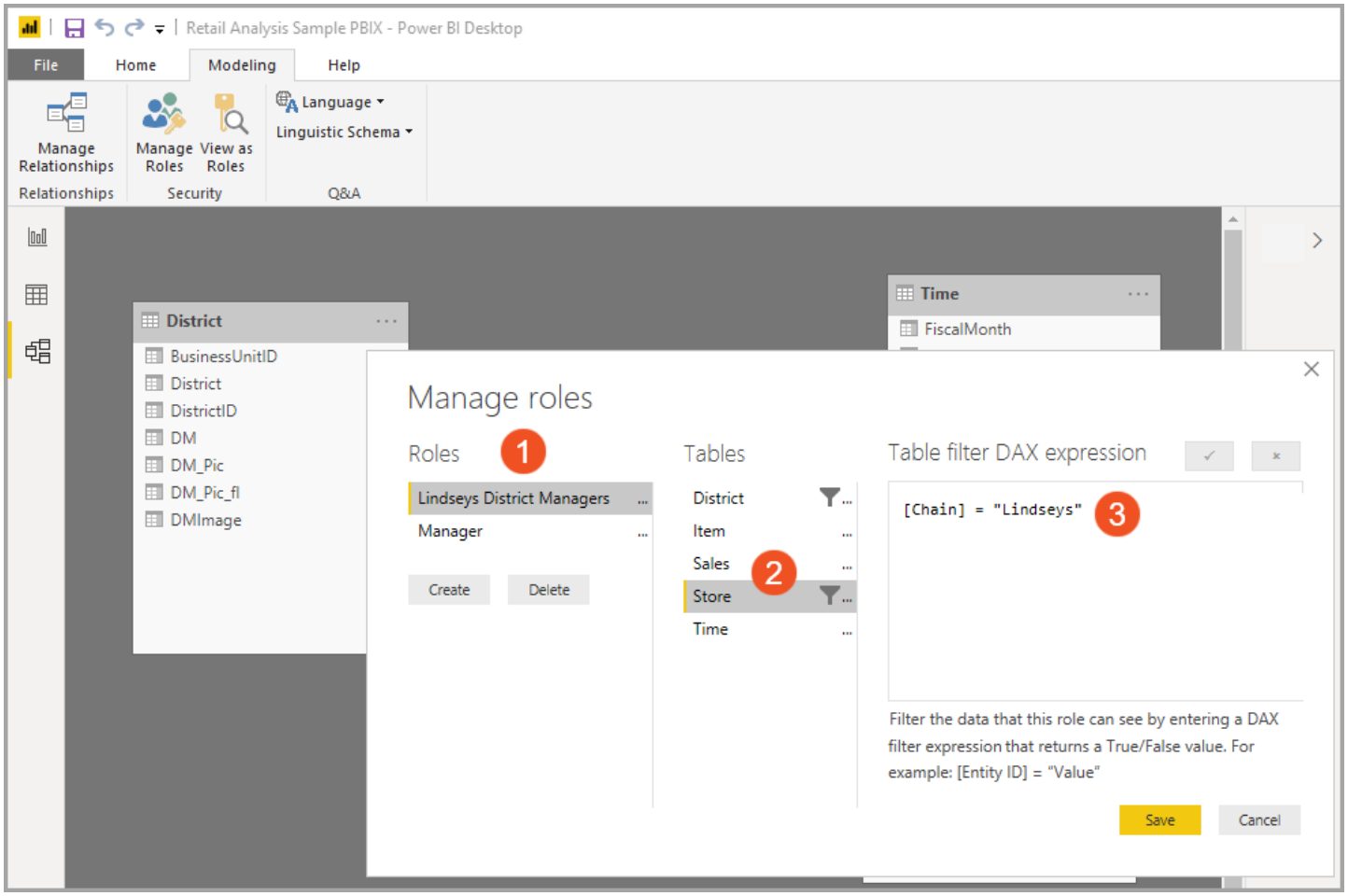
2. The first **role** created for this example setup is 'Lindseys District Managers' (step 1 in the below screenshot).

   The first **rule** created for this role is to match DM to username. Select the **District** table (step 2), then match DM to username by entering the `[DM] = USERNAME()` string into the Table filter DAX expression field (step 3). The STEP user ID will be passed to Power BI in such a way that it can be accessed via the `USERNAME()` function.



3. The second **rule** created for Lindseys District Managers (step 1 in the below screenshot) is to match Chain to the name of the relevant store chain (Lindseys).

   Select the **Store** table (step 2), then match Chain to Lindseys by entering the string `[Chain] = "Lindseys"` into the Table filter DAX expression field (step 3).

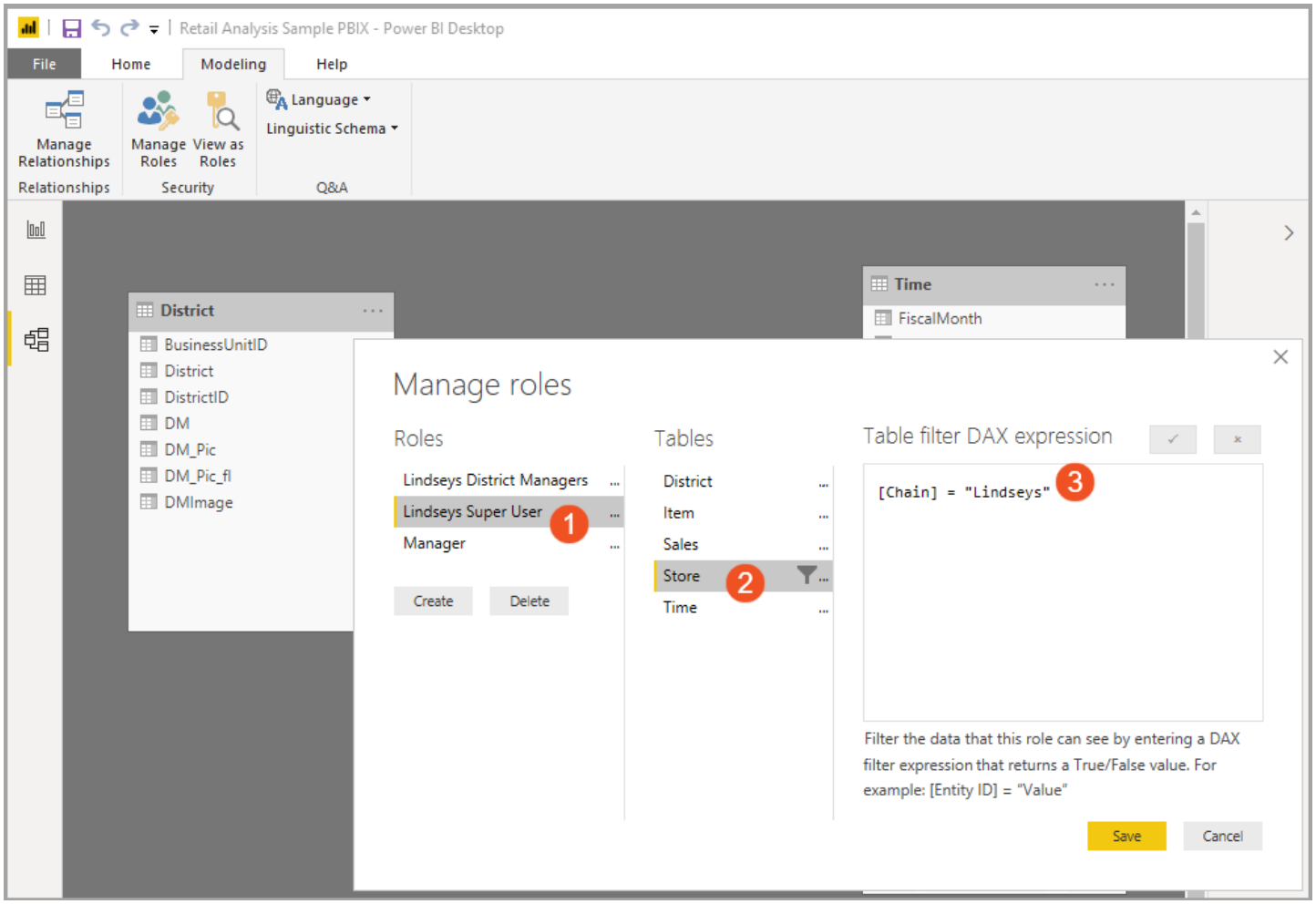


Later in the configuration, a corresponding STEP **user group** will be created named 'Lindseys District Managers.' The ID of the group defined in STEP is passed across and is used to match the Name of the **role** in Power BI. The two **rules** defined in the preceding steps will ensure that if a user is part of the 'Lindseys District Managers' group in STEP, *and* their username matches the DM field, they will only access data applicable to Lindseys and their user ID.

4. The next **role** created for this example setup is 'Lindseys Super User' (step 1 in the following screenshot).

   Create a **rule** to match Chain to Lindseys by selecting the **Store** table (step 2), then enter the string `[Chain] = "Lindseys"` into the Table filter DAX expression field (step 3).

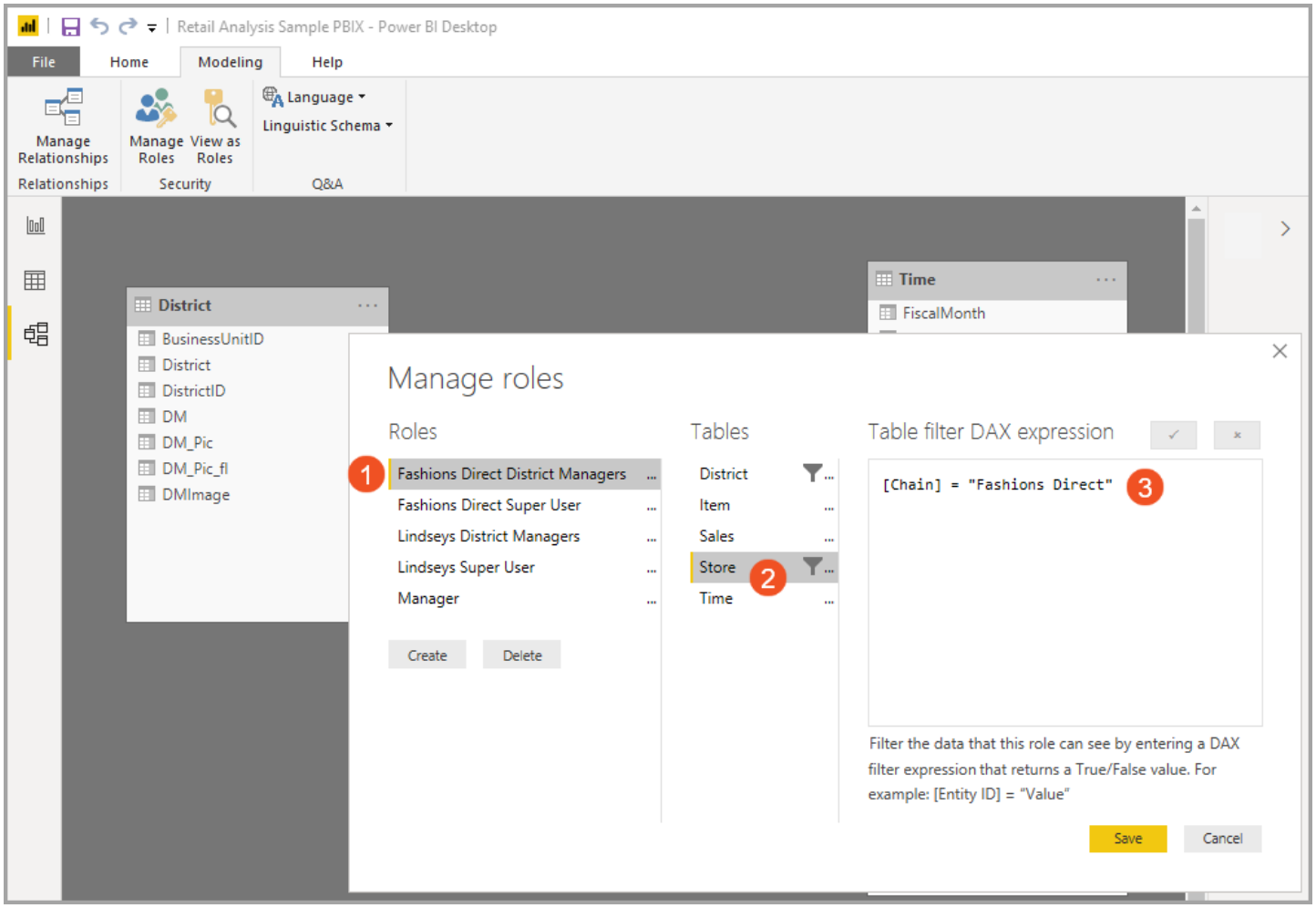
   This rule ensures that anyone who is part of the Lindseys Super User group will be able to examine all Lindseys-specific data, regardless of their user name (i.e., there is no filtering on user name applied).



5. Next, create the 'Fashions Direct District Managers' **role** (step 1 in below screenshot).

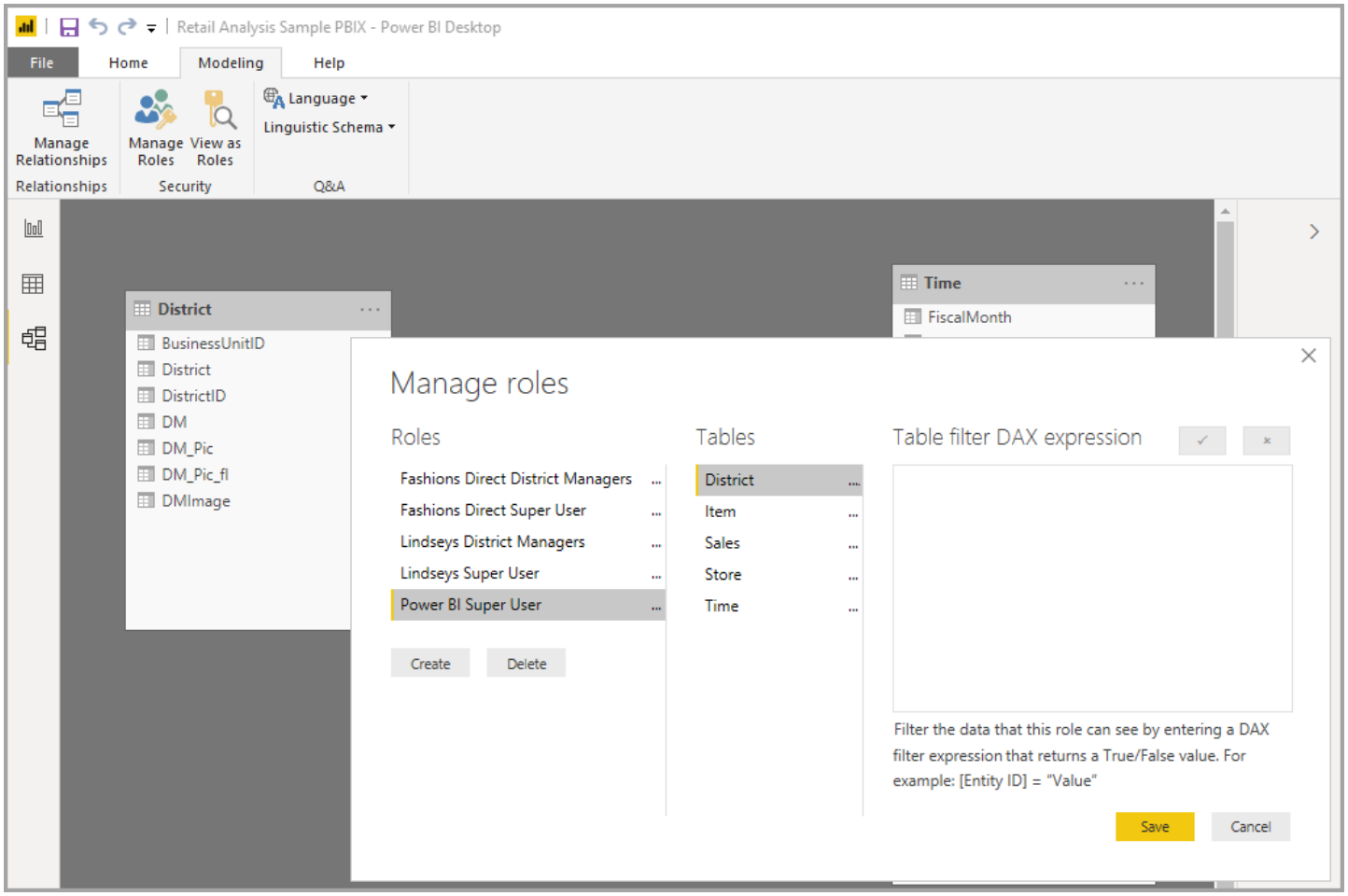
   Create a **rule** *(not pictured in the below screenshot)* to match DM to username for users within the Fashions Direct District Managers group by selecting the **District** table, then entering the `[DM] = USERNAME()` string into the Table filter DAX expression field.

   Create a second **rule** to match Chain to Fashions Direct by selecting the **Store** table (step 2), then entering the string `[Chain] = "Fashions Direct"` into the Table filter DAX expression field (step 3).



These rules ensure that if a user is part of the 'Fashions Direct District Manager' STEP user group, *and* their username matches the DM field, they will only examine data applicable to 'Fashions Direct' and their user ID.

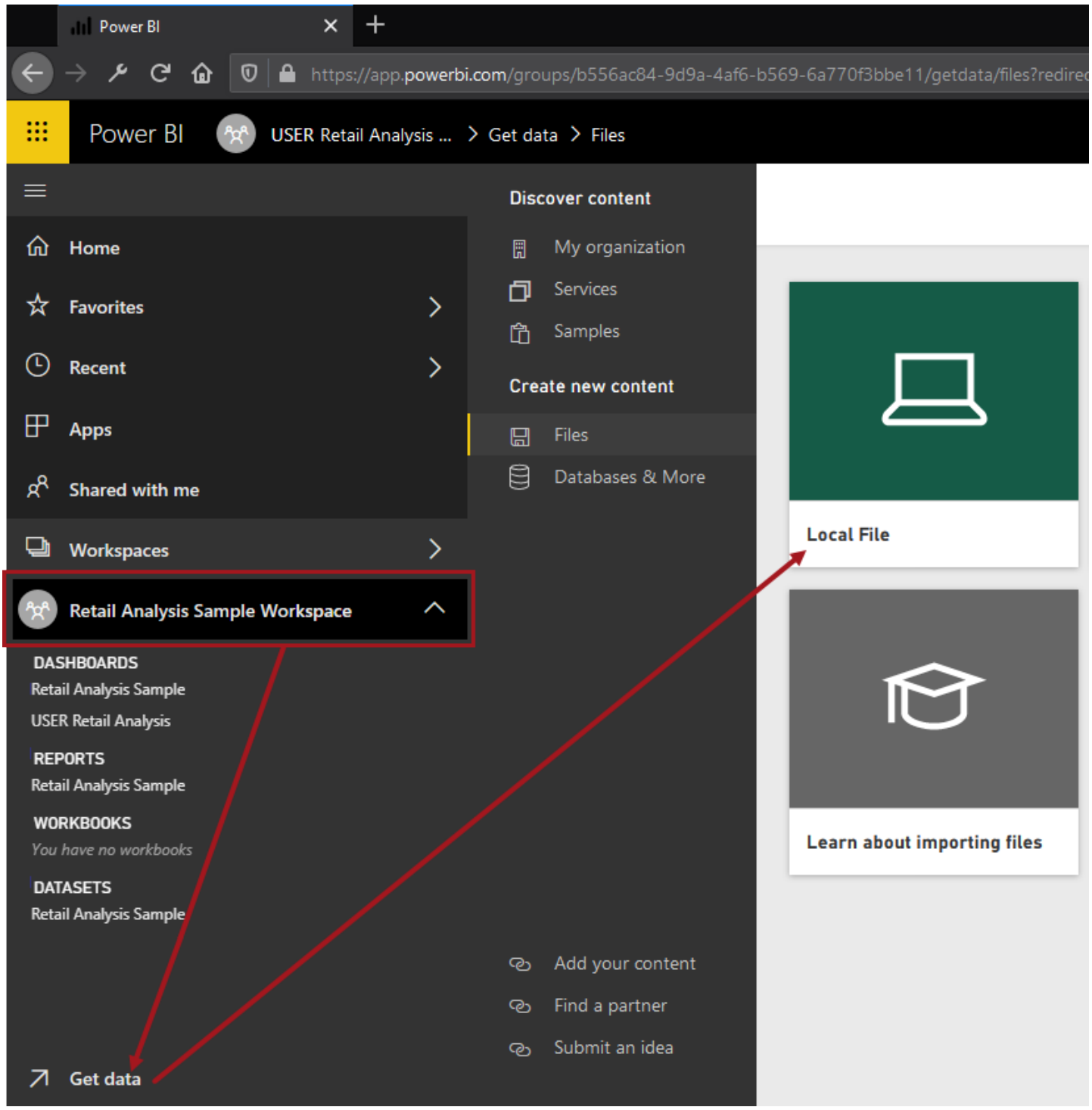
6. Create a 'Fashions Direct Super User' **role** in a similar way how the 'Lindseys Super User' group was created, including a **rule** to match Chain to Fashions Direct by selecting the **Store** table, then entering the string `[Chain] = "Fashions Direct"` into the Table filter DAX expression field.
7. Last, create a 'Power BI Super User' role that has no filtering set on any of the tables. Anyone with this role will be able to examine all data completely unfiltered.



8. Click **Save** to store the Power BI pbix file (Power BI Desktop report) to your computer. You will upload (publish) this file to Power BI to continue the setup.

Publish the Data to Power BI

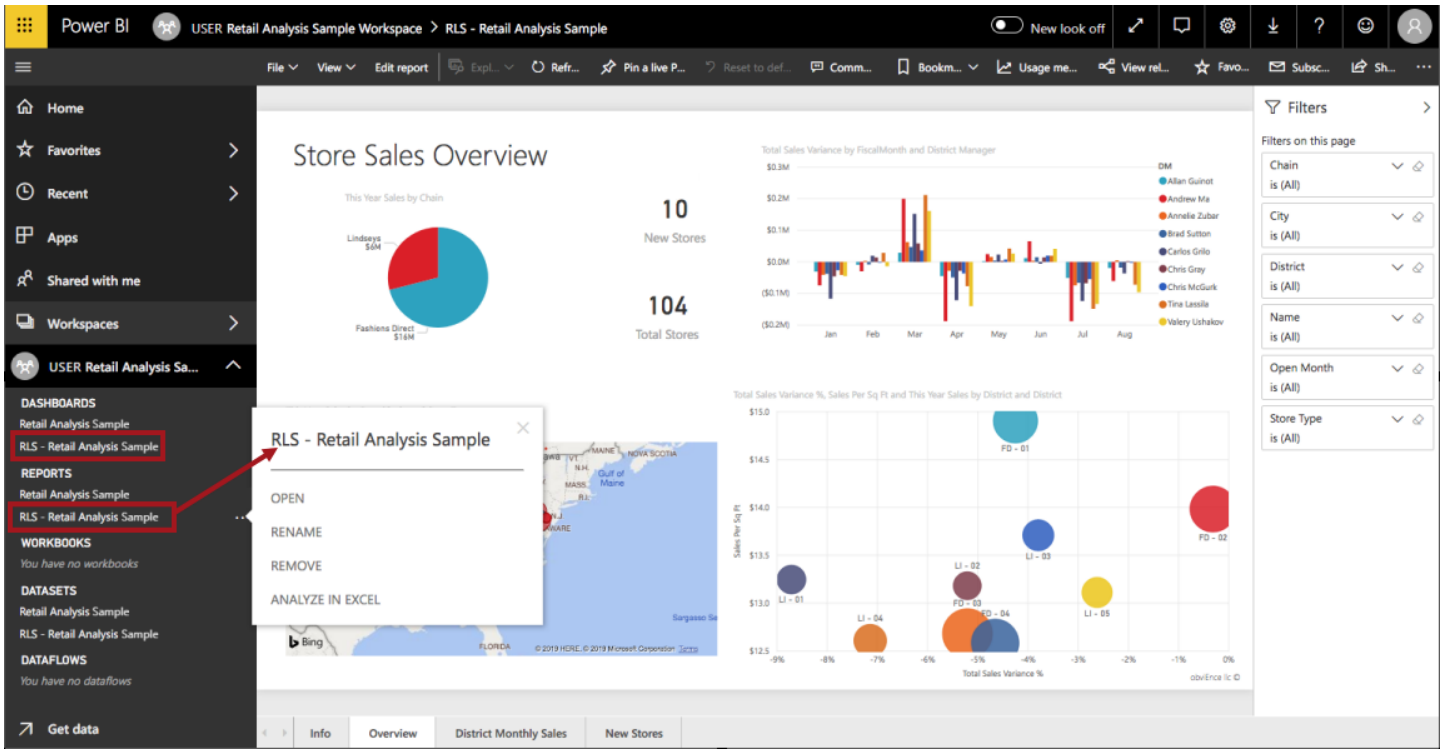
1. Log in to Power BI via https://app.powerbi.com.
2. Select the Power BI workspace to upload the Power BI report with RLS.
3. Click 'Get data,' then select **Local File**.



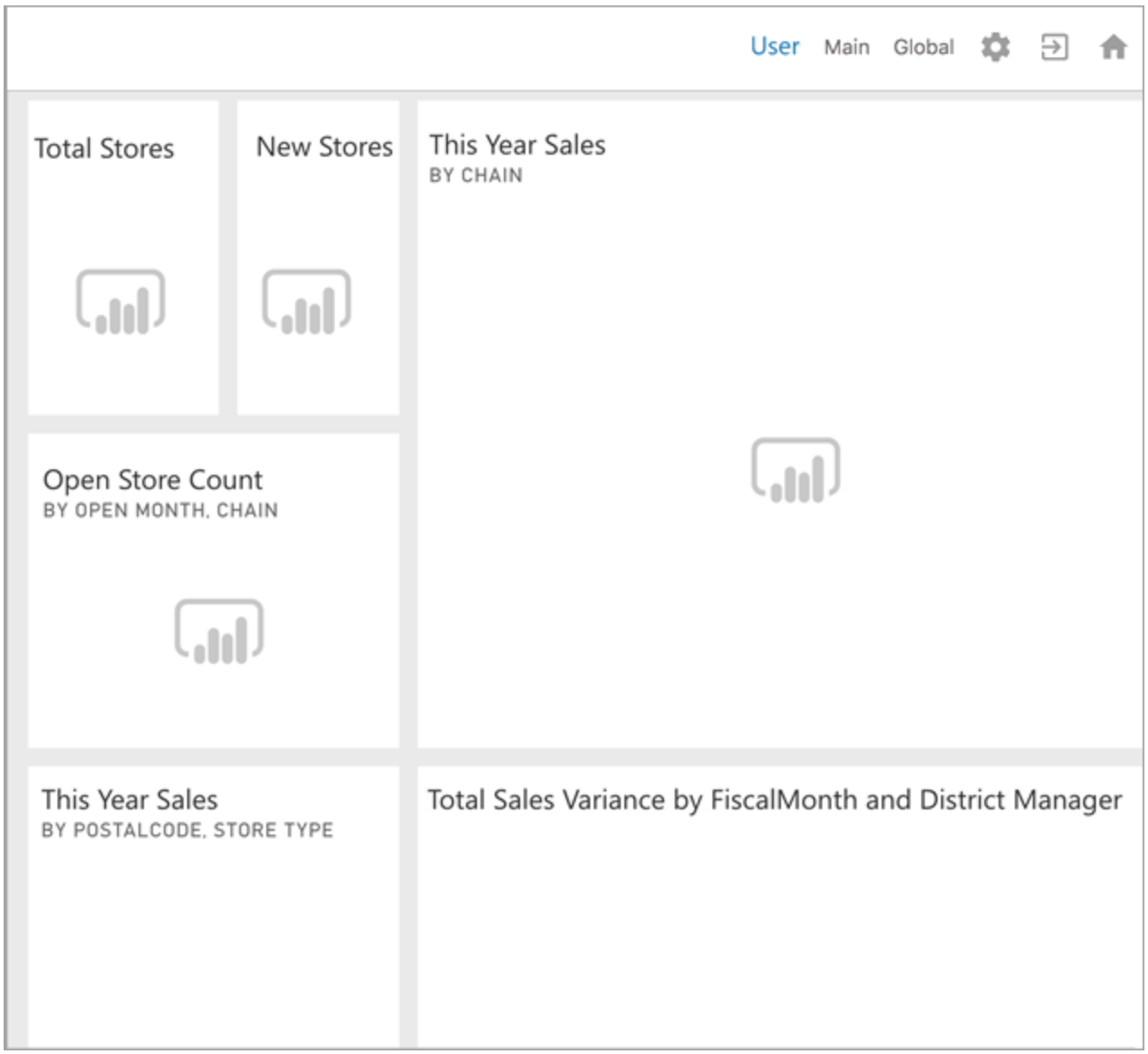
4. Upload the pbix file that you previously saved. This will upload the dataset, report, and dashboard to Power BI.

Note: The dashboard will probably be empty, so you will need to add tiles to it yourself. Refer to https://docs.microsoft.com/en-us/power-bi/service-dashboards for more information.

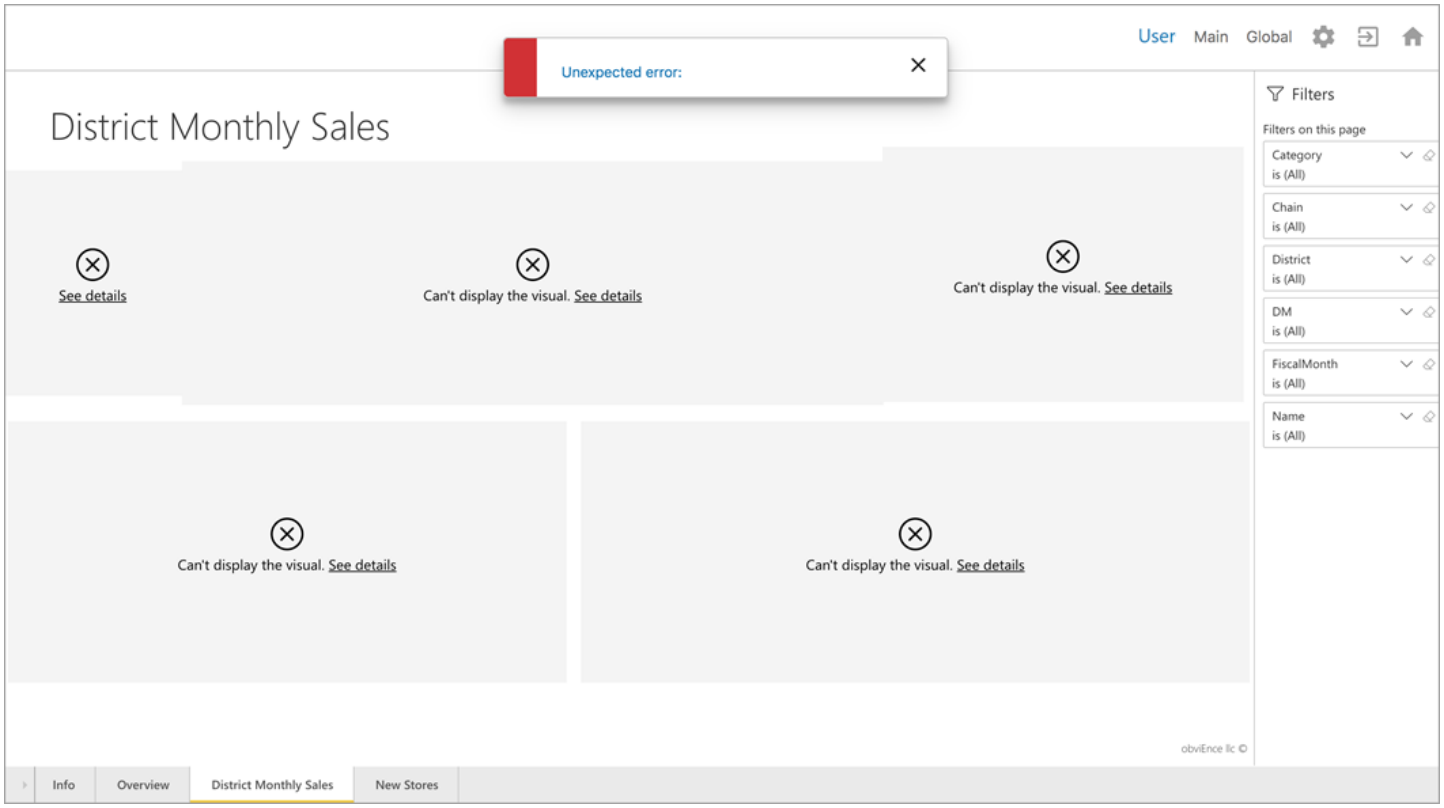
5. You now have a workspace with multiple dashboards / reports. In the below example, the RLS-enabled Power BI dashboards and reports are prefixed with **RLS**. The ones without the prefix do not have RLS applied.



At this point, no corresponding users and user groups have been set up in STEP, so there are no roles that Power BI understands. Therefore, no Power BI information will display in the Web UI. For example, if you log into the Web UI and attempt to view a **Dashboard** with RLS applied, it will look like the following screenshot:



If you attempt to view a **Report** with RLS applied, it will look like the following screenshot. This is because the logged-in user is not part of any role that Power BI understands, so Power BI has nothing to display.

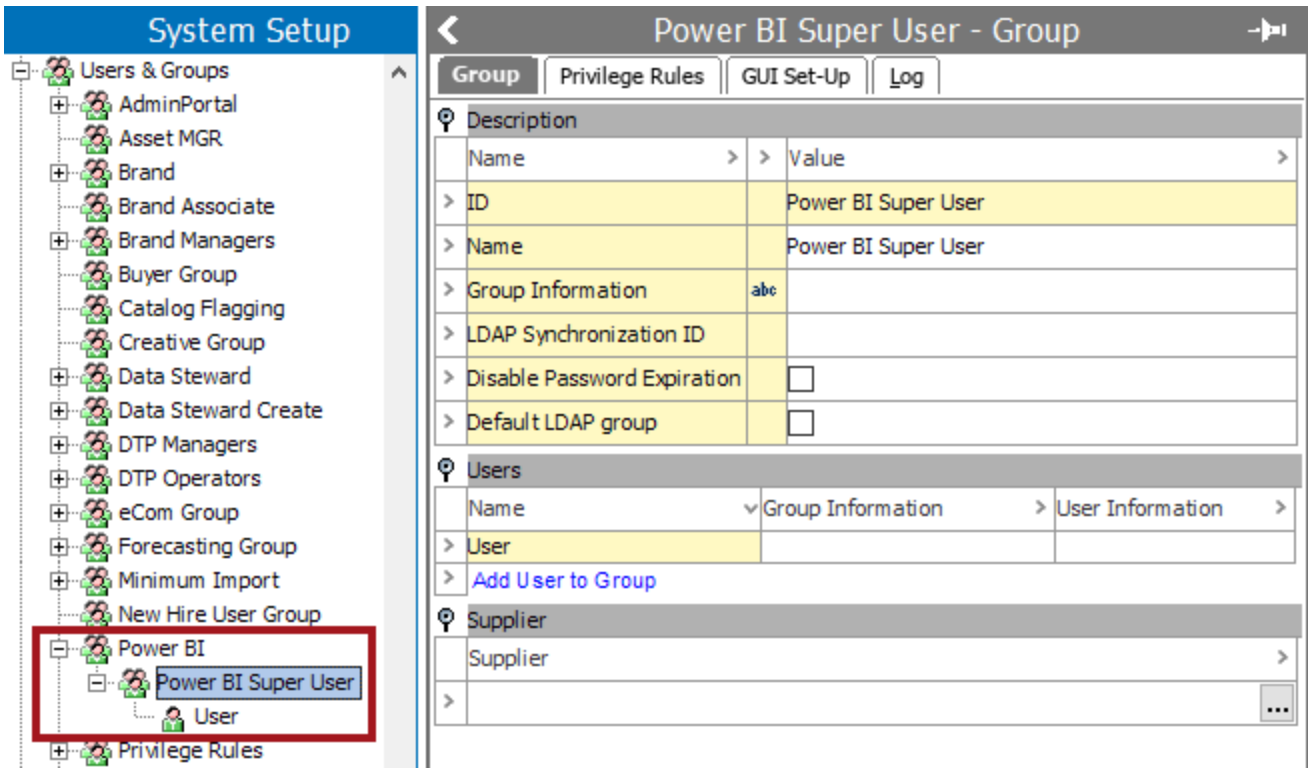


Set Up Users and Groups in Workbench and Confirm Filtering in Web UI

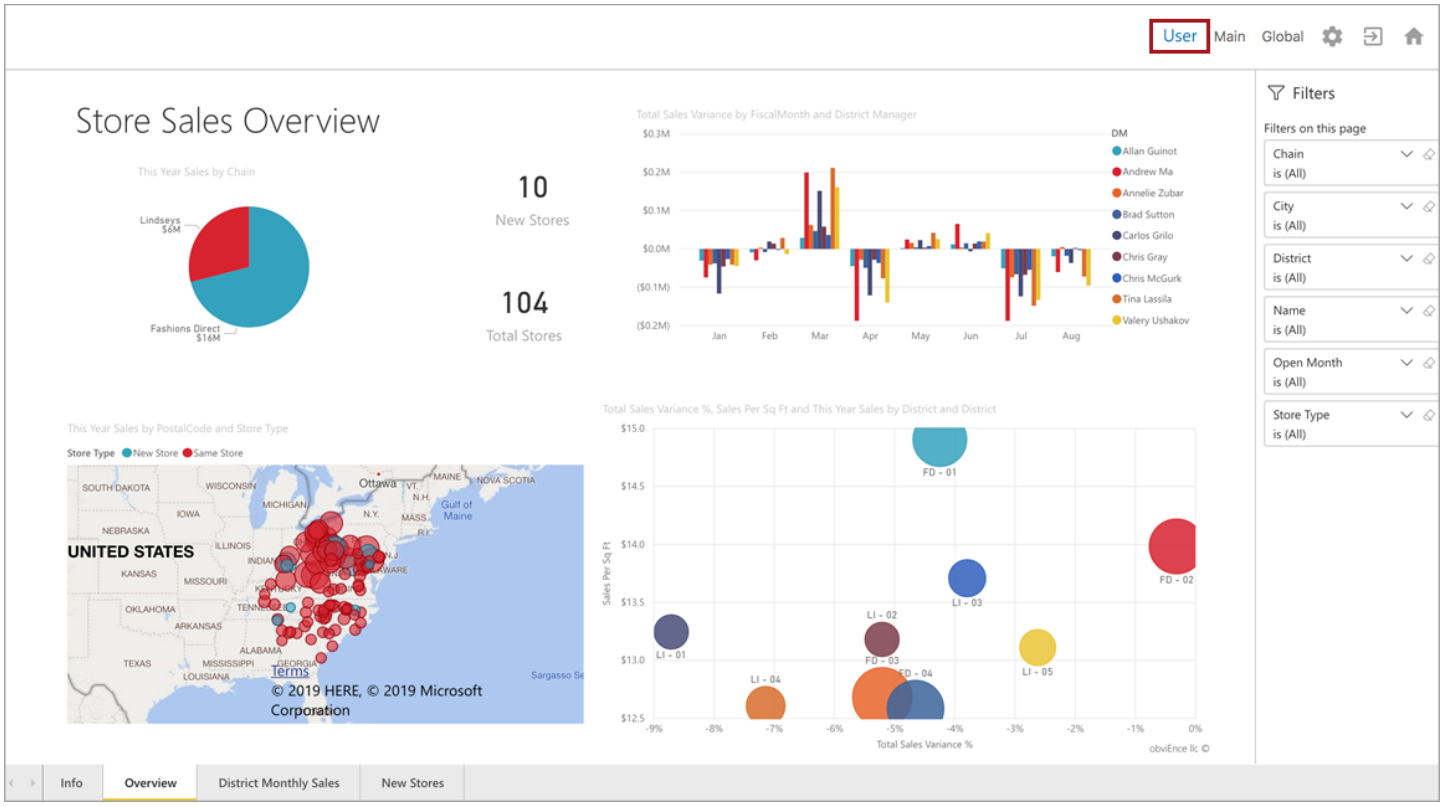
The following steps in this example setup explain a recommended practice for creating users and groups in STEP that correspond to the users and roles created in Power BI Desktop in this topic.

1. In the workbench, create a 'Power BI' user group, then create a group beneath it with the ID 'Power BI Super User.' Add the relevant Power BI super user(s) to this group, e.g., 'User.'

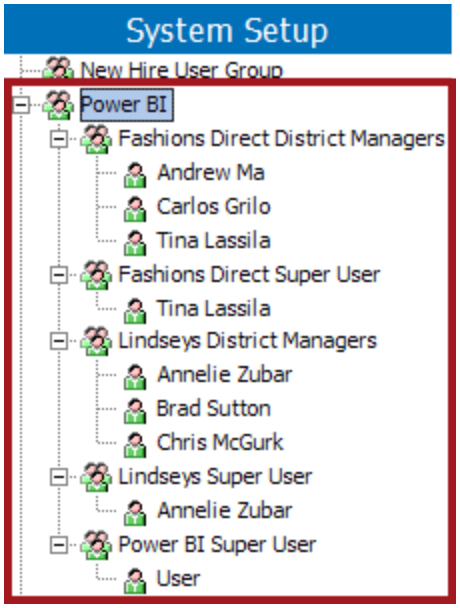
   Note: This is not a hard-coded user group ID; the ID just has to match with the name of the role that has been created in Power BI.



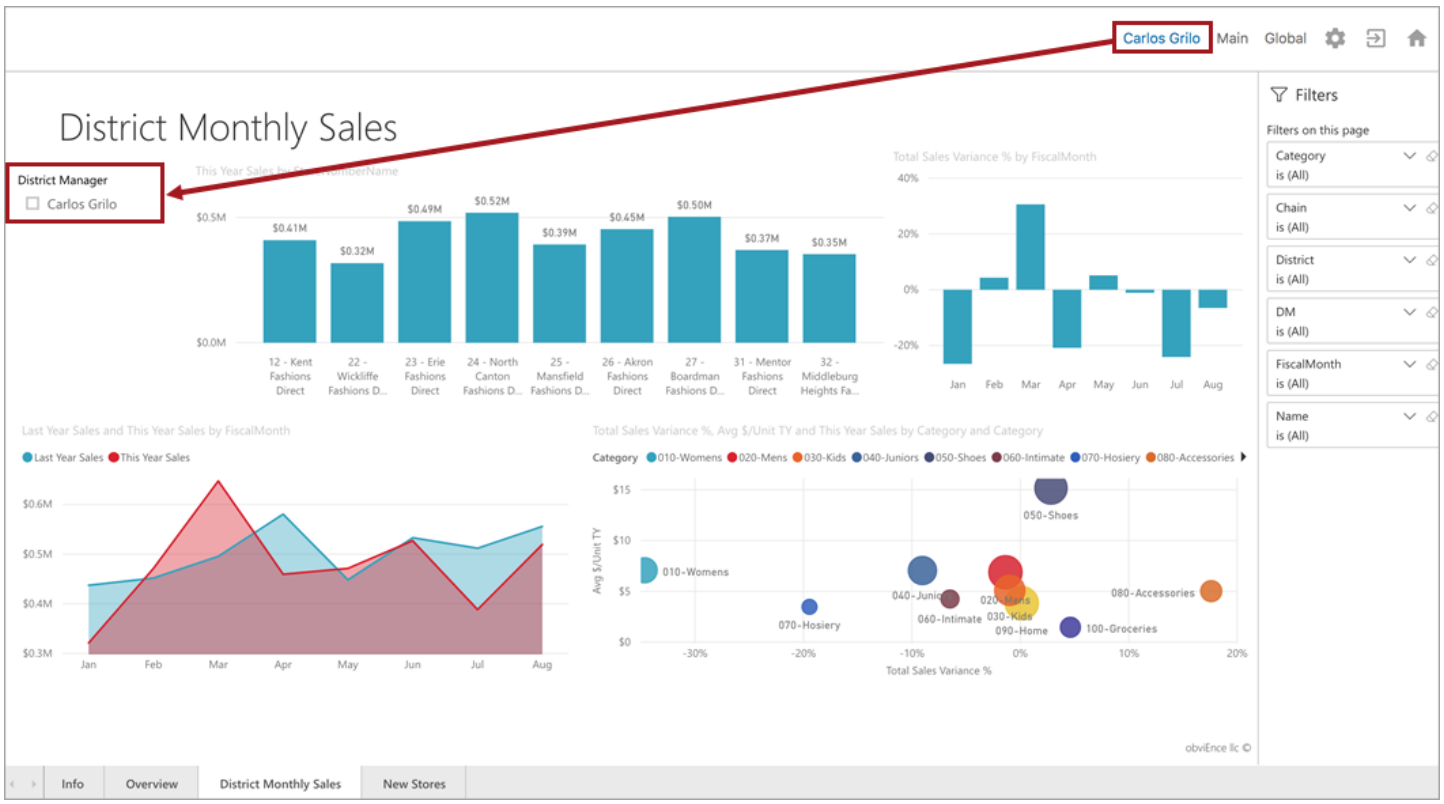
2. Log into your Web UI as 'User.' The previously failing report / dashboard should now show the full unfiltered data because 'User' is part of the Power BI Super User group.



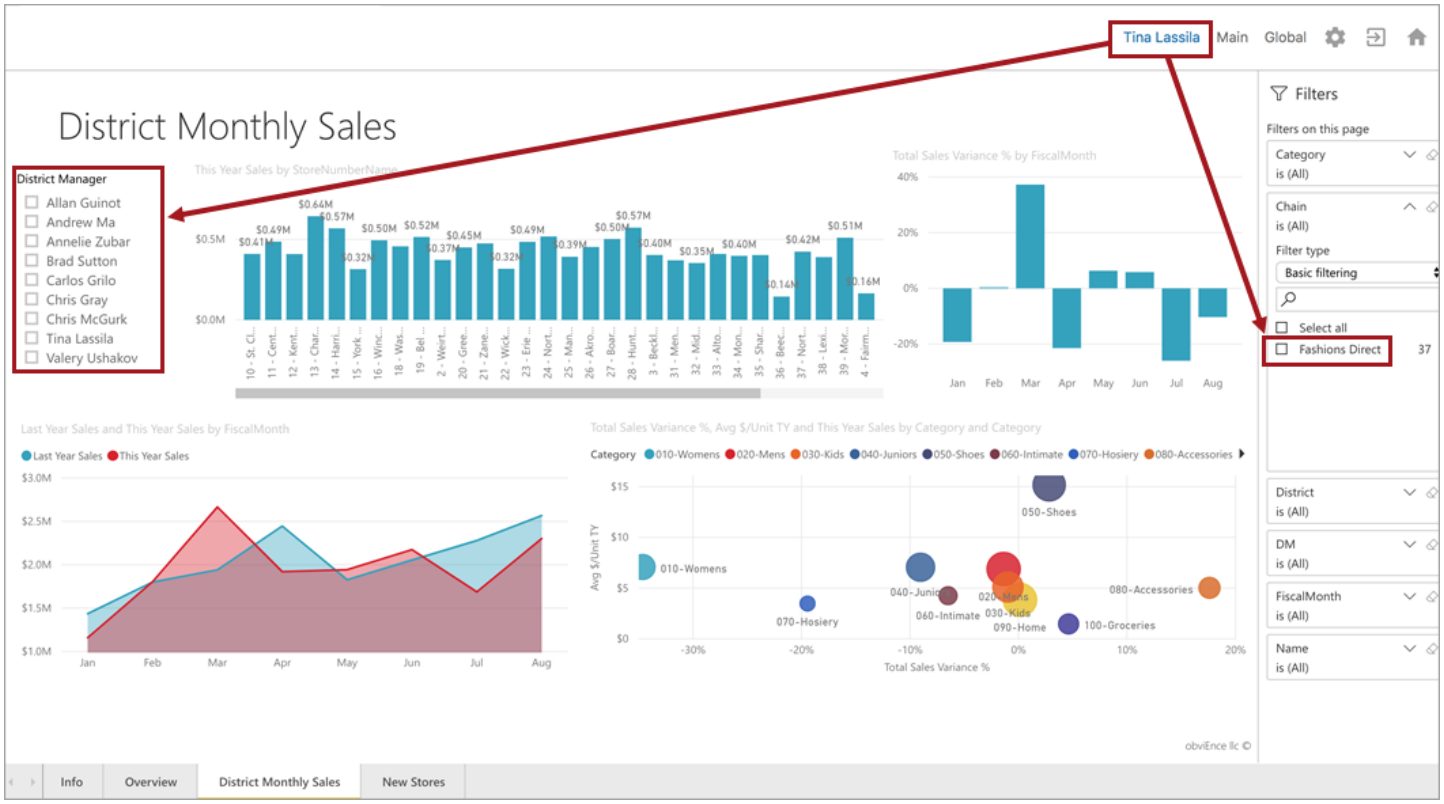
3. Next, create the following users and groups:
   - Fashions Direct District Managers (Andrew Ma, Carlos Grilo, Tina Lassila)
   - Fashions Direct Super User (Tina Lassila)
   - Lindseys District Managers (Annelie Zubar, Brad Sutton, Chris McGurk)
   - Lindseys Super User (Annelie Zubar)



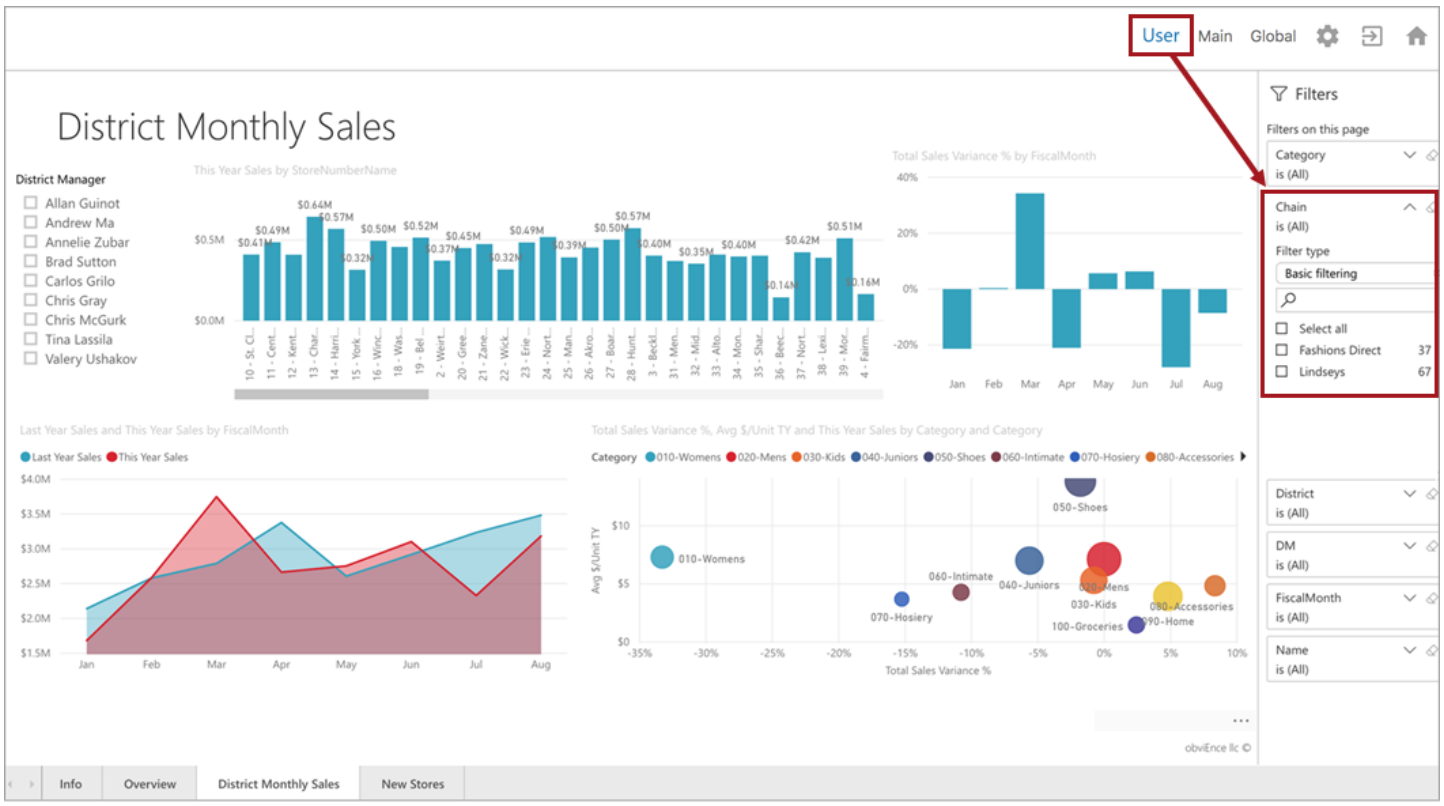
4. Log into the Web UI with one of the previously created district manager users, e.g., Carlos Grilo, who is a member of the Fashions Direct District Managers group. Power BI will filter everything appropriately.



5. Log in with a Fashions Direct super user to examine more data. e.g., Tina Lassila, who is a member of both the Fashions Direct District Managers and Fashions Direct Super User groups. If Tina were only in the Fashions Direct District Managers group, they would only access data filtered on their username. But, since they are also a member of the Fashions Direct Super User, Tina is entitled to access all Fashions Direct data.



6. Finally, log in with a Power BI Super User and check that you can examine both Fashions Direct and Lindseys data.


User
Main
Global
District Monthly Sales
District Manager
Allan Guinot
Andrew Ma
Annelie Zubar
Brad Sutton
Carlos Grilo
Chris Gray
Chris McGurk
Tina Lassila
Valery Ushakov
This Year Sales by StoreNumberName
Total Sales Variance % by FiscalMonth
Filters
Filters on this page
Category
is (All)
Chain
is (All)
Filter type
Basic filtering
Select all
Fashions Direct 37
Lindseys 67
Last Year Sales and This Year Sales by FiscalMonth
Last Year Sales
This Year Sales
Total Sales Variance %, Avg $/Unit TY and This Year Sales by Category and Category
Category
010-Womens
020-Mens
030-Kids
040-Juniors
050-Shoes
060-Intimate
070-Hosiery
080-Accessories
District
is (All)
DM
is (All)
FiscalMonth
is (All)
Name
is (All)
obviEnce llc ©
Info
Overview
District Monthly Sales
New Stores

Power BI Authentication Configuration

Microsoft follows a different integration approach than what is used for the visual integrations between STEP and Tableau or Qlik. Power BI requires the use of Microsoft's **API** to authenticate, show, interact with, and filter reports.

The visual integration between STEP and Power BI uses an 'app owns data' scenario instead of 'user owns data.' As a result, *individual users* who log into the Web UI to view Power BI information do not need a Power BI account; STEP is configured with the relevant credentials to authenticate the Power BI service.

Note: Though *individual users* do not need a Power BI account to view Power BI information in the Web UI, a licensed instance of Power BI is required for the visual integration with STEP.

Service Principal Authentication Setup Overview

The visual integration between STEP and Power BI uses the **service principal** method of authentication, which allows the application (in this case, STEP) to log in to the Power BI service on behalf of the user that is using service principal credentials. The service principal authenticates the application by using an **application ID** and an **application secret** and is configured with properties added to the sharedconfig.properties file on the STEP application server.

A general overview of how to set up Power BI using the service principal is as follows. For more detailed information, refer to the following Power BI help page: https://docs.microsoft.com/en-us/power-bi/developer/embed-service-principal#get-started-with-a-service-principal.

1. Set up Power BI with the service principal in **Microsoft Azure Active Directory (AD)**. In AD, you will perform tasks that including the following:
   - Register a server-side web application to use with Power BI.
   - Create a security group and add the application you created (e.g., Stibo Power BI Embedded) to that security group.

   For more information, refer to the following Microsoft Azure AD help page: https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-groups-create-azure-portal.
2. As a **Power BI admin**, enable service principal in the Developer settings in the **Power BI admin portal**. To access the Power BI admin portal, visit: https://app.powerbi.com/admin-portal.

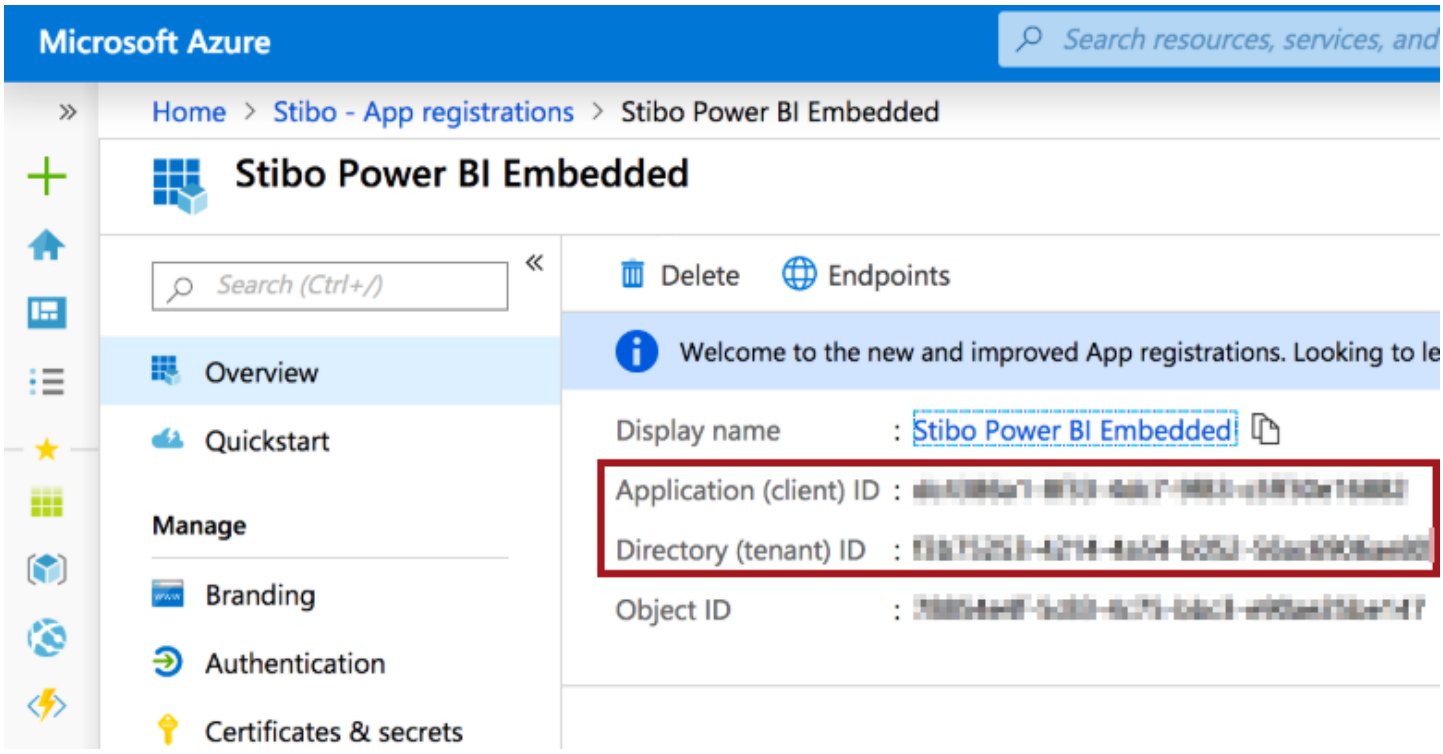
   Note: To become a Power BI admin, users must belong to the Power BI administrators group in Microsoft Azure.
3. In **Power BI**, set up your Power BI environment and add the service principal as an admin to the relevant Power BI workspace. For more information, refer to https://docs.microsoft.com/en-us/power-bi/developer/embed-service-principal.
4. Configure **STEP** to connect to the Power BI Service using the service principal. This is done by adding the configuration properties listed in the next section of this topic to the sharedconfig.properties file on your application server.

Configuration Properties

The following table explains the configuration properties required to enable authentication between STEP and Power BI.

Configuration Property	Description
PowerBI.TenantID	Power BI Tenant ID. This should be set to the 'Directory (tenant) ID' specified on the Overview screen of the App Registration in Azure Active Directory.
PowerBI.ServicePrincipal.ClientID	Power BI Service Principal Client ID. This should be set to the 'Application (client) ID' specified on the Overview screen of the App Registration in Azure Active Directory.
PowerBI.ServicePrincipal.ClientSecret	Power BI Service Principal Client Secret. This should be set to the value of the Client Secret specified on the 'Certificates & secrets' screen of the App Registration in Azure Active Directory.

The below screenshot illustrates where to locate values for the `PowerBI.TenantID` and `PowerBI.ServicePrincipal.ClientID` configuration properties in the Microsoft Azure interface:



Export of Analytics Data for BI Tools

For users who have existing BI tools that they want to integrate with, Stibo Systems offers the following methods of extracting data from STEP that can be sent to downstream BI tools:

- **Audit Message Framework**: The Audit Message Framework (AMF) is a powerful message delivery solution that sends configurable messages to an external system of the users' choice, allowing data in STEP to be extracted and exposed so it can be processed for statistical analysis. The AMF includes an out-of-the-box Audit Message Receiver JDBC Delivery Plugin, which utilizes message queues that correspond to tables in the external JDBC database into which user-specified audit messages are written. The analysis of workflow data is the typical focus for users of the AMF.
- **JDBC Delivery Method:** The Java Database Connectivity (JDBC) delivery plugin feature can be configured to automatically (or manually) make STEP data available to an analytics platform, typically Tableau or Qlik. This feature effectively closes the loop of data and visualization by making master data viewable in a data analytics dashboard, which is in turn viewable in the Web UI. The analysis of product data is the typical focus for users of the JDBC delivery method. For information regarding proper setup of JDBC in conjunction with data analytics integration with the Web UI, refer to the **Analytics Using JDBC Example** topic in this guide. For additional information on JDBC data delivery, refer to the following topics:
    - **Exporting Data via JDBC with CSV Format** in the **Data Formats** section of the **Data Exchange** documentation
    - **JDBC Delivery Method** in the **Export Manager** section of the **Data Exchange** guide
    - **JDBC Delivery Method** in the **OIEP Delivery Methods** section of the **Outbound Integration Endpoints** guide

Audit Message Framework

Important: Audit Message Framework can be used with on-premises STEP deployments only. Currently, it does not work with SaaS deployments.

Companies are increasingly using data to improve agility, deliver personalized experiences, accelerate time-to-market, and increase customer satisfaction. The **Audit Message Framework**—a powerful message delivery solution that allows users to send configurable messages to external Cassandra or JDBC-compliant database systems—can help your business achieve these objectives by exposing and extracting data in STEP that can be processed for statistical analysis.

This data can be used to support the auditing of workflows and related data, such as helping users determine where objects in a workflow are spending most of their time, and why conditions fail for particular objects in a workflow. The resulting data can be analyzed and blended with data from third-party systems, via business intelligence (BI) tools, to provide valuable insights that identify issues and improve processes. The gathering and analysis of this data can also help improve legal compliance.

Prerequisites

To access the Audit Message Framework and functionality, the Analytics commercial license must be enabled on your system. Additional setup tasks and system configurations must also be performed by Stibo Systems Technical Services team upon initial setup. This includes installing and setting up internally required components like Kafka and ZooKeeper. Contact your account manager for more information about the Analytics commercial license and its associated components / system licenses.

About this Guide

The subtopics in this documentation section, listed below, provide an overview on how to get started with the Audit Message Framework, along with sample JavaScript workflow auditing business actions that can be used for entire workflows or on individual workflow transitions.

The information in the tables within the topics below are best viewed online.

- Audit Message Framework Functionality Overview
- Audit Message Framework Configuration Properties and Monitoring
- Audit Message Framework JavaScript Binds and Public JavaScript API Methods
- Audit Message Framework Example Message
- Audit Message Framework Database Data Type Mapping
- Audit Message Framework Example Database Output
- Audit Message Framework Workflow Auditing
  - Auditing an Entire Workflow
  - Auditing by Workflow Transition

The Audit Message Framework is also available to users of systems with the Analytics components enabled, which includes Web UI access to Power BI, Tableau, and Qlik. Refer to the **Visual Integration with External Analytics Tools** section of this guide for more information.

For additional information about the JavaScript API interface and creating JavaScript based Business Rules in STEP, refer to the **Scripting API** section of the **STEP API** documentation. For additional information on Audit Message Receiver plugins, refer to the **Extension API** section of the **STEP API** documentation. The STEP API Documentation is available at [system]/sdk and is also accessible from the Start Page of your STEP instance.

Audit Message Framework Functionality Overview

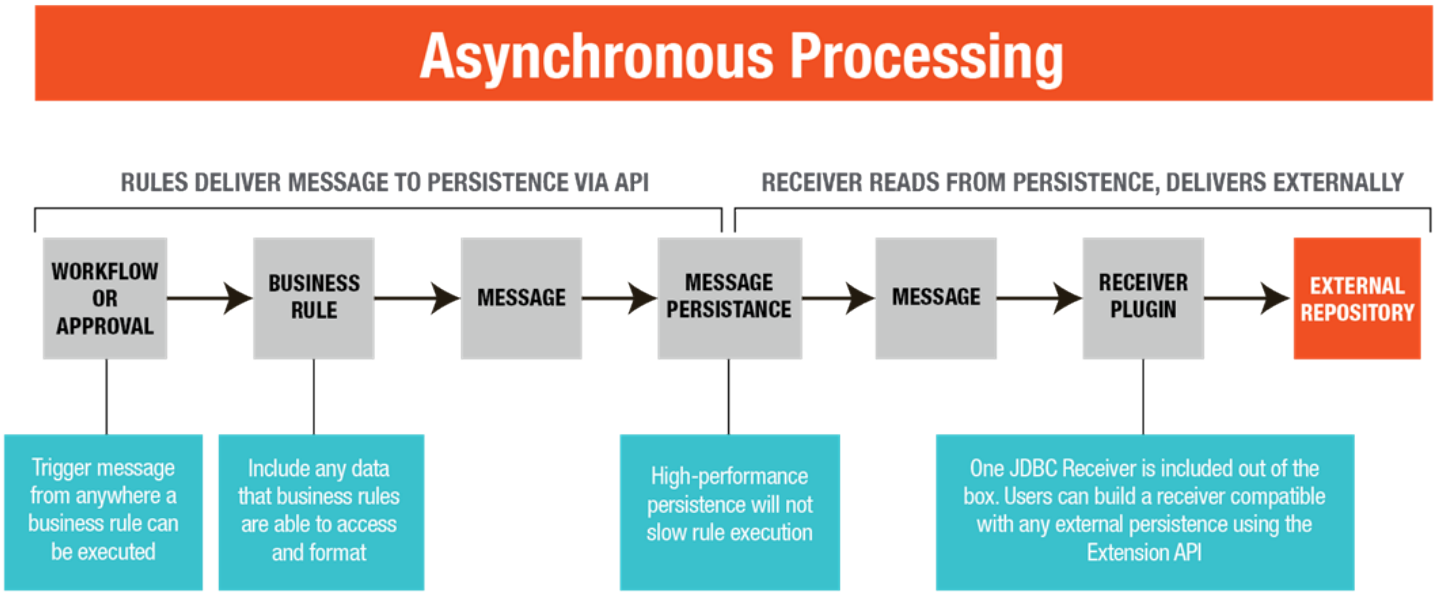
The Audit Message Framework uses the following features and functionality to process and send messages:

- Public JavaScript API methods to audit and persist STEP data, including workflow status and events. The public API methods can be used from custom code or from within JavaScript business actions.
- A simplified messaging system to deliver data to an external database via Java Database Connectivity (JDBC) and Cassandra.
- A public API interface for custom extensions.

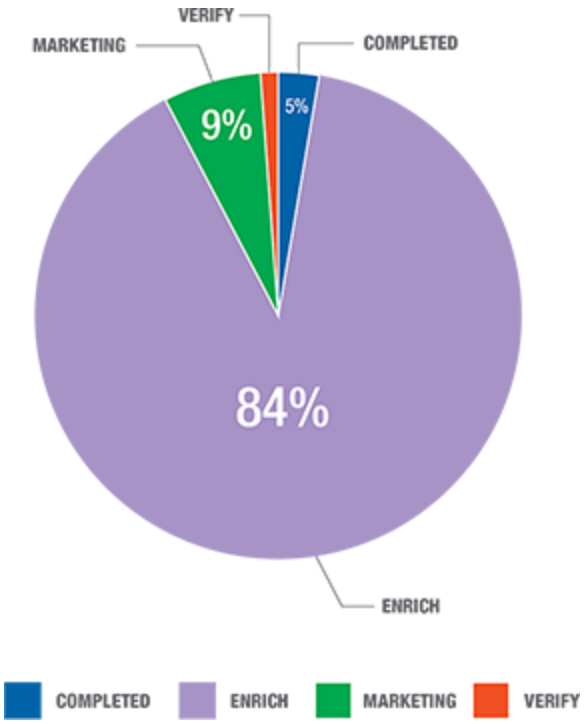
The graphic below provides a more detailed view of how the Audit Message Framework functions. A high-level summary is as follows:

1. Business rules are executed to deliver messages asynchronously to persistence via the API
2. The out-of-the-box JDBC receiver or Cassandra receiver reads these messages from persistence
3. The messages are delivered to an external repository (i.e., another database)

Note: Though the Audit Message Framework has the capability to send messages synchronously, it is recommended to send messages *asynchronously* to help minimize the performance impact on the callers of the interface.



After the information lands in the external repository, the information will likely continue its way downstream into a BI tool, such as Tableau or Qlik. Within these tools, users can visually digest the extracted information in the form of graphics and charts, such as the pie chart below. This example chart details the percentage of objects in specified workflow states at a given moment in time.


VERIFY
MARKETING
COMPLETED
9%
5%
84%
ENRICH
COMPLETED
ENRICH
MARKETING
VERIFY

Audit Message Framework Configuration Properties and Monitoring

Once the Audit Message Framework (AMF) is activated and installed, several configuration properties must be added to your sharedconfig.properties file on the STEP application server. This topic describes the most common configurations that enable the **Audit Message Receiver JDBC Delivery Plugin** and the **Audit Message Receiver Cassandra Delivery Plugin**, both of which ship out-of-the-box with the AMF.

Since the AMF solution provides the flexibility for users to create their own plugins, these configuration settings are not required if a different plugin is used. Users may choose to write their own plugins if they want to deliver messages to a location that cannot be written to via JDBC or Cassandra, e.g., directly to the file system, or to a MongoDB database.

Audit Message Receiver JDBC Delivery Plugin Configuration Properties

The following tables lists the configuration properties for the Audit Message Receiver JDBC Delivery Plugin and their descriptions:

Configuration Property	Description
`AuditMessaging.JDBCReceiver.DriverPath`	Full path to JDBC driver jar required to connect to the JDBC database.
`AuditMessaging.JDBCReceiver.DriverClass`	Name of the JDBC driver class required to connect to the JDBC database.
`AuditMessaging.JDBCReceiver.URL`	URL, host name and port number, to allow access to the JDBC database instance.
`AuditMessaging.JDBCReceiver.UserName`	Name of user to use when accessing JDBC database instance.
`AuditMessaging.JDBCReceiver.Password`	Password for user to use when accessing JDBC database instance.
`AuditMessaging.JDBCReceiver.TableName`	Comma-separated list of the names of database tables to insert audit messages into in JDBC database instance. Each table name can be preceded by a **topic** using the format 'myTopic = myDatabaseTable.' If no topic is specified for the database table, the name of the database table will be used as the topic. For example, a property set to:

Configuration Property	Description
	`AuditMessaging.JDBCReceiver.TableName = MyTopic=MyDBTable1, MyDBTable2` will result in the topics 'MyTopic' and 'MyDBTable2.' Messages sent to a particular topic will be inserted into the corresponding database table. Valid characters for topics and table names are the ASCII alphanumerics, '.', '_', and '-'. More information on this configuration and how it is used with **topics** is provided in the **Audit Message Framework JavaScript Binds and Public JavaScript API Methods** topic in this guide.

Configuration Property Examples

The following is a sample configuration for a MySQL database version 8.0:

```
AuditMessaging.JDBCReceiver.DriverPath = C:/mysql-connector-java-8.0.12.jar
AuditMessaging.JDBCReceiver.DriverClass = com.mysql.cj.jdbc.Driver
AuditMessaging.JDBCReceiver.URL = jdbc:mysql://localhost:3306/sys
AuditMessaging.JDBCReceiver.UserName = user_name
AuditMessaging.JDBCReceiver.Password = password
AuditMessaging.JDBCReceiver.TableName = Audit=AuditMessagesDBTable
```

The following is a sample configuration for an ORACLE database version 11g:

```
AuditMessaging.JDBCReceiver.DriverPath = E:/oracle-jar/ojdbc6.jar
AuditMessaging.JDBCReceiver.DriverClass = oracle.jdbc.driver.OracleDriver
AuditMessaging.JDBCReceiver.URL = jdbc:oracle:thin:@//66.66.66.166:1521/somedb
AuditMessaging.JDBCReceiver.UserName = user_name
AuditMessaging.JDBCReceiver.Password = password
AuditMessaging.JDBCReceiver.TableName =
WorkflowAudit=WorkflowAuditDBTable,AnotherDBTable
```

Audit Message Receiver Cassandra Delivery Plugin Configuration Properties

The following tables lists the configuration properties for the Audit Message Receiver Cassandra Delivery Plugin and their descriptions:

Configuration Property	Description
`AuditMessaging.CassandraReceiver.KeySpaceName`	Name of the keyspace namespace used for data replication.
`AuditMessaging.CassandraReceiver.DataCenter`	Name of the data center you are connecting to. This is an optional setting. If not set, the default value of 'datacenter1' will be used.
`AuditMessaging.CassandraReceiver.URL`	URL, host name and port number, to allow access to the Cassandra database instance.
`AuditMessaging.CassandraReceiver.UserName`	Name of user to use when accessing the Cassandra database instance.
`AuditMessaging.CassandraReceiver.Password`	Password for user to use when accessing the Cassandra database instance
`AuditMessaging.CassandraReceiver.TableName`	Comma-separated list of the names of database tables to insert audit messages into in the Cassandra database instance. Each table name can be preceded by a **topic** using the format 'myTopic = myDatabaseTable.' If no topic is specified for the database table, the name of the database table will be used as the topic. For example, a property set to: `AuditMessaging.CassandraReceiver.TableName = MyTopic=MyDBTable1, MyDBTable2` will result in the topics 'MyTopic' and 'MyDBTable2.' Messages sent to a particular topic will be inserted into the corresponding database table. Valid characters for topics and table names are the ASCII alphanumerics, '.', '_', and '-'. More information on this configuration and how it is used with **topics** is provided in the **Audit Message Framework JavaScript Binds and Public JavaScript API Methods** topic in this guide.

Configuration Property Examples

The following is a sample configuration for a Cassandra database version 3.11.4 running locally:

```
AuditMessaging.CassandraReceiver.KeySpaceName = test_keyspace
AuditMessaging.CassandraReceiver.URL = 127.0.0.1:9042
AuditMessaging.CassandraReceiver.UserName = cassandra
AuditMessaging.CassandraReceiver.Password = cassandra
AuditMessaging.CassandraReceiver.TableName = Audit=AuditMessagesDBTable
```

Audit Messaging Monitoring in STEP System Administration

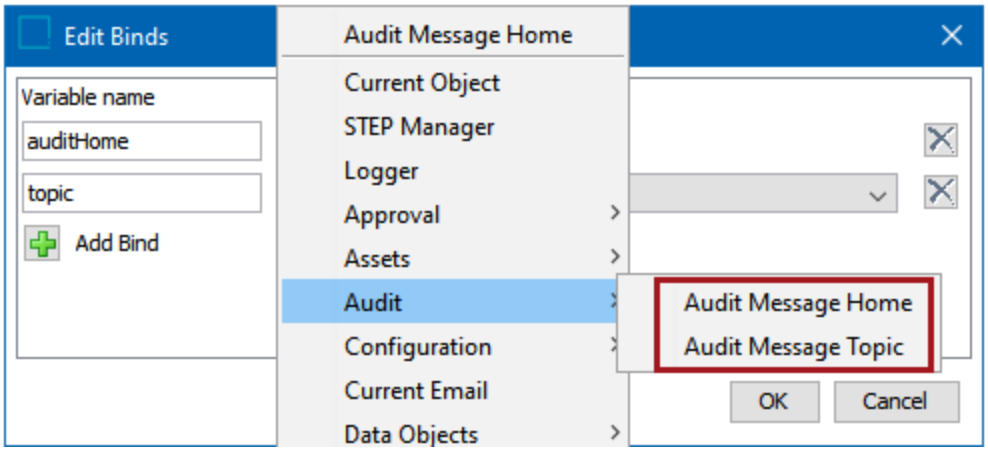
Two sensors are available in the Admin Portal to help with monitoring of the Audit Messaging Framework. These sensors are located in the Admin Portal on the **Monitoring** tab under Additional Links > Sensors > **Sensors for external monitoring**.

The Admin Portal is accessed by clicking the STEP System Administration link on your Start Page. For more information on the Admin Portal, refer to the **Administration Portal** documentation.

Audit Message Framework JavaScript Binds and Public JavaScript API Methods

The Audit Messaging Framework ships with two out-of-the box Audit Message Receiver plugins. These plugins listen for messages based on one or more **topics**, then output these messages to either an external JDBC or Cassandra database.

Two JavaScript **binds** are used to enable the business rules used to send messages within the Audit Message Framework—**Audit Message Home** and **Audit Message Topic**. These binds are located within the **Audit** category for 'Execute JavaScript' business actions.



These binds provide access to several JavaScript API **methods**, which are described in the below table:

JavaScript API Bind	JavaScript API Methods Used for Each Bind	Method Description
Audit Message Home	getTopicByID	Gets the Audit Message Topic ID that the receiver plugin will subscribe to. Returns null if there are no Audit Message Receiver plugins registered to receive this Audit Message Topic.
Audit Message Topic	sendMessage	Sends an audit message to an Audit Message Topic synchronously. If any errors occur with the send, a RuntimeException will be thrown. **Note:** As the message will be sent synchronously, it will wait for the message to be sent to the message queue successfully before returning the success code.
	sendMessageAsync	Sends an audit message to an Audit Message Topic asynchronously. If

JavaScript API Bind	JavaScript API Methods Used for Each Bind	Method Description
		any errors occur with the message send, error information will be written to the step.0.log, but the message will be lost. **Note:** This is the recommended call, which helps minimize the performance impact on the callers of the interface.
	getTopicID	Gets the Audit Message Topic ID.

Topics

Topics (also referred to as Audit Message Topics) are different queues within the message framework that enable the system to route and process messages dependent on their origin, i.e., messages from different topics could be written to different locations and have different content. Topics are defined by (and subsequently subscribed to by) the Audit Message Receiver plugins, which can handle the messages as desired. The list of topics is generated by the Audit Message Receiver plugins installed on the system.

When used with the out-of-the-box Audit Message Receiver plugins, a topic is a message queue that corresponds to a table in the receiving JDBC or Cassandra database into which user-specified audit message(s) / analytics data will be written.

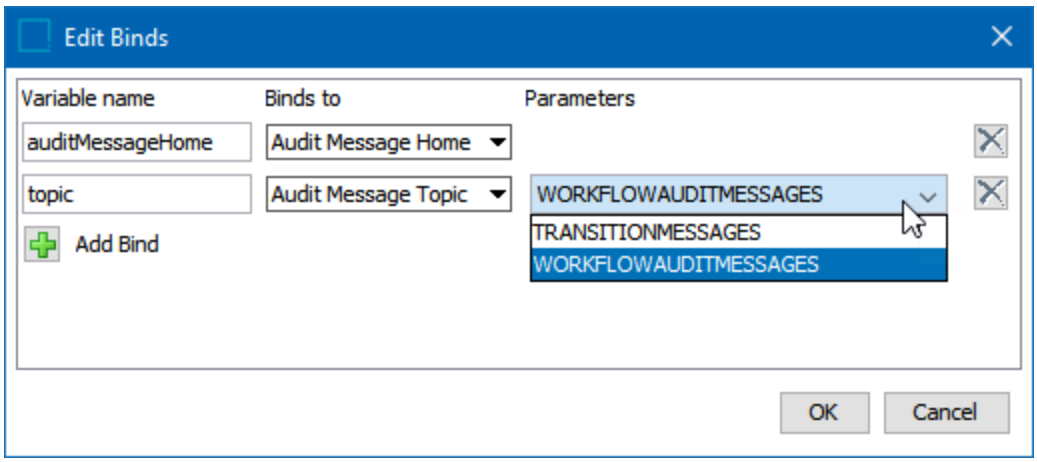
Note: Messages can be written to different tables within a single database when using the out-of-the-box Audit Message Receiver Delivery Plugins. Functionality is not supported to write messages to multiple databases of the same kind.

An Audit Message Receiver plugin can define multiple topics that it is interested in, and multiple Audit Message Receiver plugins can define the same topic if they so choose. Messages for a particular topic will be delivered to all plugins that subscribe to that topic.

When configuring business rules for the Audit Message Framework, the **sendMessage** and **sendMessageAsync** methods are available by binding directly to a particular topic. Topics are selected from the **Parameters** dropdown list of the **Audit Message Topic** bind.

When using the Audit Message Receiver JDBC Delivery Plugin, to add topics to the Parameters dropdown, they must be specified in the sharedconfig.properties file in the `AuditMessaging.JDBCReceiver.TableName` property. When using the Audit Message Receiver Cassandra Delivery Plugin, the `AuditMessaging.CassandraReceiver.TableName` property must be used.

Refer to the **Audit Message Framework Configuration Properties and Monitoring** topic for more information on how to configure these properties.

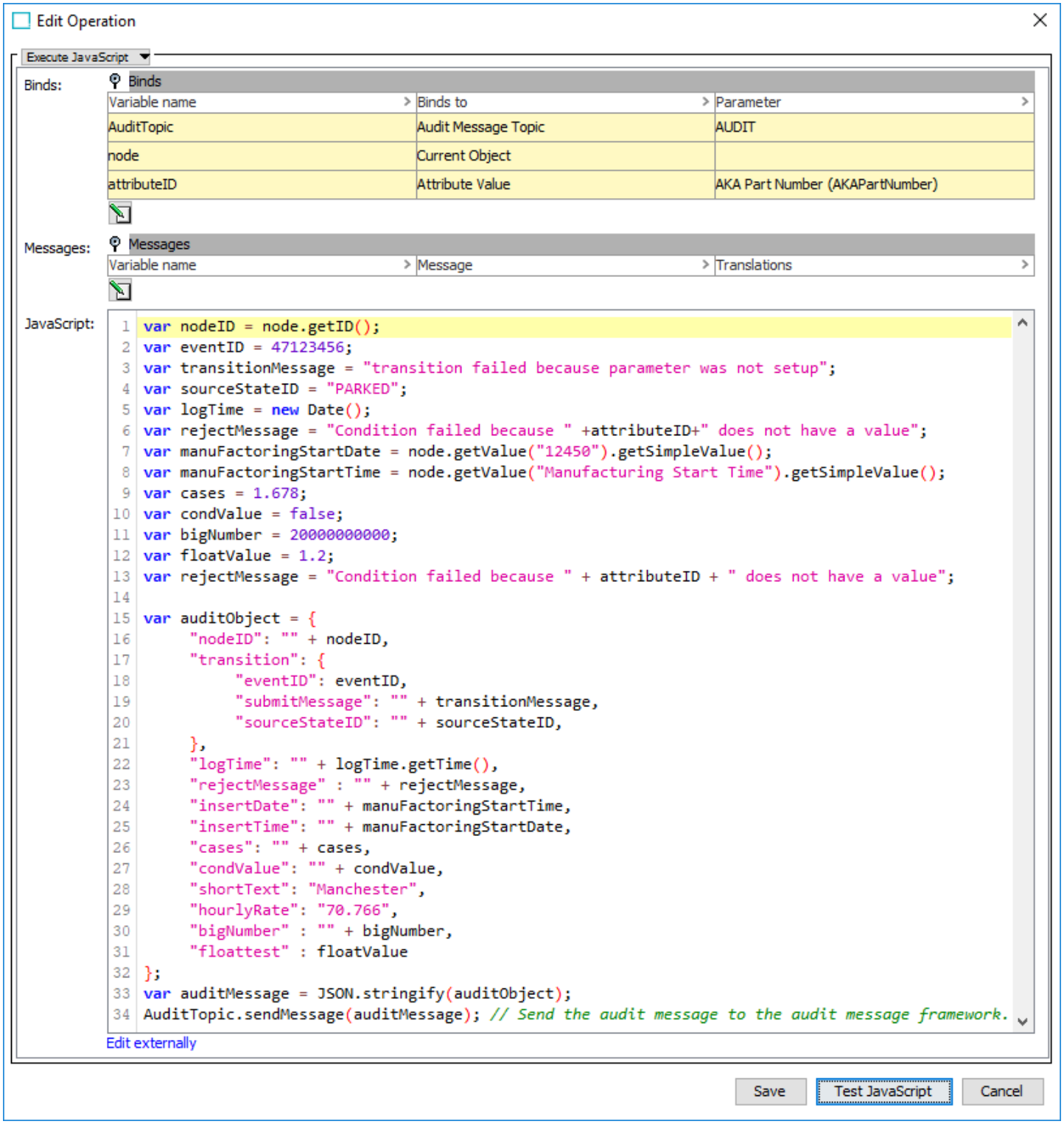

Edit Binds
Variable name
Binds to
Parameters
auditMessageHome
Audit Message Home
topic
Audit Message Topic
WORKFLOWAUDITMESSAGES
TRANSITIONMESSAGES
WORKFLOWAUDITMESSAGES
Add Bind
OK
Cancel

Audit Message Framework Example Message

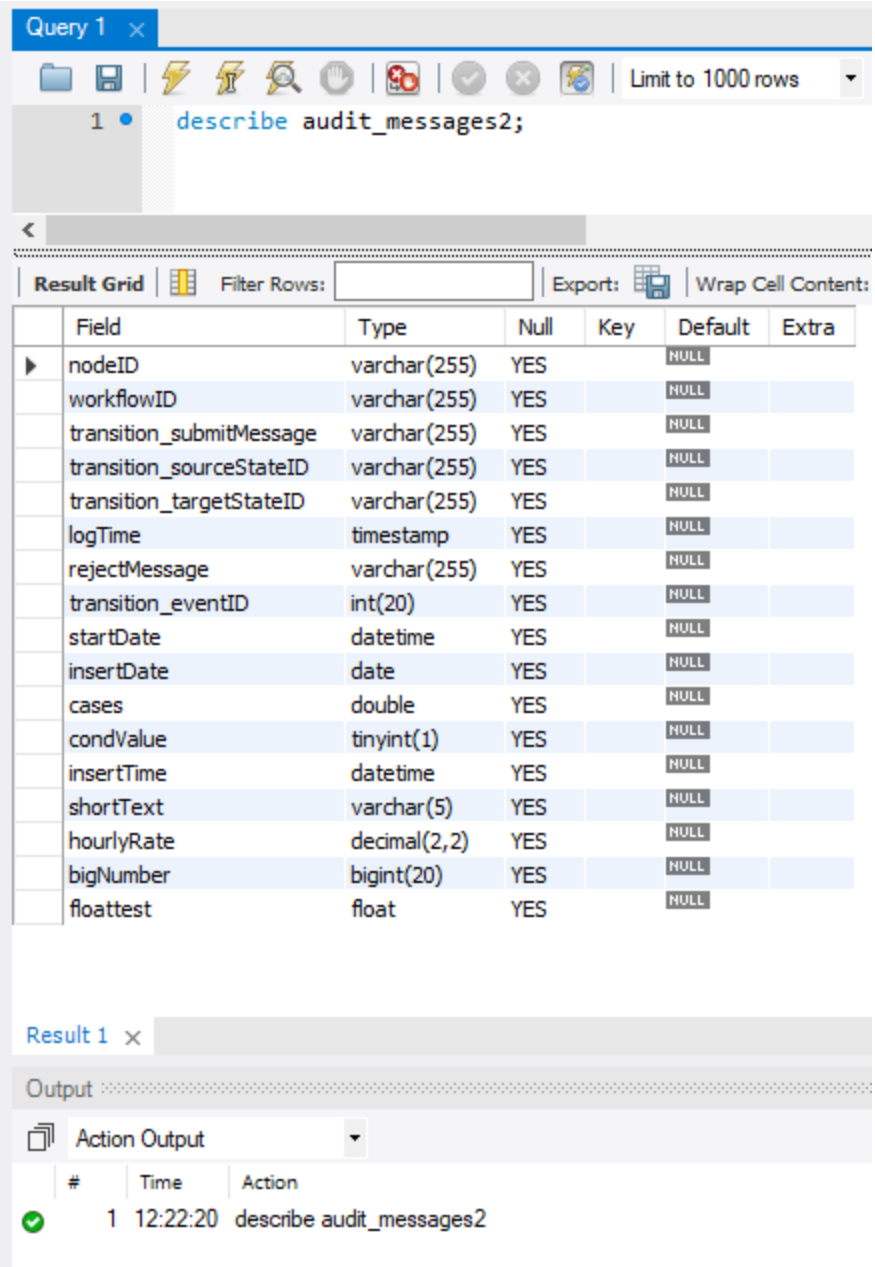
To send an audit message, it is necessary to either create a JavaScript bind to an audit message topic or to find an audit message topic via the API. For more information on these binds and topics, refer to the **Audit Message Framework JavaScript Binds and Public JavaScript API Methods** topic.

Once a connection has been made to a topic, a JSON structure object must be created, which maps JSON field values to database table record values.

The functionality for both the Audit Message Receiver JDBC Delivery Plugin and the Audit Message Receiver Cassandra Delivery Plugin are the same. The following screenshot shows an example message for the Audit Message Receiver JDBC Delivery Plugin receiver:



The topic is configured via the `AuditMessaging.JDBCReceiver.TableName` setting to map to the table called AUDIT_MESSAGES2. This external database table is shown in the screenshot below, which shows the view of a database query in a third-party SQL client. For information about this configuration, refer to the **Audit Message Framework Configuration Properties and Monitoring** topic.



For additional information about mapping JSON field names to database column names in AMF, refer to the next topic in this guide, **Audit Message Framework Database Data Type Mapping**.

Audit Message Framework Database Data Type Mapping

The Audit Message Framework supports two types of database mapping, one for each of the supported integration plugins—the **Audit Message Receiver JDBC Delivery Plugin** for JDBC-compliant databases (such as Oracle and MySQL) and the **Audit Message Receiver Cassandra Delivery Plugin** for Cassandra databases. This topic outlines the mapping of these data types for both plugins.

JDBC Database Data Type Mapping

When sending JSON-formatted messages from STEP to an external JDBC-compliant database, the table field names in STEP are directly mapped to the database column names in the external database, ignoring case. A JSON field that is part of a JSON object has its parent name prepended to its name with an additional '_' character. So, for example, in the JavaScript, the JSON field 'sourceStateID' will be exactly mapped to a table column named 'transition_sourceStateID'. The JSON field value will be inserted in the matching table column value if a type match can be made. If a match cannot be made, a warning message is written to the STEP log.

The following type conversions are supported for JDBC database mapping:

Database Field Type	Mapping
`BIGINT`	Field value can be a String, Integer, or Long. If a String, it is assumed to be a Long number value.
`BIT`	Field value can be a String or a Boolean. If a Boolean, the field is set to 1 (true) or 0 (false). If a String, it is assumed to be either the text 'true' or the text 'false.'
`BOOLEAN`	Field value can be a String or a Boolean. If a String, it is assumed to be either the text 'true' or the text 'false.'
`DATE`	Field value can be a String and is assumed to adhere to the date format standard ISO_DATE.
`DECIMAL`	Field value can be a String or a Double. If a String, it is assumed to be a Double number value.
`DOUBLE`	Field value can be a String or a Double. If a String, it is assumed to be a Double number value.
`FLOAT`	Field value can be a String or a Float. If a String, it is assumed to be a Float number value.
`INTEGER`	Field value can be a String or an Integer. If a String, it is assumed to be an Integer number value.

Database Field Type	Mapping
`REAL`	Field value can be a String or a Double. If a String, it is assumed to be a Double number value.
`TIME`	Field value can be a String, Integer, or Long. If an Integer or Long, it is assumed to be the number of milliseconds since January 1, 1970. If a String, it is assumed to adhere to the date format standard ISO_DATE_TIME.
`TIMESTAMP`	Field value can be a String, Integer, or Long. If an Integer or a Long, it is assumed to be the number of milliseconds since the January 1, 1970. If a String, it is first assumed to be a Long value representing the number of seconds since January 1, 1970. If not, then it is assumed to be a date String adhering to the date format standard ISO_LOCAL_DATE_FORMAT.
`VARCHAR`	Field value must be a String.

JDBC Database Record Updating

For JDBC messages, the target system is checked for a field called MD_ID for Oracle databases and _ID for non-Oracle databases, which is assumed to be the key field. If a database column name and a JSON field name both exist that matches this, then the first search is for a record in the table with the matching JSON field value. If a match is found, then the record is updated and the audit message is inserted into the target database.

Cassandra Database Data Type Mapping

If sending a message to a Cassandra database, the JSON field values are mapped as follows. The first column contains the name of the Cassandra data type.

Database Field Type	Mapping
`ASCII`	Field value must be a String.
`BIGINT`	Field value can be a String, Integer, or Long. If a String, it is assumed to be a Long number value.
`BOOLEAN`	Field value can be a String or a Boolean. If a String, it is assumed to be either the text 'true' or the text 'false.'
`DATE`	Field value must be a String, which is assumed to follow the ISO_DATE format.

Database Field Type	Mapping
`DECIMAL`	Field value can be a String or a Double. If a String, it is assumed to be a Double number value.
`DOUBLE`	Field value can be a String or a Double. If a String, it is assumed to be a Double number value.
`FLOAT`	Field value can be a String, Double, or Float. If a String, it is assumed to be a Float number value.
`INET`	Field value must be a String, which is assumed to be a hostname.
`INT`	Field value can be a String or an Integer. If a String, it is assumed to be an Integer value.
`SMALLINT`	Field value can be a String or an Integer. If a String, it is assumed to be a Short.
`TEXT`	Field value must be a String.
`TIME`	Field value can be a String, Integer, or Long. If a String, it is assumed to be a time in HH:mm:ss format. If a number, then it is assumed to be the number of milliseconds since midnight.
`TIMESTAMP`	Field value can be a String, Integer, or Long. If a String, then it is assumed to be a Long. The value is the number of milliseconds since January 1, 1970.
`TINYINT`	Field value can be a String, Boolean, or Integer. If a Boolean, the field is either set to 1 (true) or 0 (false). If a String, it is assumed to be either the text 'true' or the text 'false.'
`VARINT`	Field value can be a String, Integer, or Long. If a String, then it is assumed to be a BigInteger value.

Cassandra Database Record Updating

For Cassandra messages, messages are always inserted and rely on the database providing the update or insert functionality. This can be achieved by making a column a primary key, in which case the database will always update rather than insert if it finds a matching record value. To always insert, create a primary key using a UUID or TIMEUUID data type. These always create new unique values and do not provide a matching JSON field value.

UPSERT Functionality for JDBC Database Tables

When using the Audit Message Receiver JDBC Delivery Plugin, the default behavior when processing messages is to insert each message into the database as a new database entry. An alternative is to 'upsert' messages by using the UPSERT command, which allows for existing records to be overwritten if it is determined that a record for the same object has already been written. This feature can help reduce the maintenance and manipulation required to update and obtain the latest status for a dataset that is being queried; for example, to obtain a simple status of workflows.

To support UPSERT, an '_ID' field (for non-Oracle databases) or 'MD_ID' (for Oracle databases) must be added to the JSON message that is sent from STEP to downstream systems. To enable the field, the corresponding table in the users' external database must also have an _ID / MD_ID column defined. When processing an audit message, the external database table will be checked for a record with the same _ID /MD_ID field value as the new audit message. If a match is found, the record is updated; otherwise a new record is created.

As an example, the field in the outgoing message that contains the '_ID' key could be set to be a combination of the nodeID and workflowID:

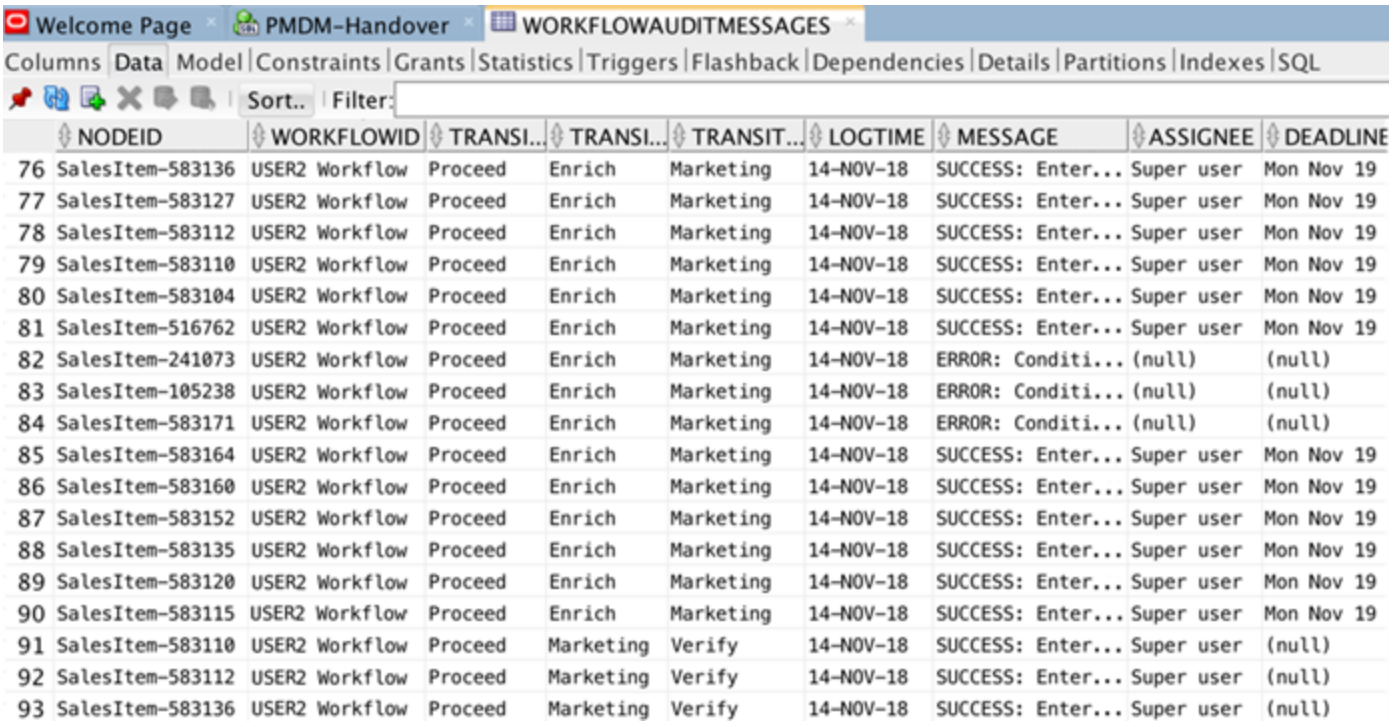
```
var auditObject = {
   "_ID": "" + nodeID + "_" + workflowID,
   ...
}
```

Note: As indicated above, the update column is called MD_ID for an ORACLE database and _ID for a non-ORACLE database. The _ID / MD_ID column must also be indexed in the external database to make it searchable, since the system searches existing records before making the decision to insert a new record or overwrite an existing one. Once a record is overwritten, the log timestamp can be used to determine if the record is old or new.

Audit Message Framework Example Database Output

The out-of-the-box Audit Message Receiver JDBC Delivery Plugin maps JSON fields to column values in an external database via a JDBC interface. The messages sent via the **sendMessage** interfaces for topics to which the plugin subscribes are assumed to be a stringified JSON structure (i.e., a JSON structure converted to a string). When the plugin receives a message, it will be converted back to a JSON structure and processed (in this case, written to a database via JDBC).

Below is an example output of audit messages sent to an external Oracle database table. Multiple target tables are supported. Mapping from the database tables within STEP to the external database is performed by JSON object Key:Value mapping to these database table columns.


Welcome Page | PMDM-Handover | WORKFLOWAUDITMESSAGES

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. Filter:

	NODEID	WORKFLOWID	TRANSI...	TRANSI...	TRANSIT...	LOGTIME	MESSAGE	ASSIGNEE	DEADLINE
76	SalesItem-583136	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
77	SalesItem-583127	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
78	SalesItem-583112	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
79	SalesItem-583110	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
80	SalesItem-583104	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
81	SalesItem-516762	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
82	SalesItem-241073	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	ERROR: Conditi...	(null)	(null)
83	SalesItem-105238	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	ERROR: Conditi...	(null)	(null)
84	SalesItem-583171	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	ERROR: Conditi...	(null)	(null)
85	SalesItem-583164	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
86	SalesItem-583160	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
87	SalesItem-583152	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
88	SalesItem-583135	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
89	SalesItem-583120	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
90	SalesItem-583115	USER2 Workflow	Proceed	Enrich	Marketing	14-NOV-18	SUCCESS: Enter...	Super user	Mon Nov 19
91	SalesItem-583110	USER2 Workflow	Proceed	Marketing	Verify	14-NOV-18	SUCCESS: Enter...	Super user	(null)
92	SalesItem-583112	USER2 Workflow	Proceed	Marketing	Verify	14-NOV-18	SUCCESS: Enter...	Super user	(null)
93	SalesItem-583136	USER2 Workflow	Proceed	Marketing	Verify	14-NOV-18	SUCCESS: Enter...	Super user	(null)

Audit Message Framework Workflow Auditing

By using the **Audit Message Topic** bind, JavaScript business rules can be applied to entire workflows or to individual workflow transitions to send audit messages. The JavaScript creates a JSON message object and uses the bind's API **sendMessageAsync** method. The JSON message is received and handled by the plugin associated with a specific topic. The examples in this guide use the out-of-the-box Audit Message Receiver JDBC database delivery plugin, but the functionality is the same when using the Audit Message Receiver Cassandra database delivery plugin.

Note: The JSON message format is required for both the Audit Message Receiver JDBC Delivery plugin and Audit Message Receiver Cassandra Delivery plugin. A custom plugin could handle messages in an entirely different format if desired (e.g., a comma separated list).

The topics in this documentation section provide configuration instructions and sample JavaScript code for auditing workflows both at the workflow-wide level and at the individual transition level:

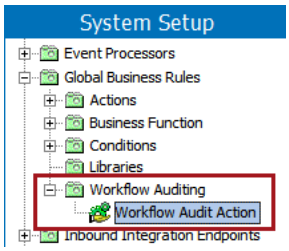
- The **Auditing an Entire Workflow** topic provides information about the default 'Workflow Audit Action' business action that is automatically created in STEP when the Audit Message Framework component is activated on your system.
- The **Auditing by Workflow Transition** topic provides two sample audit message business actions that can be used to audit individual workflow transitions. One example captures information about a workflow business condition failure, and the second captures a workflow state 'on entry' audit message.

Auditing an Entire Workflow

To audit an entire workflow using one single business action, a workflow-wide audit message business action can be applied, enabling the action to be evaluated and log messages for every transition within the workflow. When using a workflow-wide action, there is no need to apply an audit action to each transition individually, which can be a time-consuming task when configuring complicated workflows.

When the Audit Message Framework component is activated on your system, an audit message business action, **Workflow Audit Action** (ID = AuditMessaging.WorkflowAuditAction), is created in System Setup under the 'Global Business Rules' folder in the 'Workflow Auditing' subfolder. This business action contains JavaScript code that can be used to enable analytics for any workflow out of the box.

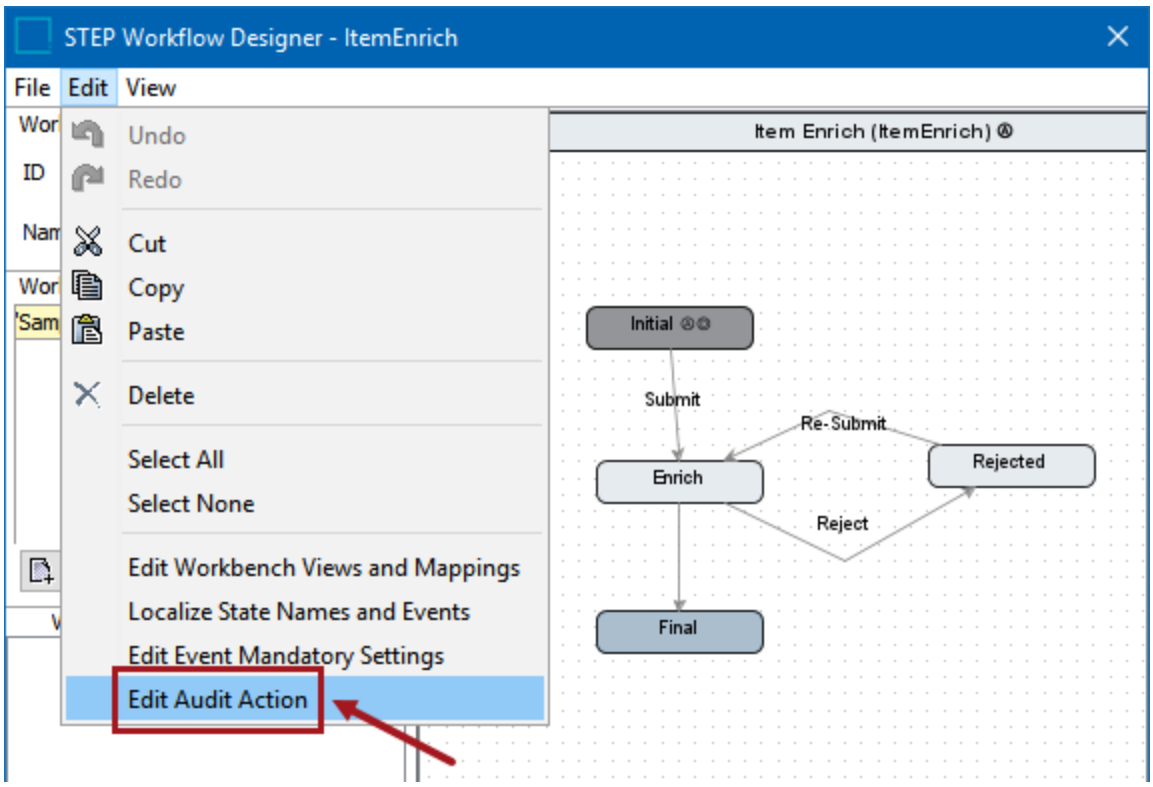
Note: If a business action with the ID 'AuditMessaging.WorkflowAuditAction' already exists on your system, the action will not be moved to the 'Workflow Auditing' subfolder upon installation of the Audit Message Framework. The action will be kept where it is.



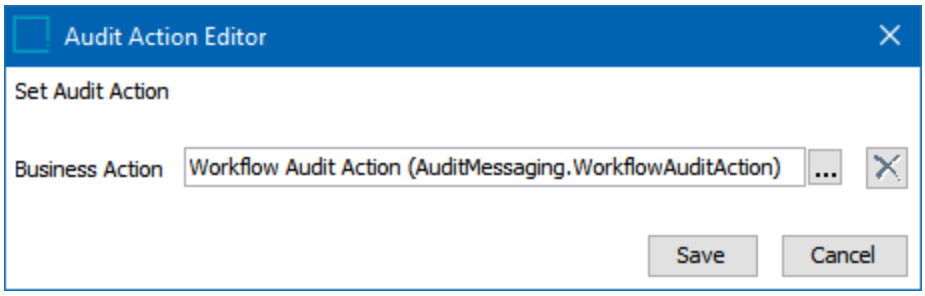
Applying an Audit Message Business Action to an Entire Workflow

Workflow-wide audit message business actions are assigned to workflows on a workflow-by-workflow basis. To apply a workflow-wide audit message business action:

1. Navigate to the relevant workflow in System Setup, then right-click and select **Edit Workflow**.
2. In the STEP Workflow Designer, click Edit > **Edit Audit Action**.



3. Clicking 'Edit Audit Action' displays the **Audit Action Editor** dialog. Click the ellipsis button (...) to select the desired audit message business action. In this example, the out-of-the-box 'Workflow Audit Action' business action is shown.



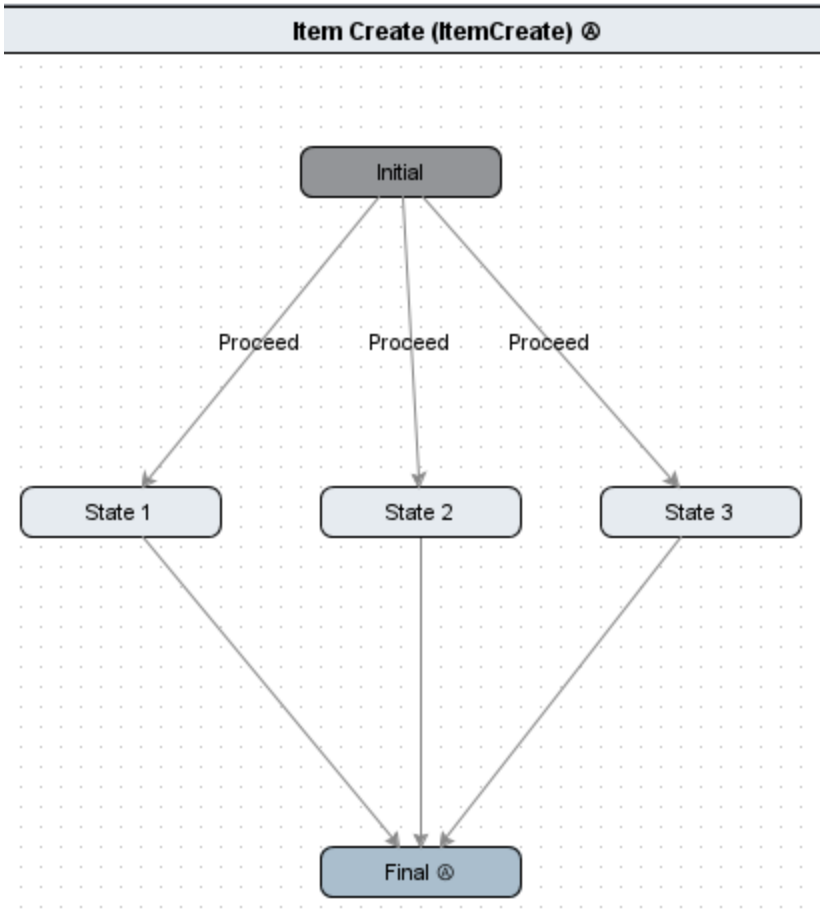
4. Click **Save** to apply the audit action to the workflow.

Workflow Audit Action Transition Evaluation

Once applied, the default audit message business action is automatically triggered for each workflow transition and runs during the evaluation phase of transitions, i.e., it is run after the transitions have been evaluated. This ensures access to the reject message(s) *before* any transition-local actions. The action also runs on the transition going into the initial state of the workflow, sending an audit message when an object enters the workflow.

Sample Audited Workflow

In the following example workflow, there are the 'Proceed' transitions, each going to a different state. When the user selects to Proceed, the transitions are evaluated in the following order: Proceed (State 1), Proceed (State 2), and Proceed (State 3). Each transition is evaluated until one is found that passes the transition conditions.



If all three Proceed transitions are evaluated to be rejected, the Workflow Audit Action will run once and will include information that the transition evaluation was rejected, along with the rejection messages from all three Proceed transitions.

If one of the Proceed transitions is evaluated to be accepted, the Workflow Audit Action will run once and will include information that the transition evaluation was successful.

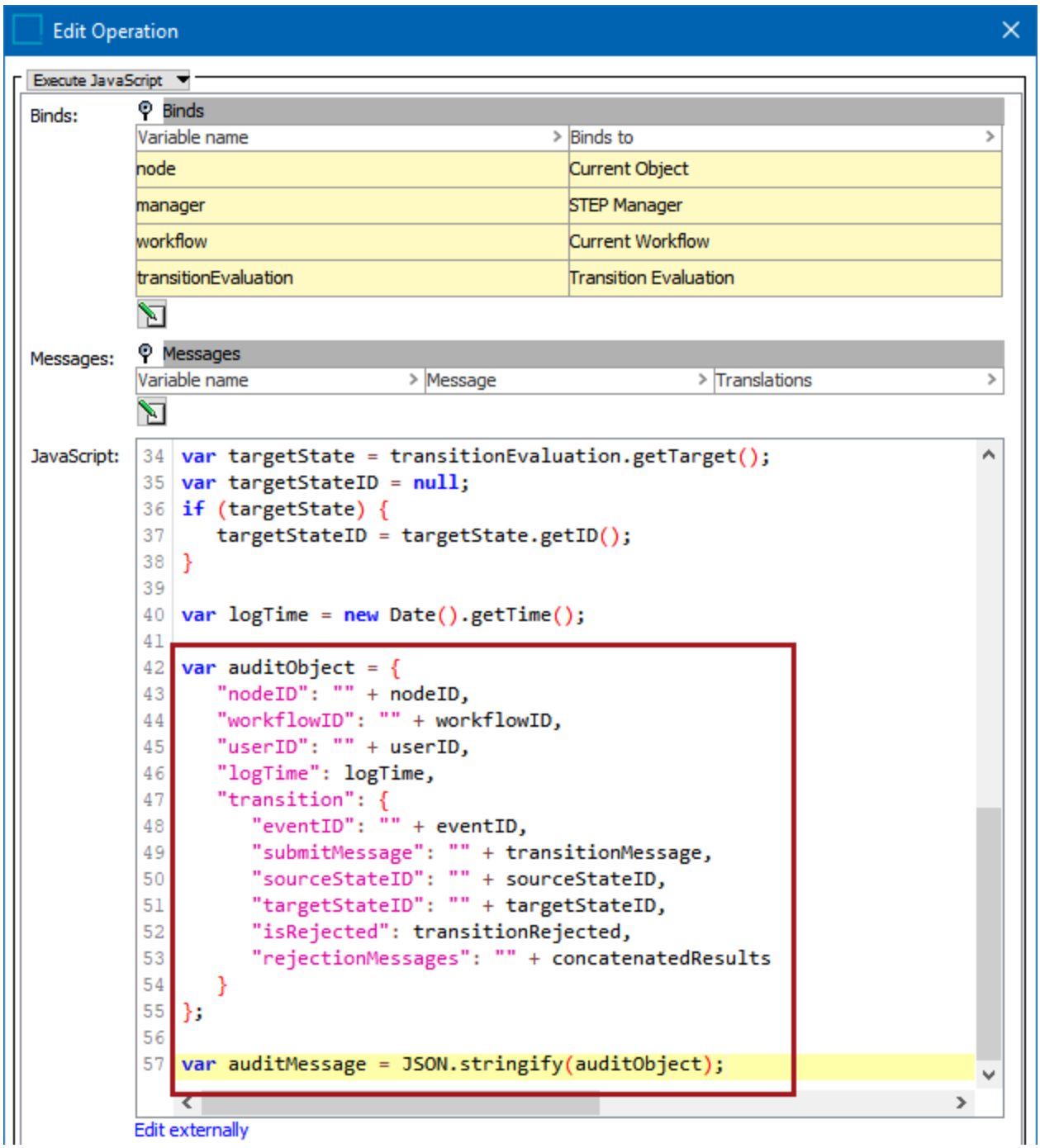
Important: The Workflow Audit Action should never be used to attempt to make changes to data. Any changes made data in the Workflow Audit Action are rolled back if the evaluated transition is rejected. Audit Messages are sent whether the transition is rejected or not.

Workflow Audit Action JavaScript Code

The following screenshot shows a portion of the JavaScript code for the out-of-the-box **Workflow Audit Action** business action. The highlighted section of the code is what builds the JSON object that is sent as the audit message. The JSON message is received and handled by the plugin associated with a specific topic; in this case, the Audit Message Receiver JDBC database delivery plugin.

The code in its entirety is provided after the screenshot.

Important: The Workflow Audit Action will not work directly out of the box; a **bind** must be made to the relevant audit messaging topic in the external JDBC database table before this action can produce an audit message. Refer to the **Audit Message Framework JavaScript Binds and Public JavaScript API Methods** topic for more details on Audit Message Framework binds.



Script

```
var nodeID = node.getID();
var userID = manager.getCurrentUser().getID();
var workflowID = workflow.getID();

var event = transitionEvaluation.getEvent();
```

```
var eventID = null;
if (event != null) {
   eventID = event.getID();
}

var transitionMessage = transitionEvaluation.getMessage();
var transitionRejected = transitionEvaluation.isRejected();
var resultMessages = transitionEvaluation.getResultMessages();

if (resultMessages.size() === 0) {
   var concatenatedResults = null;
} else {
   var concatenatedResults = "Evaluation Results (" + resultMessages.size() + "): ";
}

var resultMessageIter = resultMessages.iterator();
while (resultMessageIter.hasNext()) {
   concatenatedResults = concatenatedResults + resultMessageIter.next() + "; ";
}

var sourceState = transitionEvaluation.getSource();
var sourceStateID = null;
if (sourceState) {
   sourceStateID = sourceState.getID();
}

var targetState = transitionEvaluation.getTarget();
var targetStateID = null;
if (targetState) {
   targetStateID = targetState.getID();
}

var logTime = new Date().getTime();

// var auditObject = {
//    "_ID": "" + nodeID + "_" + workflowID,
//    ...
// }
var auditObject = {
   "nodeID": "" + nodeID,
   "workflowID": "" + workflowID,
   "userID": "" + userID,
   "logTime": logTime,
   "transition": {
      "eventID": "" + eventID,
      "submitMessage": "" + transitionMessage,
      "sourceStateID": "" + sourceStateID,
```

```
      "targetStateID": "" + targetStateID,
      "isRejected": transitionRejected,
      "rejectionMessages": "" + concatenatedResults
   }
};

var auditMessage = JSON.stringify(auditObject);
```

Auditing by Workflow Transition

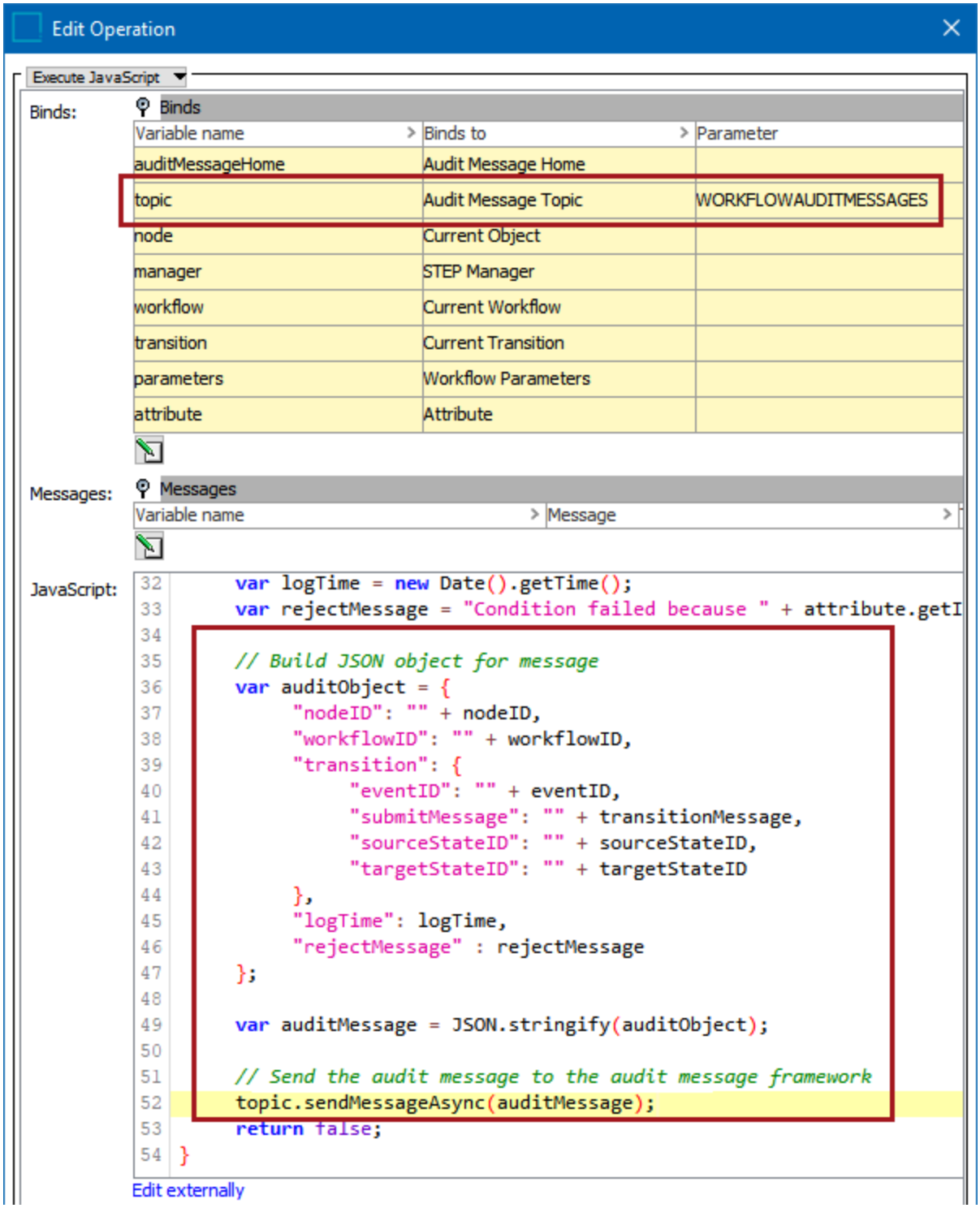
An alternative to auditing an entire workflow with a single default audit messaging business action is to audit workflows at the transition level. Audit business actions can be used on individual transitions *instead of* or *in addition to* a default workflow-wide business action. If used in addition to a default auditing business action, the actions applied at the transition level will be evaluated *after* the workflow-default rule.

The **Audit Message Topic** bind can be used to send messages like the two following examples, each of which creates a JSON message object and uses the bind's API **sendMessageAsync** method. The JSON message is received and handled by the plugin associated with a specific topic; in this case, the out-of-the-box Audit Message Receiver JDBC database delivery plugin.

Note: The sample business actions in this topic are not automatically created in your system upon activation of the Audit Message Framework component. Additionally, the provided JavaScript may need alterations to conform to your specific business requirements.

Sample JavaScript for a Workflow Business Condition Failure

The following script sends an audit message if a condition in a workflow fails:



Script

```
var attributeValue = node.getValue(attribute.getID()).getSimpleValue();
```

```
if (attributeValue) {
	// Attribute has a value. Everything is OK.
	return true;
} else {
	// Attribute does not have a value. Log the transition as failed.
	var nodeID = node.getID();
	var userID = manager.getCurrentUser().getID();
	var workflowID = workflow.getID();

	var event = transition.getEvent();
	var eventID = null;
	if (event != null) {
		eventID = event.getID();
	}

	var transitionMessage = transition.getMessage();

	var sourceState = transition.getSource();
	var sourceStateID = null;
	if (sourceState) {
		sourceStateID = sourceState.getID();
	}

	var targetState = transition.getTarget();
	var targetStateID = null;
	if (targetState) {
		targetStateID = targetState.getID();
	}

	var logTime = new Date().getTime();
	var rejectMessage = "Condition failed because " + attribute.getID() + " does not
have a value";

	// Build JSON object for message
	var auditObject = {
		"nodeID": "" + nodeID,
		"workflowID": "" + workflowID,
		"transition": {
			"eventID": "" + eventID,
			"submitMessage": "" + transitionMessage,
			"sourceStateID": "" + sourceStateID,
			"targetStateID": "" + targetStateID
		},
		"logTime": logTime,
		"rejectMessage" : rejectMessage
	};
```

```
        var auditMessage = JSON.stringify(auditObject);

        // Send the audit message to the audit message framework
        topic.sendMessageAsync(auditMessage);
        return false;
}
```

Sample JavaScript for a Workflow State On Entry Audit Message

The following script sends an audit message when a specified object enters a specified workflow state:

Edit Operation

Execute JavaScript

Binds:

Binds

Variable name	Binds to	Parameter
topic	Audit Message Topic	WORKFLOWAUDITMESSAGES
node	Current Object	
manager	STEP Manager	
workflow	Current Workflow	
transition	Current Transition	
parameters	Workflow Parameters	
attribute	Attribute	

Messages:

Messages

Variable name	Message	Translatio

JavaScript:

```
31      taskAssigneeID = task.getAssignee().getID();
32      taskEntryTime = task.getEntryTime();
33      taskDeadline = task.getDeadline();
34  }
35
36  // Build JSON object for message
37  var auditObject = {
38      "nodeID": "" + nodeID,
39      "workflowID": "" + workflowID,
40      "userID": "" + userID,
41      "transition": {
42          "eventID": "" + eventID,
43          "submitMessage": "" + transitionMessage,
44          "sourceStateID": "" + sourceStateID,
45          "targetStateID": "" + targetStateID
46      },
47      "task": {
48          "assigneeID": "" + taskAssigneeID,
49          "entryTime": "" + taskEntryTime,
50          "deadline": "" + taskDeadline
51      }
52  };
53
54  var auditMessage = JSON.stringify(auditObject);
55
56  // Send the audit message to the audit message framework
57  topic.sendMessageAsync(auditMessage);
```

Edit externally

Script

```
var nodeID = node.getID();
var userID = manager.getCurrentUser().getID();
var workflowID = workflow.getID();

var event = transition.getEvent();
var eventID = null;
if (event != null) {
	eventID = event.getID();
}

var transitionMessage = transition.getMessage();

var sourceState = transition.getSource();
var sourceStateID = null;
if (sourceState) {
	sourceStateID = sourceState.getID();
}

var targetState = transition.getTarget();
var targetStateID = null;
if (targetState) {
	targetStateID = targetState.getID();
}

var taskAssigneeID = null;
var taskEntryTime = null;
var taskDeadline = null;

var task = node.getTaskByID(workflow.getID(), targetState.getID());
if (task) {
	taskAssigneeID = task.getAssignee().getID();
	taskEntryTime = task.getEntryTime();
	taskDeadline = task.getDeadline();
}
// Build JSON object for message
var auditObject = {
	"nodeID": "" + nodeID,
	"workflowID": "" + workflowID,
	"userID": "" + userID,
	"transition": {
		"eventID": "" + eventID,
		"submitMessage": "" + transitionMessage,
		"sourceStateID": "" + sourceStateID,
		"targetStateID": "" + targetStateID
	},
```

```
        "task": {
                "assigneeID": "" + taskAssigneeID,
                "entryTime": "" + taskEntryTime,
                "deadline": "" + taskDeadline
        }
};

var auditMessage = JSON.stringify(auditObject);

// Send the audit message to the audit message framework
topic.sendMessageAsync(auditMessage);
```

Analytics using JDBC Example

One of the prime uses cases for deploying JDBC as a method of delivering data from STEP to a database is to populate dashboards for data analytics tools like Tableau and Qlik. To aid proper setup of a STEP integration with data analytics that makes use of the JDBC method, a use case specific to data analytics integrations is described below. For more information on setting up a Tableau or Qlik analytics integration in the Web UI, refer to the **Visual Integration with Tableau or Qlik** topic in this guide.

One common approach to configuring data analytics dashboards is to enable display of historical data. As an example, a user has configured a Tableau dashboard to display regional sales data for a specific product. For this user, displaying sales figures for that same product a year ago, or two years ago, would greatly enhance the utility of the dashboard. The way to enable display of historical data using the JDBC method of delivering data is done in the mapping step of the Export Manager and outbound integration endpoint (OIEP).

To implement this users will need to:

- **Install required drivers** for JDBC
- **Configure properties** in the sharedconfig.properties file
- **Select format** of CSV, using the required settings

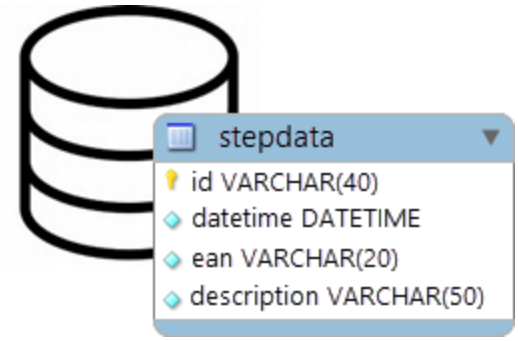
These steps are described in detail in the **Exporting Data via JDBC with CSV Format** topic in the **Data Exchange** documentation.

The following steps, which are described in detail in the text below, are specific to this use case:

- **Map Data** to include the required action field and calculated attribute for date / time
- **Select Delivery Method** of JDBC, using the necessary settings

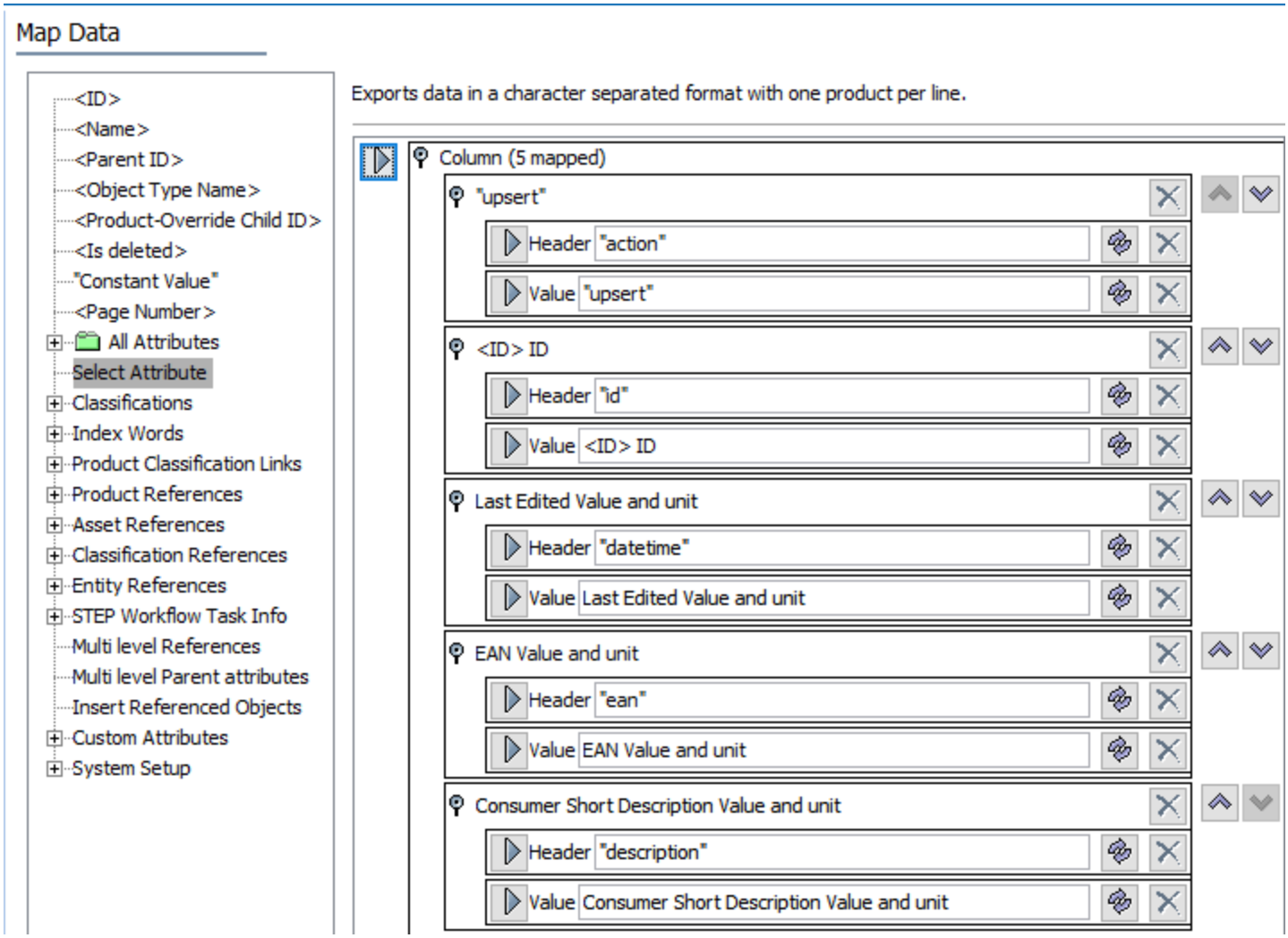
Map Data

With this example, data will be published to a table ('stepdata') that has the following layout:



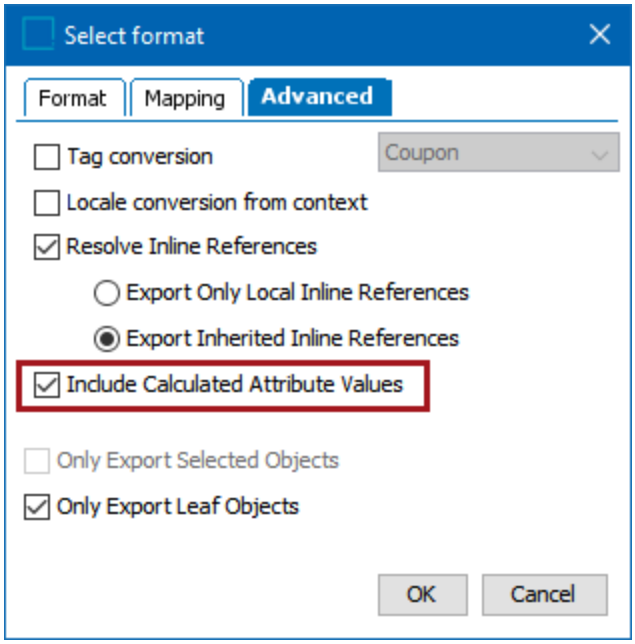
The table has the potential to contain multiple rows representing the same object, and the 'datetime' column will hold information about the latest STEP revision date. This can be achieved with a calculated attribute, mapped in the export configuration as described below. For more information on calculated attributes, refer to the **Calculated Attributes** topic in the **System Setup**.

Below is an example of an 'upsert' action in the Map Data step for the external database table shown above. This setup will direct the process to look for four configured attributes (ID, DATETIME, EAN, and Consumer Short Description) in the destination database table, and either insert the object (if not found), or update the object (if found):



- The column headers in the exported CSV file must match the table column headers in the destination database. Map a constant value on the Header row for each mapped object to supply the external database table column header. For example, notice that the external table has a row named 'description' but is mapped to the STEP attribute 'Consumer Short Description.' The header parameter is used to make the exported data match the external table. For details about using the constant value data source, refer to the **Constant Value - Data Source Outbound** topic in the **Data Exchange** documentation.
- Create an additional mapped column with the header of 'action' and the appropriate value of either delete or upsert. Use the transformation button to change the text displayed for both the Header and the Value.
- Create a calculated attribute using the function 'revisioneditdate()' and make it valid for your exported products. Then map the calculated attribute for export and update the header to match the column in the external table. Details on the 'revisioneditdate()' function are included in the **Other Functions** topic in the **Resource Materials** online help documentation.

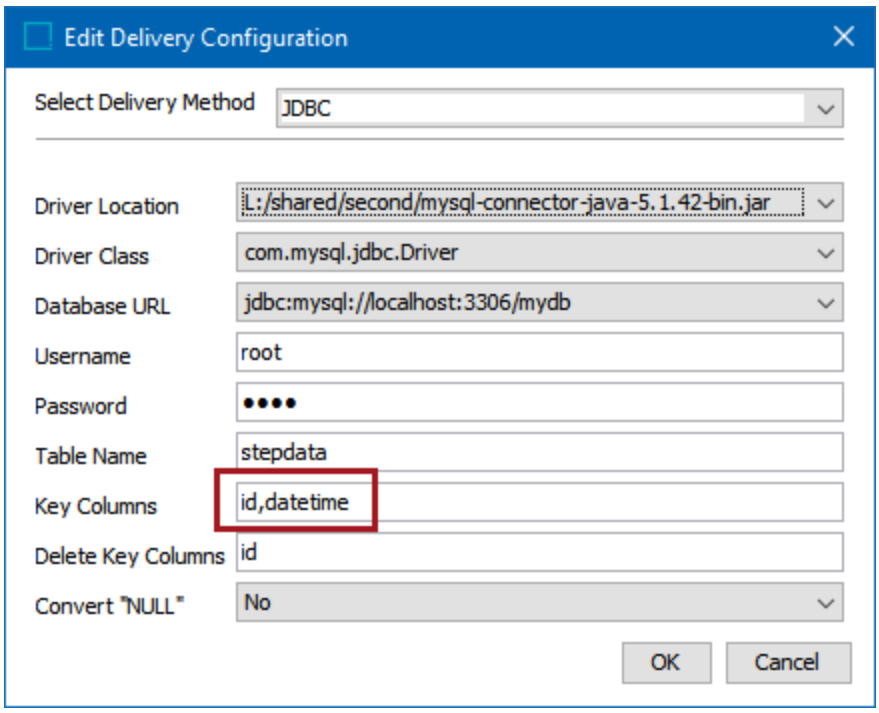
The example above uses the 'datetime' header, which is mapped to the 'Last Edited' calculated attribute. In addition to the other steps required to enact the JDBC method, the 'Include Calculated Attribute Values' parameter on the 'Advanced' step (shown below) must be checked.



If using an OIEP, one output template would be created to handle Create and Modify events, and a second output template would handle Delete events. In this way, separate mapping is available for each, allowing for one to include the upsert action and the other to include the delete action.

Select Delivery Method

The appropriate delivery configuration for this scenario can be found below. Notice that both the 'id' and the 'datetime' column values are used as keys for the upsert action, meaning both values must match to determine whether to insert or update the object. When an exact match for both keys is found, the object is updated; when no match is found, the object is inserted.



The value of this configuration is that if multiple instances of an object with the same ID but different datetime values (representing, for instance, different revisions of the object) are exported, the upsert action will insert into the table all instances of the single object. The effect of this setup is that various revisions of the same STEP data will be published to the table, giving users the ability to view historical data for objects in a data analytics dashboard. Further, if the Web UI has been configured to display data analytics dashboards, then Web UI users can view historical data on STEP objects.