



IBM Db2 Web Query for i Designer Organizing Content

Release 2.4.0

Contents

1. Organizing Content Into Pages	5
Creating Pages	6
Styling Pages	10
Working With Content Containers.....	10
Resizing Content on a Page	14
Editing Page, Section, and Container Properties.....	14
Hiding Content From Devices.....	18
Automatically Advancing Slides in a Carousel Container.....	21
Applying Themes and Styles to Pages.....	22
Applying Custom CSS and JavaScript Code to a Page.....	27
Styling Pages With Custom CSS.....	28
Enhancing Pages With Custom JavaScript.....	30
Db2 Web Query Designer JavaScript API Classes.....	36
Class: ibaObject.....	37
Class: ibaPage.....	38
Class: ibaSection.....	43
Class: ibaContainer.....	45
Class: ibaAccordionContainer.....	50
Class: ibaCarouselContainer.....	55
Class: ibaPanelContainer.....	59
Class: ibaTabContainer.....	64
Class: ibaContent.....	69
Adding Panels to a Responsive Container	70
Linking to External Content From a Page	70
Enabling Container Customization	73
Providing Access to Content Items and Db2 Web Query for i Tools	77
Creating Custom Templates	79
Passing Parameter Values to a Page	81
Bookmarking Control Selections in a Page	86
Exporting a Page as a PDF or Image	87
Legal and Third-Party Notices	89

Organizing Content Into Pages

Pages use containers to display multiple content items, and provide run-time options such as the ability to expand and collapse containers, navigate between items in a multi-content container, apply and clear filter selections, and even use custom functionality added using JavaScript. You can customize and style the entire page as well as sections, containers, and filter controls on the page using an intuitive set of options as well as CSS themes and custom CSS enhancements.

You can create a page in two ways. On the Db2 Web Query Hub or the Db2 Web Query Home Page, click the plus button and then click *Assemble Visualizations* to create pages using existing content from your environment. Alternatively, click the plus button and click *Create Visualizations* on the Hub, or click *Visualize Data* on the Home Page to create new content. When you create new content and click *Convert to page*, the visualization transforms from a single chart or report into a page.

You can run and share a page on its own, or add it to a portal to provide more context in combination with other pages.

In this chapter:

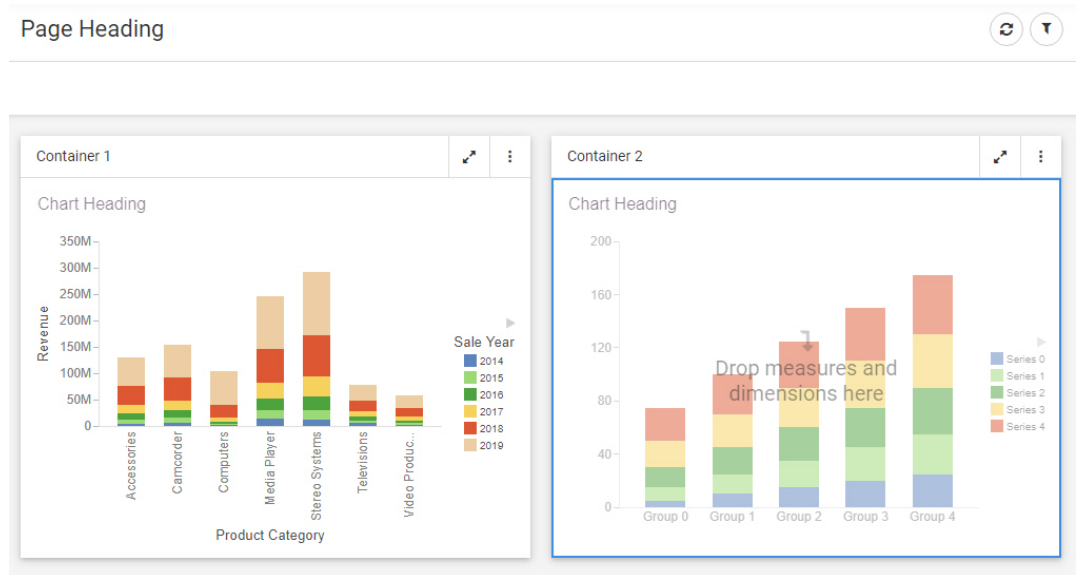
- ☐ [Creating Pages](#)
 - ☐ [Styling Pages](#)
 - ☐ [Adding Panels to a Responsive Container](#)
 - ☐ [Linking to External Content From a Page](#)
 - ☐ [Enabling Container Customization](#)
 - ☐ [Providing Access to Content Items and Db2 Web Query for i Tools](#)
 - ☐ [Creating Custom Templates](#)
 - ☐ [Passing Parameter Values to a Page](#)
 - ☐ [Bookmarking Control Selections in a Page](#)
 - ☐ [Exporting a Page as a PDF or Image](#)
-

Creating Pages

You can create a page in two ways. Create a visualization with new content using the *Create Visualizations* option on the plus menu of the Db2 Web Query Hub or the *Visualize Data* option on the Db2 Web Query Home Page, and then transform the chart or report into a page by clicking *Convert to page*. You can then create new charts and reports directly within the page. Alternatively, you can create a page from existing content by clicking *Assemble Visualizations* on the plus menu of the Db2 Web Query Hub or Db2 Web Query Home Page to open Db2 Web Query Designer in assemble mode. Once you select a page template, you can start adding content from your repository to the page. This content can include images, embedded URLs, and other existing content that is unavailable when you convert a single chart or report to a page.

Pages can include multiple sections, each with multiple containers for your content. Each of these containers can also be configured to contain multiple content items, providing a thoroughly comprehensive overview of your data. Pages also include run-time filtering, content navigation, and content customization, allowing users to analyze, explore, and enhance the information provided on each page.

To add a second item to a visualization, such as a chart or report, click *Convert to page* on the Visualization toolbar. This transforms the single content item into a page. The first content item is moved into a container. Click *Add container*, drag a container from the Container tab on the sidebar, or drag a field or existing content item into an empty area of the page to add a second container. A visualization to which a second content item has just been added is shown in the following image.



Select an item on the page from the canvas or the outline to edit it using a relevant set of tools and options, whether the item is a content item or a page component. The Settings and Format tabs on the Properties panel update accordingly. When assembling a page from existing content, you can also select multiple components, sections, or filters while holding down the Ctrl key in order to edit or style multiple items of the same type at once. Additionally, some options are applied hierarchically. For example, setting a theme for the page applies that theme to any sections, containers, and content items that use the default style option.

You can add more new content to the page by clicking *Add container*, or by clicking the *Container* tab on the sidebar and dragging a container onto an empty area of the page from the Resources panel. You can then select the content area of the new container and create a new chart or report within it. You can also drag a field onto an empty area of the page to create a new container and content item in the location where you dropped the field.

If you are creating a page from existing content, you start by selecting a page template. Select the Blank template to add content and containers to the page yourself, or select a preset layout with containers to which you can add your content. The InfoApp 1 template provides a collapsible filter panel and movable filter grid to which you can add filter controls, which are generated when you add parameterized content to the page. The Workbench template provides a resource tree that you can use to navigate to content items in your repository, which you can then run in the target container on the right. You can also create your own templates and save them in the Global Resources area of your repository.

Once you have selected a page template, you can add existing content from your repository to the page by clicking *Content* on the sidebar, and then navigating to and dragging an item onto the page or into a container. These items can be external charts, reports, HTML pages, URLs, images, and more. You can navigate between folders in the Resources panel and locate an item by name using the search bar.

You can also duplicate an item by right-clicking the container or subsection that contains it and clicking *Duplicate*. This makes it easy to develop content by maintaining the styling, fields, and other properties from the existing content item and container. The title of the duplicated item is reused, with the text *(Copy)* added to the end. If you duplicate the content in a single-item panel container, a new container with the same content is added to the page. When you right-click a multi-content container, you can click *Duplicate Container* to duplicate the entire container, including all content, or click *Duplicate Content* to duplicate only the visible tab, accordion panel, or carousel slide.

You can move and resize containers on the page to organize them, or select a preset layout from the content picker to automatically create and arrange containers into a preset pattern.

You can save space on the page by placing multiple content items together in a tab, carousel, or accordion container.

If you are creating a page from existing content, to change a standard panel container into a tab, accordion, or carousel container, select *Content* on the sidebar and then drag a content item onto the container, already populated with content, on the canvas. The entire container is highlighted, indicating that the item will be dropped into the container. When you release the item into the container, you are prompted to replace the existing content, add the item in a new tab, accordion panel, or carousel slide, or cancel the operation.

Alternatively, to instantly convert a container to another type, you can right-click it on the canvas and point to *Convert to*, then select a container type of your choice. If you convert a container with multiple items into a panel container, which only supports one content item, each content item from the original container is added to a separate panel container. For example, if you convert a tab container with three tabs into a panel container, three panel containers are created.

You can combine multiple, existing containers on a page into a single multi-content container. Hold the Ctrl key and click the toolbars of multiple containers to select them, then right-click a container, point to *Combine*, and select a container type to combine them into. You can combine containers into a tab, accordion, carousel, or panel group container. The Combine option is unavailable when combining existing link tile, grid, and panel group containers, and for the Explorer widget, since these are special containers that do not contain standard content.

You can also combine containers by holding the Ctrl key and dragging one container onto another. When you drop the container, the Combine containers dialog box opens, allowing you to select the type of container into which to combine the contents of the containers involved, or to cancel the operation. If you combine the selected containers, the new container occupies half of the page width and is placed within the available empty space on the page. Once the containers have been combined, you can hold the Ctrl key and drag other containers on the page into the grouped multi-content container or panel group.

To delete a container from a page, right-click it and click *Delete container*. If the container includes multiple items, you can click *Delete Tab*, *Delete slide*, or *Delete Area* to delete a single sub-component, depending on the container type. You can also delete the selected container or sub-component by pressing the Delete key.

When creating new content in a page, you can filter the entire page by dragging a field from the Resource panel into the Filter toolbar, just like when you create a content item. You can select default values for the filter from the resulting filter control, and right-click it to change other filter properties. If you are creating a page with existing content, then when parameterized items are added to a page, a badge appears on the Filters tab of the sidebar, alerting you that you can create filter controls. Open the Filters tab and click *Add all filters to page* to create filter controls for all parameters, or right-click an individual parameter in the list and click *Add to page* to create a filter control for a single parameter. These controls will apply filters to any items that are parameterized for the same parameter variable and field.

As you create your page, you can click *Run in new window* to see a run-time view of your page, allowing you to interact with filtering, built in container controls, In-Document Analytics, and any unlocked or interactive content on the page. You can check these variables by clicking *Info* on the Visualization toolbar.

Be sure to periodically save your page by clicking *Save* on the Db2 Web Query Designer toolbar.

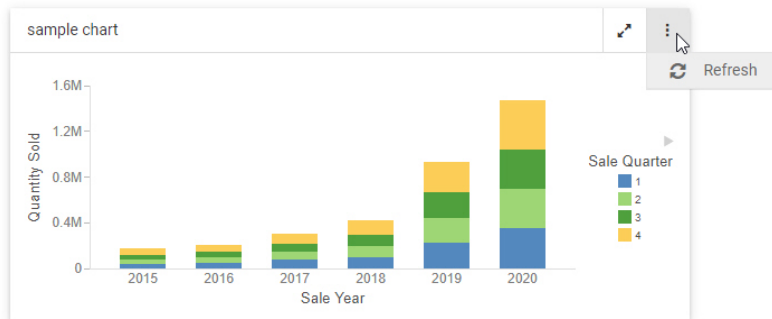
Styling Pages

A number of tools in Db2 Web Query Designer support your efforts to apply styling to pages and page components. You can resize content, hide content from smaller devices if it interferes with a responsive layout, edit page, section and container properties, apply filters, themes and styles, and create custom templates.

Working With Content Containers

When you add content to a page, it is automatically placed in a container. To move and resize the content item, move and resize the container into which it was placed by dragging the container or its resizing handles. You can also configure and style a container by selecting it on the page and then using the options on the Settings and Format tabs of the Properties panel. Note that the options for a container are separate from those for the content item inside the container. To select the container, click the container toolbar, or select a container from the Outline tab of the sidebar. Empty containers can be added to the page by selecting the *Container* tab on the sidebar and dragging the selected container type into the canvas. A highlighted area shows where the container will be placed.

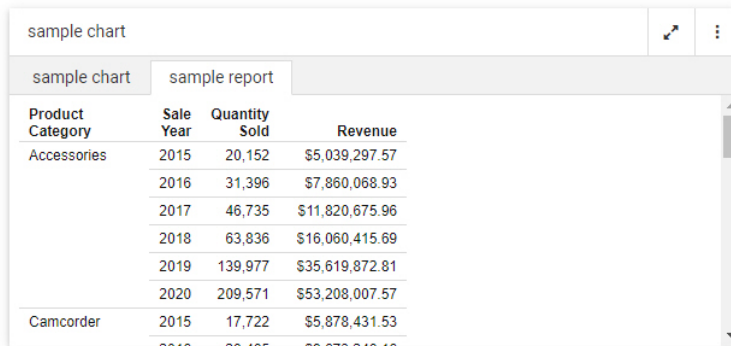
There are several types of containers. The default container for a single item is a panel container. The panel container is a basic container type that includes a title bar and toolbar with a run-time menu, as shown in the following image.



The run-time menu provides the ability to refresh the content in the container by default. If the content currently visible in the container uses the AHTML output format, additional In-Document Analytic run-time options are added to this menu. For more information about In-Document Analytics, see *Overview of In-Document Analytics*. You can also add custom options to the run-time menu and to the container toolbar using embedded JavaScript. For more information, see [Enhancing Pages With Custom JavaScript](#) on page 30.

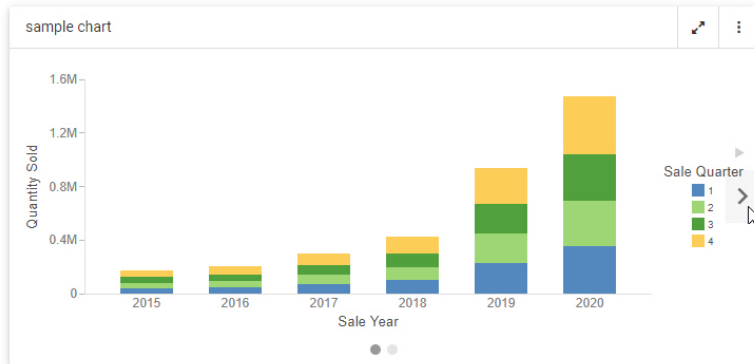
To add multiple items, such as charts, reports, and images, to a single container, use the tab, carousel, or accordion container types. These container types include the same features as the panel container, as well as controls to navigate between multiple content items. You can add another content item to a container by selecting the *Content* tab on the sidebar, navigating to an item, and dragging it onto the existing container. You are given options to add the new item to a new tab, panel, or slide, or replace the visible, existing item in the container.

In a tab container, navigate between items using the tabs below the title bar. You can double-click the text in a tab to change it, and right-click a tab to delete, duplicate, or reorder it. A tab container is shown in the following image.

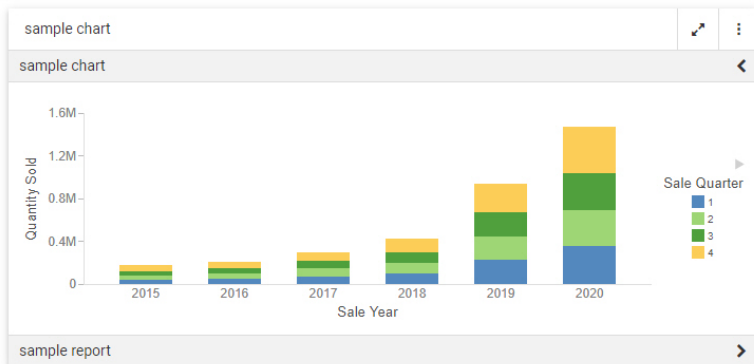


sample chart		↗	⋮
sample chart	sample report		
Product Category	Sale Year	Quantity Sold	Revenue
Accessories	2015	20,152	\$5,039,297.57
	2016	31,396	\$7,860,068.93
	2017	46,735	\$11,820,675.96
	2018	63,836	\$16,060,415.69
	2019	139,977	\$35,619,872.81
	2020	209,571	\$53,208,007.57
Camcorder	2015	17,722	\$5,878,431.53

A carousel container allows you to switch between content items by clicking a dot at the bottom of the container, or by clicking arrows that appear when you point to the left or right side of the container. When creating a page, right-click a carousel container to duplicate, reorder, or delete a slide. You can set a carousel container to automatically cycle between content items using the options described in [Automatically Advancing Slides in a Carousel Container](#) on page 21. A carousel container, with navigation arrows, is shown in the following image.



Accordion containers place content in collapsible panels. You can expand and collapse items in an accordion container by clicking a panel label. When creating a page, right-click an individual panel in an accordion container to rename, reorder, duplicate, or delete the panel. An accordion container is shown in the following image.



Other types of containers serve more specific purposes. The grid container type, only available in pages assembled from existing content, can be used as a filter container, as an alternative or in addition to the filter toolbar. Once you add filter controls from your content to a page, you can drag them from the default location in the filter toolbar into the grid control. Select the filter bar within the panel from the outline, and then select the *Format* tab to change the number of columns in the grid container. You can also drag a text label or submit button to the grid from the Control tab on the sidebar.

You can use a panel group container to keep a set of vertically arranged panels together when responsive folding occurs as the size of the page changes. To use a panel grid container, add it to the page, then drag new containers, content items, or fields into it. For more information, see [Adding Panels to a Responsive Container](#) on page 70.

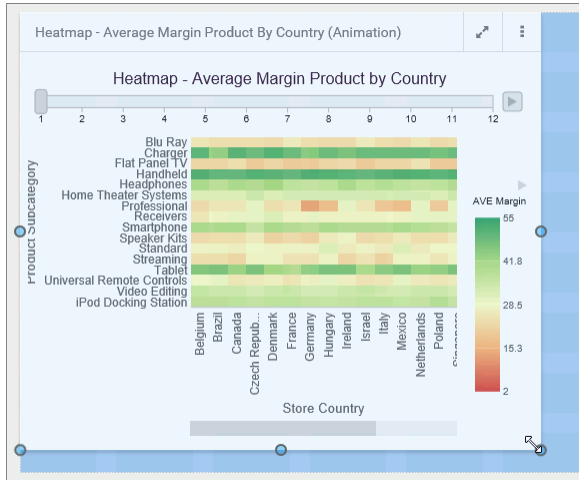
A link tile container allows you to display a content item or image on a page that can be clicked to access additional, outside content. Drag the link tile container onto the page, then, from the *Settings* tab, use the link tile Background, Content, Attach all parameters, and Target properties, to set the displayed and target content, and configure how the link tile behaves. For more information, see [Linking to External Content From a Page](#) on page 70.

You can use the Explorer widget to provide access to the Workspaces view of the Db2 Web Query Hub or Db2 Web Query Home Page. This allows anyone accessing the page, or a portal containing the page, to open and run content items and create new content, depending on their privileges. In the Explorer widget, you can navigate through workspaces, folders, and content items from the Resources tree and Content area, and create new content using options on the Action bar. For more information, see [Providing Access to Content Items and Db2 Web Query for i Tools](#) on page 77.

Right-click any type of container to access a shortcut menu of options. These include the ability to refresh the container content, change the title of the container, access the Settings or Format tabs, duplicate the visible content item, change the container type, or delete the container. If you select multiple containers, you also have the option to combine them into a tab, accordion, or carousel container.

Resizing Content on a Page

You can resize content on a page, at any time, using the sizing handles and the shaded placement area that appear on the canvas. When you hover over a container on the canvas, a series of handles appear. Drag a handle in the direction that you want to resize the item, as shown in the following image.



The content container snaps to the nearest row and column when you resize it. To change the number of columns in a section on the page, allowing more granular container widths, select a section, open the *Format* tab, and select an option from the Number of Columns area. 12 is the default. You can also change the amount of space between containers by selecting the entire page and typing a size for the Margin property, on the Format tab.

Editing Page, Section, and Container Properties

The Properties panel displays the properties for the element that is selected. To access container properties, select a container from the canvas or outline. To access section properties, click the grid inside the canvas, or select a section from the outline. To access page properties, click the page header or the toolbar, or select a page from the outline. If the page header and the toolbar are hidden, you can also access page properties by clicking the canvas outside a grid section.

The following properties are available when you select an entire page.

Settings tab:

- ☐ **ID.** Contains a read-only unique CSS identifier.

- ☐ **Classes.** Allows you to add one or more custom CSS classes that you can reference in custom JavaScript and CSS code.
- ☐ **Show page heading.** Toggles between hiding and showing the title in the header.
- ☐ **Show page toolbar.** Toggles between hiding and showing the page toolbar.
 - ☐ **Show page refresh.** Toggles between showing and hiding the refresh button on the page toolbar.
 - ☐ **Show export.** Toggles between showing and hiding the Export menu on the page toolbar. You can export a page as a PDF or as a .png image. Select just the *PDF* or just the *Image* check box to make just one of these export options available from the page, in which case the Export menu turns into an Export button.
- ☐ **Include Page Filters.** Select this check box to add a filter toolbar to the page, even if no filters have been added yet. Clear this check box to remove the filter toolbar.
 If *Include Page Filters* is selected, you can select one of the following options to change the position of the filter toolbar:
 - ☐ **Below Header.** The filter toolbar is placed below the page toolbar and header. This is the default.
 - ☐ **Above Header.** The filter toolbar is placed above the page toolbar and header.
 - ☐ **Left Position.** The filter toolbar is aligned vertically along the left side of the page, below the page header.
 - ☐ **Modal.** The filter toolbar is accessible in a modal window overlaid onto the page. Click the *Show filters* button on the page toolbar to open the modal window and make filter selections.

Format tab:

- ☐ **Theme.** Allows you to select a theme for the page. The default options are Designer 2018, Light, Midnight, and Vivid. Administrators can configure additional themes that will be available to users from this property. For more information, see [Applying Themes and Styles to Pages](#) on page 22.
- ☐ **Margin.** Controls the size of the margin between the border of the page and the content.

Note: If the margin property is set to 0, you cannot select a section on the page from the canvas or access the section shortcut menu options. As a workaround, change the margin property value to 20px temporarily to gain access to the section and its options.

- ☐ **Maximum width.** Controls the maximum width of the page.
- ☐ **Page Heading Style.** Provides a selection of typeface styles for the page heading.
Additional header text formatting can be done using CSS. For example, you can assign a class to the page and use the font-family property to change the header text font.

The following properties are available for sections.

Settings tab:

- ☐ **ID.** Contains a read-only unique CSS identifier.
- ☐ **Classes.** Allows you to add one or more custom CSS classes that you can reference in custom JavaScript and CSS code.
- ☐ **Collapsible.** Toggles between collapsible and noncollapsible modes of the selected section.

Format tab:

- ☐ **Style.** Allows you to choose a style for the selected section, or select *Default* to use the theme selected for the page.
- ☐ **Row height.** Sets the height of the section that is currently selected on the canvas. The default value is 60px.

Note: Changing the row height does not change the margin, which can be configured on the page level.

- ☐ **Number of Columns.** Allows you to change the number of columns in a section, enabling more granular panel widths. Panels snap to the nearest column and row when you resize them. 12 columns are used by default, and 28, 45, and 80 column options are also available.

The following properties are available for containers.

Settings tab:

- ☐ **Container Settings.** Provides access to the following properties.
 - ☐ **ID.** Contains a read-only unique CSS identifier.
 - ☐ **Classes.** Allows you to add one or more custom CSS classes that you can reference in custom JavaScript and CSS code.
 - ☐ **Show title.** Toggles between hiding and showing the title.
 - ☐ **Show toolbar.** Toggles between hiding and showing the container toolbar.

- ☐ **Show On.** Allows you to hide the selected container or item from displaying on specified devices. The options include desktop, tablet, and mobile.
- ☐ **Autoplay interval.** Available for carousel containers. Allows carousel slides to automatically cycle based on the selected interval, in seconds. When set to 0, autoplay is disabled.
- ☐ **Rerun content.** Available for carousel containers. When the Autoplay interval is set to a value higher than 0, the Rerun content option allows you to refresh the content on each slide of a carousel container when it appears. If Rerun content is not enabled, then the content on each slide only loads the first time it appears.
- ☐ **Container Customization.** Provides options that allow users to change the content in a page at run time. Includes the following properties.
 - ☐ **Lock container.** If disabled, allows users to change content in the panel at run time. This property is enabled by default.
 - ☐ **Select content from.** Allows you to set the initial directory that users will see when they click the *Add content* button.

Note: To see the full value displayed in the field, widen the Properties panel area by clicking the separator and dragging it to the right.
 - ☐ **Lock path.** When enabled, limits the content selection area to the directory that is specified in the Path property. This property is enabled by default.
 - ☐ **Flatten list.** If enabled, hides the folder hierarchy and the breadcrumb trail within the directory that is specified in the Path property. This property is disabled by default.

Note: This property is especially useful if you want to display different widgets to different users based on their roles. For example, you can create a series of subfolders with content and then apply security rules to show and hide these subfolders from users based on their group membership. Then you can configure the *Path* property to display the parent of these subfolders and flatten the list. Now each user can only see the widgets they are authorized to see in a simple list without any folders to navigate.
 - ☐ **Hide tags.** If enabled, hides tag buttons in a flattened list. This property is disabled by default. It remains inactive when the *Flatten list* property is disabled.
 - ☐ **Initial view.** Determines the initial view of the directory that users will see when they click the *Add Content* button. The options are Grid and List.

Note: Users can change the view at run time inside the Select Item dialog box.

Format tab:

- ☐ **Style.** Allows you to choose a style for the selected container or item.

The following options are available for content items in a page assembled from existing content. New content items in a page provide standard content creation options, in addition to these.

Settings tab:

- ☐ **Content Properties.** Provides access to the following properties.
 - ☐ **ID.** Contains a read-only unique CSS identifier of the content item. You can reference this ID in custom JavaScript and CSS code as well as drill-down targeting procedures.
 - ☐ **Classes.** Allows you to add one or more custom CSS classes that you can reference in custom JavaScript and CSS code as well as drill-down targeting procedures.

Format tab:

- ☐ **Content Format.** Provides access to the following property.
 - ☐ **Theme.** Select a theme to apply to the content item. You can explicitly select a theme, or have the item inherit the theme used in the page. When the item is part of a page assembled from existing content, you can also select *Use content theme* to use the theme originally applied to the item, ensuring that it does not get overwritten by the page or portal theme. For more information, see [Applying Themes and Styles to Pages](#) on page 22.

Hiding Content From Devices

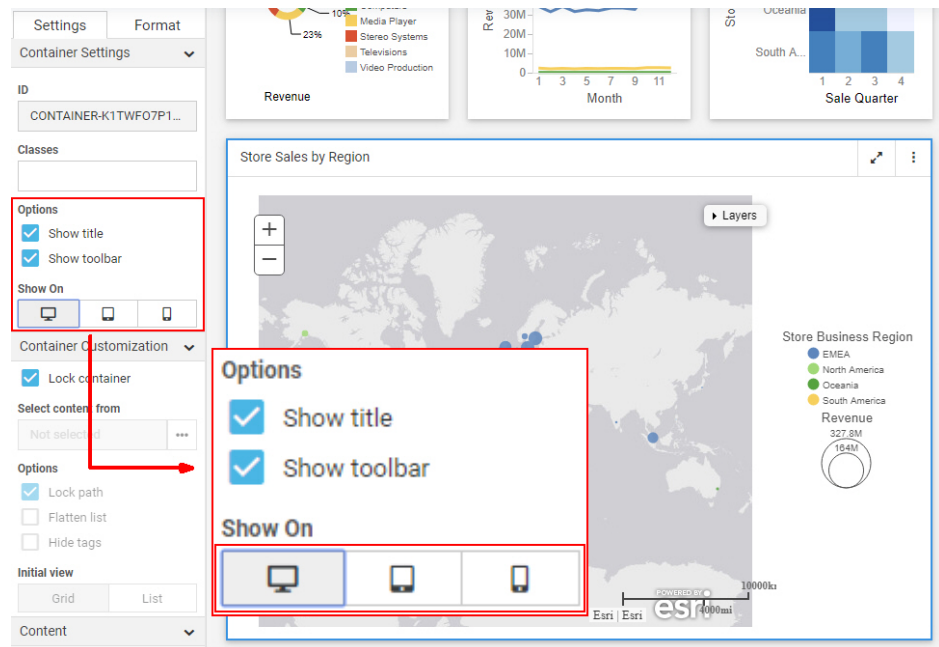
You can choose to hide certain content items from smaller devices, especially when dealing with large or multi-faceted items like maps or HTML pages. When an item is hidden from a device, the other items around it re-flow into the empty space and retain the responsive layout.

Procedure: How to Hide a Content Item From Devices

1. In Db2 Web Query Designer, select a large item that you want to hide from small-screen devices.
2. On the Properties panel, on the Settings tab, under the Show On property, click a device icon to hide the selected item from this type of device.

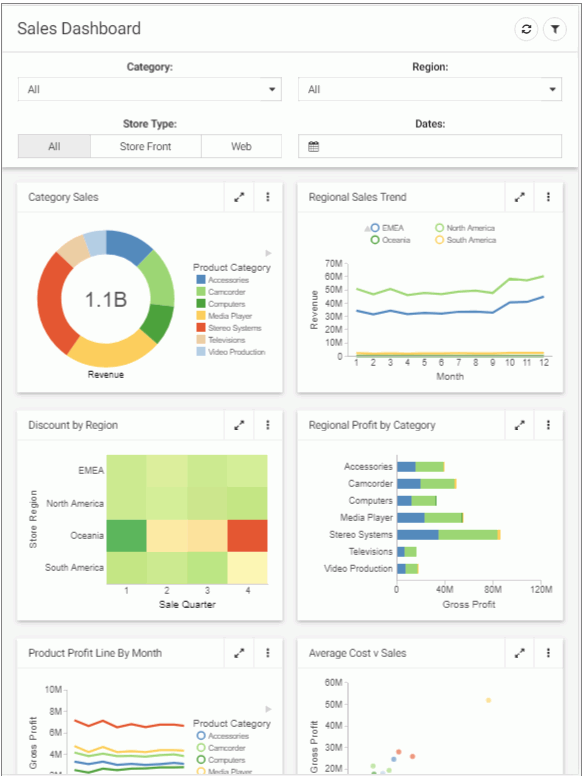
Note: All three device buttons are selected, by default, making the item visible on all devices. Manually clear the buttons to hide the item from specific device types.

An example of hiding a large item from mobile phones and tablets is shown in the following image.



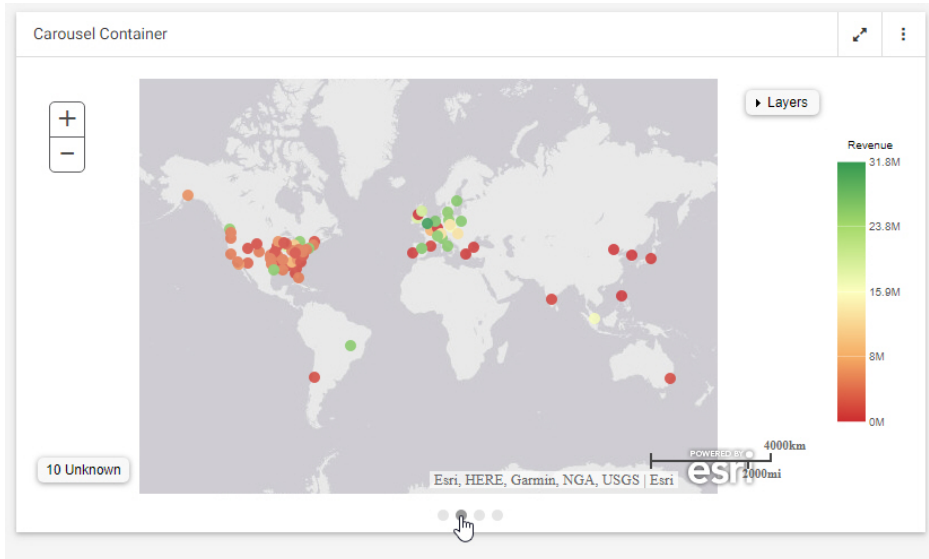
3. Save the page.
4. Run the page on a device, from which the item was hidden.

The item does not display. All remaining content items rearrange to cover the empty space, as shown in the following image.



Automatically Advancing Slides in a Carousel Container

Using a carousel container is one way to put multiple charts, reports, and other content items into a single container. Carousel containers include an unobtrusive control that allows users to cycle through the items added to it, as shown in the following image.



Unlike tabbed or accordion containers, carousel containers also include the option to autoplay the content within them, allowing you to show all of the content in the carousel container without requiring user interaction.

To enable autoplay for a carousel container, select the container and open the Settings tab on the Properties panel. The Autoplay interval and Rerun content properties are available in the Container Settings section. To enable autoplay, set the Autoplay interval to a value higher than zero (0) to enable autoplay. The Autoplay interval is the amount of time, in seconds, that each slide is shown before advancing to the next slide.

When the Autoplay interval is set, the Rerun content option becomes available. This option is turned off, by default. When enabled, the content on a slide of the carousel container refreshes each time it is displayed. When Rerun content is not enabled, the content on each slide only loads the first time it is displayed, and it is not refreshed unless manually prompted from the menu on the container toolbar.

Applying Themes and Styles to Pages

While customizing your page, you can apply themes and styles to various page elements. The general theme of the page is defined by the Theme setting, which you can configure in the Properties panel, on the page level. A theme affects the look of the entire set of elements on the page, including colors, opacity, and typeface styles. Themes also dictate the color schemes available as styles on the Format tab when editing sections, filters, or containers.

By default, the theme selected for a page is passed on to content in the page. You can select content items inside a container and change the theme for each one individually from the Format tab. If the content item was added to the page from the Content tab, you can additionally choose to use the original theme of the content item, instead of inheriting the page theme or selecting a specific theme to use.

There are four themes that Db2 Web Query Designer offers:

- ☐ Designer 2018
- ☐ Light
- ☐ Midnight
- ☐ Vivid

To add a theme to this list, create a folder in the Global Resources area, in the Custom folder within the Themes folder. In that folder, you can add a custom StyleSheet to use with charts and reports, and a custom CSS file to use for pages. The .sty file and .css file must both be named *themes*.

Note: To facilitate the creation of a custom theme, you can copy and modify properties from one of the themes included with Db2 Web Query. This makes it easy to identify the default classes used in a Db2 Web Query Designer page.

Once you select a theme, you can further modify it by configuring styles. You can also save your unique combination of a theme and styles as a custom theme.

Procedure: How to Apply a Theme to a Page

1. In Db2 Web Query Designer, click the page toolbar to select the entire page, and then click the *Format* tab on the Properties panel.

The style properties appear.

2. In the Page Style section, from the Theme property drop-down list, select the theme that you want to use.

The page refreshes with the new theme.

3. Save your changes.

Procedure: How to Apply Styles to Sections

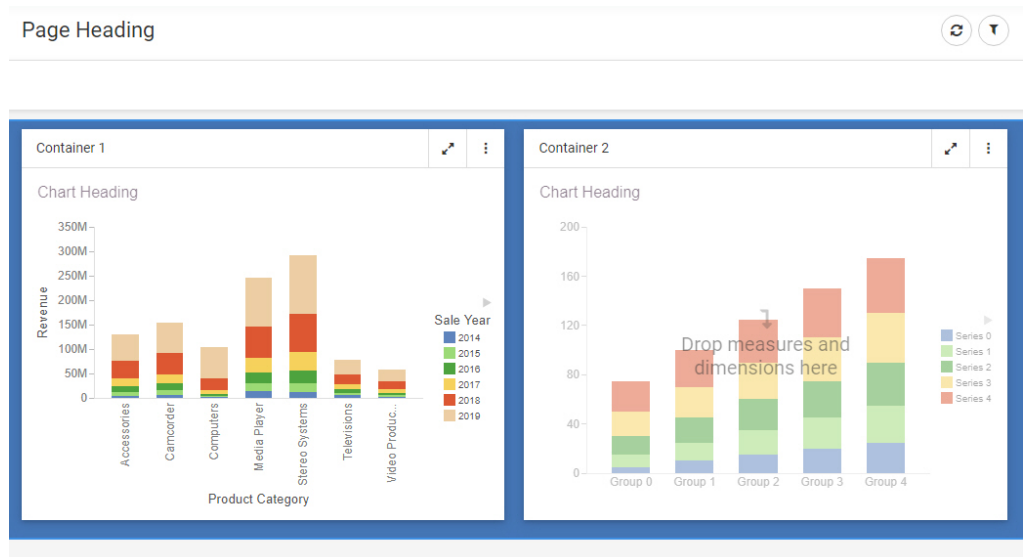
1. In Db2 Web Query Designer, select a page section by clicking the background area near a container, or by selecting a section from the outline.
2. On the Properties panel, click the *Format* tab.

The Style properties appear.

3. Click a style that you want to use.

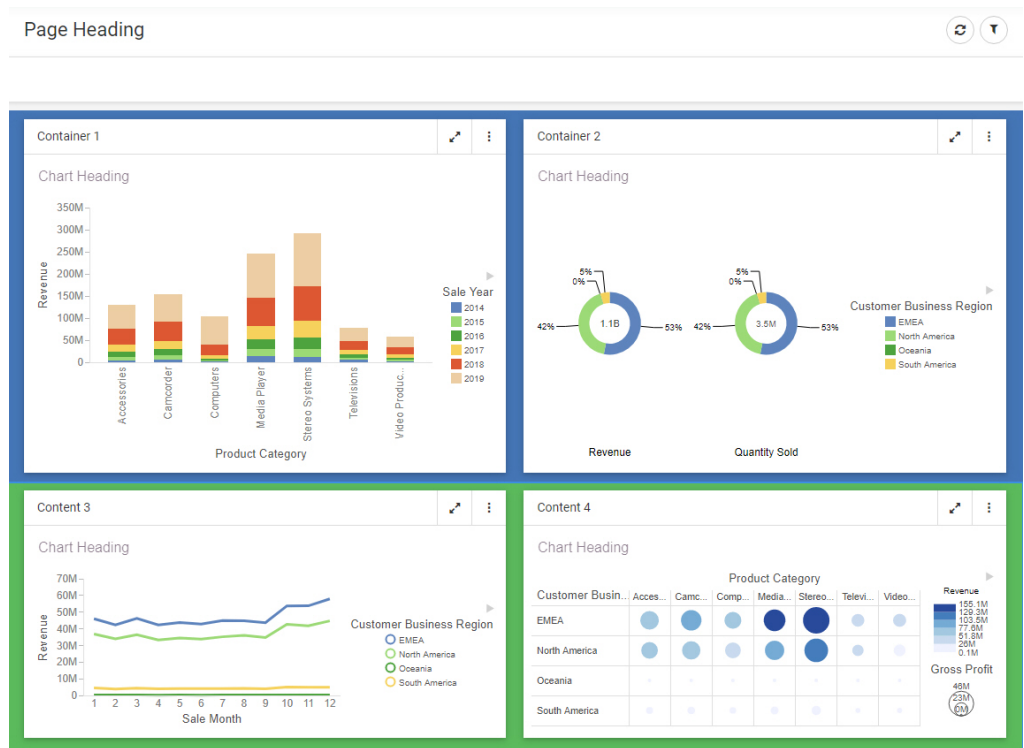
Note: Styles are based on classes in the selected theme. For example, use the class `.pd-style-default-color` to configure the Default style of the theme, use the class `.pd-style-color2` to configure Style 2 of the theme, and use the class `.pd-style-color3` to configure Style 3 of the theme.

The page refreshes and applies the style to the section, as shown in the following image, where the selection of Style 2 has made the first section blue.



4. Optionally, add more sections and apply styles to them.
5. Click the new section and apply a style to it, as described in step 3.

The page refreshes and applies the new style. You can apply different styles to different sections, as shown in the following image.



6. Save your changes.

Procedure: How to Apply Styles to Containers

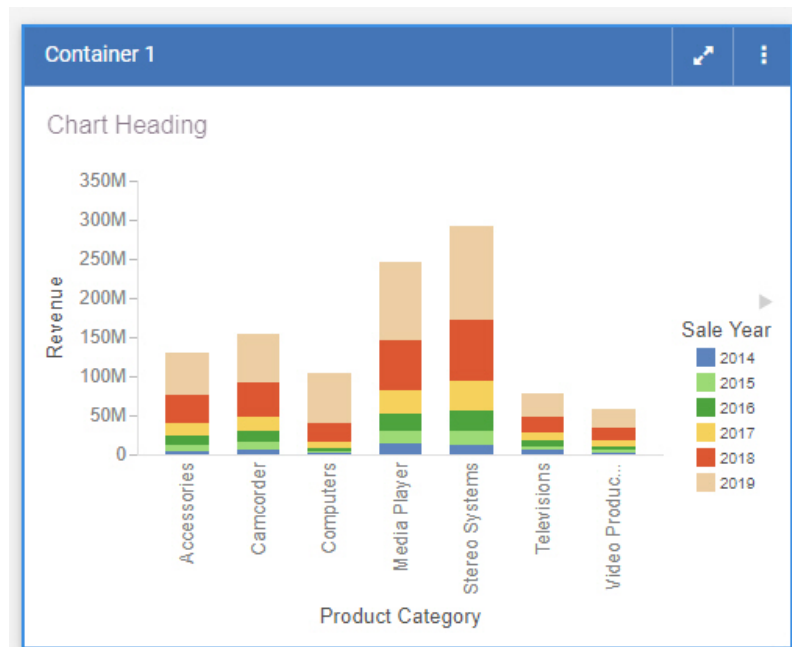
1. In Db2 Web Query Designer, create a visualization with multiple containers.
2. Click a container, and, on the Properties panel, click the *Format* tab.

The Style properties appear.

3. Click a style that you want to use.

Note: Styles are based on classes in the selected theme. For example, use the class `.pd-style-default-color` to configure the Default style of the theme, use the class `.pd-style-color2` to configure Style 2 of the theme, and use the class `.pd-style-color3` to configure Style 3 of the theme.

The page refreshes and applies the style to the container, as shown in the following image.



- Optionally apply styles to other containers on the canvas.

Note: When creating a visualization from existing content, to apply the same style to multiple containers, multi-select containers by holding the Ctrl key, and then click a style button.

- Save your changes.

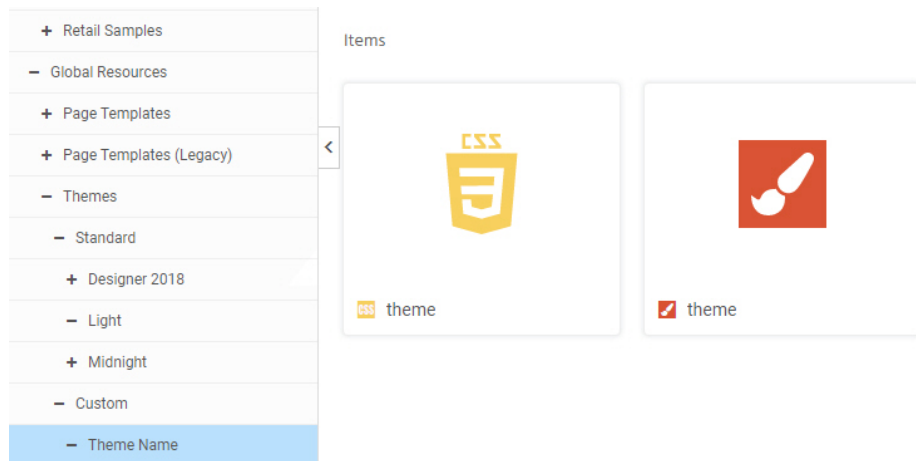
Procedure: How to Create a Custom Theme

- Sign in to Db2 Web Query as an administrator.
- On the Hub or Home Page, click *Workspaces* and, from the Resources tree, expand the *Global Resources* folder, and then expand the *Themes* folder.
- Open the *Custom* folder, and then click *Folder* on the Action bar.
The New Folder dialog box opens.
- Populate the Title field with the name of your custom theme, and click *OK*.

The custom theme folder is created. Your theme CSS file will reside in this folder. If you know which CSS classes should be used for your theme, you can create a new CSS text file, add your code, and save it with the name *theme*. The file must be called *theme* to be available for selection in Db2 Web Query Designer. To facilitate the process, you can modify an existing theme CSS file. In this example, we copy and modify the theme CSS file for the Light theme.

5. Expand the *Standard* folder, and then expand the *Light* folder.
6. Copy the theme CSS file, and paste it inside your new custom theme folder.

The following image shows the correct hierarchy of the custom theme file.



Note: Do not modify the name of the theme CSS file. It is imperative that both the CSS file and StyleSheet in your custom theme folder are called *theme*. The name of the folder in which the file resides is the theme name that is visible in Db2 Web Query Designer.

7. Right-click the newly copied theme CSS file, and then click *Edit*.

The Text Editor window opens.

8. Modify the code to achieve the desired look of the theme.

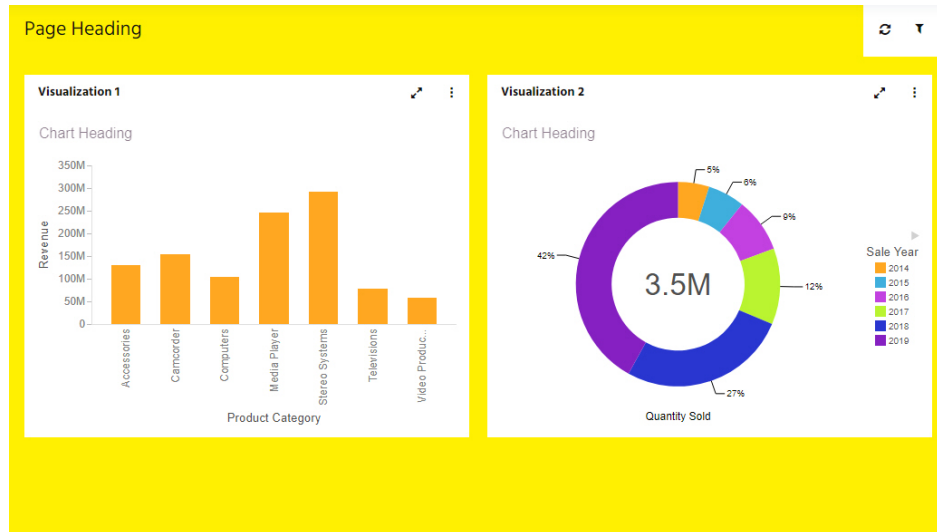
In this example, you could change the background color from white #fff to yellow #ffff00.

9. Save and close the Text Editor.
10. Optionally, configure a related content theme by creating a StyleSheet (.sty) file called *theme*. You can copy this from an existing theme folder, such as the Light folder, as well.

The styling properties in the StyleSheet will be applied automatically to content on the page when your custom theme is used, unless that content has a different theme explicitly assigned.

11. In Db2 Web Query Designer, apply the new custom theme to a page, as described in [How to Apply a Theme to a Page](#) on page 22.

An example of a new theme applied to a page is shown in the following image.



Notice that the content theme, or StyleSheet, associated with the page theme was automatically applied to the content on the page.

12. Save your changes.

Applying Custom CSS and JavaScript Code to a Page

You can apply custom cascading style sheet (CSS) properties and JavaScript code to a page, allowing you to significantly enhance the page with countless styling options and run-time behaviors using code that you write yourself. CSS makes it easy to style different elements on a page by applying various styling properties to them, while JavaScript is a robust programming language that you can use to customize a page with dynamic and interactive features not normally available in Db2 Web Query Designer.

This feature is intended for content developers with CSS and JavaScript coding experience. The CSS and JavaScript options are available to developers, administrators, and users with text editor access.

You can add custom CSS and JavaScript to a page by typing code into the CSS and Javascript text editors, respectively. To access the CSS or JavaScript text editor, click *Outline* on the sidebar, and then select *CSS* or *JavaScript*. The text editor opens over the canvas, and you can type your code into it, as shown in the following image.



The code that you write can reference classes that you assign to different components on the page by selecting them and assigning a class name, using the *Classes* property on the *Format* tab.

To return to the canvas, click the close icon in the corner of the page.

When you use the text editor, the Db2 Web Query Designer toolbar provides additional options to copy, cut, paste, and find and replace your text. The text editor also includes built-in features such as syntax highlighting, code folding, line numbering, autocomplete, alerts, indent guides, and a status bar. These features facilitate CSS and JavaScript code development and debugging by improving clarity and navigability in the text editor.

Note: Custom CSS and JavaScript is not displayed in the canvas. You must run the visualization to see the results of your custom code.

Styling Pages With Custom CSS

Cascading style sheets, or CSS, is a commonly used language that allows you to style different components on a page. CSS properties include font style, text color, background color, and much more. Custom CSS can be applied to specific objects on a page by specifying a CSS class for the object, and then using a class selector in your CSS code.

To specify a CSS class for an object on a page, select the object and, on the *Properties* panel, type a class name into the text box for the *Classes* property. You can specify multiple classes for an object by separating the class names with spaces, and you are encouraged to use the same class names for multiple items on the page. When you assign attributes to a class in the *CSS* tab, they will affect all elements on the page that are assigned to that class.

Procedure: How to Change the Color of a Panel Using CSS

You can change the background color of a panel by assigning a class to the panel and then adding the background-color attribute to the class using CSS.

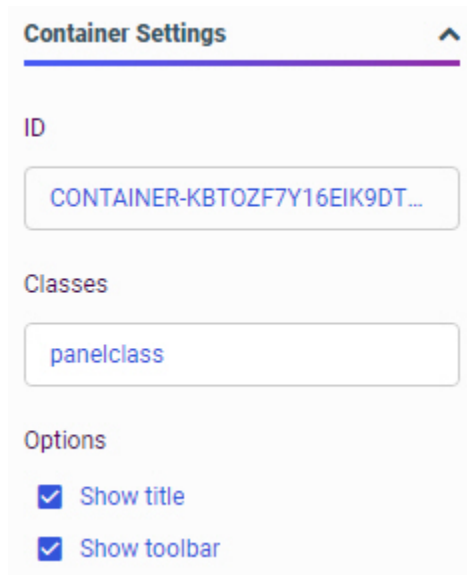
1. Open Db2 Web Query Designer. On the Db2 Web Query Hub, click the plus menu and click *Create Visualizations*, or, on the Db2 Web Query Home Page, click *Visualize Data*.

Db2 Web Query Designer opens in a new browser tab.

2. Select a workspace and a data source available from that workspace.

Once you select a data source, Db2 Web Query Designer loads with options to create a single content item.

3. Click the *Convert to page* button to transform the visualization from a single content item to a page.
4. Optionally, create new content in the visualization and add new containers to the page for additional content.
5. Select a container on the page.
6. On the Settings tab of the Properties panel, in the text box for the *Classes* property, type a class name of your choice, as shown in the following image.



7. Click *Outline* on the sidebar to display the visualization outline.
8. Click *CSS*.

The CSS text editor opens.

9. Add a CSS declaration to change the background-color attribute for the class used by the panel.

- a. Reference the class name that you specified in the Classes property.

Type a period (.) followed immediately by the class name, then an opening curly bracket ({}), and press the Enter key.

The closing bracket is added automatically.

- b. Within the brackets, type *background-color*, followed by a colon and then a color string.

For example, to make the color of a panel, for which *panelclass* is the value of the Classes property, to a bright blue, you could use the following CSS declaration.

```
.panelclass {  
    background-color: rgb(0,180,240);  
}
```

10. Save the visualization and close Db2 Web Query Designer.

11. Run the visualization from the Hub or Home Page.

The custom CSS is applied to the specified container.

Enhancing Pages With Custom JavaScript

You can use custom JavaScript to enhance the functionality of components on a page. Any component created on the page using the Db2 Web Query Designer canvas can be modified in the JavaScript text editor by referencing a CSS class name specified in the Classes field of the Properties panel for the object, or using the Db2 Web Query Designer JavaScript API, to standardized classes that correspond to different component types in a page or portal. Each component class can utilize a set of predefined methods in addition to custom JavaScript. You can even use JavaScript to add objects to a page entirely through code, without using the canvas at all.

Procedure: How to Add a Button and a Menu Item to a Panel Container Using Custom JavaScript

You can use custom JavaScript code to add a button and a menu item to a panel that you have added to a page. You can configure the button and menu item to execute a command of your choice, such as a URL call.

For information about different JavaScript object classes, see the [Db2 Web Query Designer JavaScript API Classes](#) on page 36 reference information below.

1. Open Db2 Web Query Designer. On the Db2 Web Query Hub, click the plus menu and click *Create Visualizations*, or, on the Db2 Web Query Home Page, click *Visualize Data*.

Db2 Web Query Designer opens in a new browser tab.

2. Select a workspace and a data source available from that workspace.

Once you select a data source, Db2 Web Query Designer loads with options to create a single content item.

3. Click the *Convert to page* button to transform the visualization from a single content item to a page.
4. Optionally, create new content in the visualization and add new containers to the page for additional content.
5. Select a container on the page.
6. On the Settings tab of the Properties panel, in the text box for the *Classes* property, type a class name of your choice.
7. Click *Outline* on the sidebar to display the visualization outline.
8. Click *JavaScript*.

The JavaScript text editor opens.

9. To have the button and menu item appear when the page loads, use the *iba_pageloaded* event listener by adding the following syntax:

```
window.addEventListener("iba_pageloaded", function (e){
});
```

The code to add a button and menu item will go within this event listener command.

10. Add a button to the panel using JavaScript.

- a. First, within the *iba_pageloaded* event listener, create a variable to represent the panel, such as the following:

```
var panel = document.querySelector(".class").ibaObject;
```

where:

panel

Is the name of the variable that you define to represent the panel.

class

Is the name you previously typed in the *Classes* property for the panel on the page.

- b. After the panel variable, define the style and appearance of the button and add it to the panel, such as in the following syntax example. This example uses a home icon and places the button before the default resize button.

```
var button = panel.addButton(  
  {"glyphClasses": "fa fa-home", "class": "buttonClass",  
   "tooltip": "tooltip text"},  
  ".pd-container-title-button-resize", true);
```

where:

button

Is the name of the variable that you define to represent the button.

panel

Is the variable representing the panel, defined in step 10a.

fa fa-home

Is a glyph class for a Font Awesome icon that looks like a house.

buttonClass

Is a CSS class assigned to the button itself. You can use this to apply CSS styling to the button, or reference the button elsewhere in your JavaScript code.

tooltip text

Is the text that you want to appear in the tooltip when you point to the button.

.pd-container-title-button-resize

Is the class of the resize button on the panel. This is the sibling of the button you are currently adding.

true

When the value of the before property is set to *true*, the button is placed before the sibling. Otherwise, it is placed after the sibling. If no sibling is specified, it is ordered last.

- c. After defining the panel button, create an event listener that allows the button to execute a specified action. The following example causes the button to open the IBM website in a new window when it is clicked.

```
button.addEventListener("click", function(){  
  window.open("https://www.ibm.com/");  
});
```

where:

button

Is the variable name that you assigned to the button in step 10b.

11. Add a menu item to the panel's run-time menu using JavaScript.

- a. Create a variable to define the menu item, such as in the following example.

```
var menu = panel.addItem({
  "text": "Menu text", "glyphClasses": "fa fa-globe", "class":
  "menu-class"},
  ".class>.ibx_menu_item", true);
```

where:

menu

Is a variable name that you want to use to represent the menu item.

panel

Is the name of the variable representing the panel to which the menu is added.

Menu text

Is the text of the menu item.

fa fa-globe

Is a glyph class for a Font Awesome icon that looks like the Earth.

menu-class

Is a CSS class assigned to the menu item itself. You can use this to apply CSS styling to the menu item, or reference the menu item elsewhere in your JavaScript code.

".class>.ibx_menu_item"

Is a selector for the Refresh button in the panel menu, where *class* is the CSS class assigned to the panel using the *Classes* property.

true

When the value of the *before* property is set to *true*, the menu item is placed before the sibling, which in this case is the Refresh option. Otherwise, it is placed after the sibling. If no sibling is specified, it is ordered last in the menu.

- b. After defining the menu item, create an event listener that allows the button to execute a specified action. The following example causes the menu item to run an HTML page in a new window through a URL call when it is clicked.

```
menu.addEventListener("click", function(){
  window.open(
    "http[s]://hostname:12331/webquery/run.bip?
    BIP_REQUEST_TYPE=BIP_LAUNCH&BIP_folder=IBFS%253A%252FWFC%252FRRepository
    %252FPublic%252F2019%252FHTML%252F&BIP_item=html_page.htm"
  );
});
```

where:

menu

Is the variable name that you assigned to the menu item in step 11a.

The complete custom JavaScript may resemble the following.

```
window.addEventListener("iba_pageloaded", function (e){
  var panel = document.querySelector(".map-panel").ibaObject;
  var ibmsite = panel.addButton({
    "glyphClasses": "fa fa-home", "class": "ibmButton", "tooltip": "Click
to display help."},
    ".pd-container-title-button-resize", true);
  ibmsite.addEventListener("click", function(){
    window.open("https://www.ibm.com/");});
  var runReport = panel.addMenuItem({
    "text": "Country Report", "glyphClasses": "fa fa-globe", "class":
"globemenu"},
    ".map-panel>.ibx_menu_item", true);
  runReport.addEventListener("click", function(){
    window.open(
      "http[s]://hostname:12331/webquery/run.bip?
BIP_REQUEST_TYPE=BIP_LAUNCH&BIP_folder=IBFS%253A%252FWFC%252FRepository%252FPublic
%252F2019%252FHTML%252F&BIP_item=Basic_2_ctrl_page.htm");
    });
});
```

12. Save the page.

Changes made using custom CSS and JavaScript do not appear on the canvas at design time. To see the impact of your custom code, you must run the visualization.

13. On the Hub or Home Page, right-click the visualization that you just created and click *Run in new window*.

The visualization opens in a new browser tab or window, and the customized button and menu item are available. When clicked, they open the links you specified for them in a new tab or window.

Procedure: How to Use JavaScript to Automatically Resize an Image on a Page

Since the panels in a responsive page resize to fit the browser window, images added to the page may not always fit properly. If the image is too big, only part of the image may show in the panel, and scrollbars may be added. If the image is too small, the panel background may show around the image.

In order to autosize an image to fit a panel, you can use jQuery to set the height and width CSS properties for an image in a specified container. You can specify the container by referencing its class, which you can set using the Classes property on the Settings tab.

An example of the final JavaScript code is shown below:

```
$('.class').closest('.grid-stack-item').addClass('autoheight');
$('.class iframe').attr('scrolling','no').on('load', function(e){
    $(this).contents().find('img').height('100%');
    $(this).contents().find('img').width('100%');
})
```

where:

class

Is a class assigned to the panel that contains the image to be autosized.

`.attr('scrolling','no')`

Disables the generation of scrollbars in the container.

`.height('100%'), .width('100%')`

Sets the image to fill the height and width of the container. This may distort the image, depending on the dimensions of the container. As an alternative, replace *100%* with *auto* for either the height or width, to maintain the height-to-width ratio at all sizes. Using *auto* may reveal the panel background or cut off part of the image, depending on the dimensions of the container.

1. On the Hub or Home Page, click the plus button and then click *Assemble Visualizations*.
Db2 Web Query Designer opens to assemble a page from existing content.
2. With *Content* selected on the sidebar, navigate to the location of an image that has been uploaded to the Repository and drag the image onto the canvas.

The image is added to a container on the page. The size of the image in the container is the absolute size of the uploaded image file.
3. Optionally, resize and reposition the container that contains the image, and add other items to the page.
4. Select the container that holds the image. On the *Settings* tab of the Properties panel, type a class name of your choice into the Classes text box. This class will be used as an identifier for the container in the JavaScript code.
5. Click *Outline* on the sidebar, and then click *JavaScript*.

The JavaScript text editor opens.

6. Paste the following code into the text editor.

```
$('.class').closest('.grid-stack-item').addClass('autoheight');
$('.class iframe').attr('scrolling','no').on('load', function(e){
    $(this).contents().find('img').height('100%');
    $(this).contents().find('img').width('100%');
})
```

7. Replace *class* in the first two lines with the class name that you specified for the container in Step 4.
8. Click the red X in the top-right corner of the page to return to the page canvas.

The image still displays at its original size. The height and width properties are only applied when the page is run.
9. Optionally, on the canvas, select the container that contains the image and, on the *Settings* tab on the Properties panel, clear the *Show title* and *Show toolbar* check boxes to hide the container title and toolbar, so that the image is the only item in the container.
10. On the Visualization toolbar, click *Run in new window* to run the page and execute the JavaScript code that you just added.

The image stretches to fill the entire container at run time.

Db2 Web Query Designer JavaScript API Classes

As part of the Db2 Web Query Designer JavaScript API, you can use the following predefined classes.

- ☐ **ibaObject.** The base automation object.
- ☐ **ibaAccordionContainer.** Accordion container automation object.
- ☐ **ibaCarouselContainer.** Carousel container automation object.
- ☐ **ibaContent.** Content panel automation object.
- ☐ **ibaPage.** Page automation object. The event `iba_pageloading` will be triggered on the global window object. `event.data` will be the `ibaPage` object.
- ☐ **ibaPanelContainer.** Panel container automation object.
- ☐ **ibaPortal.** Portal automation object.
- ☐ **ibaSection.** Page section automation object.
- ☐ **ibaTabContainer.** Tab container automation object.

A help plug-in with information about each class is packaged with your Db2 Web Query installation. It is available through a URL at the following address:

[http\[s\]://hostname:12331/webquery/web_resource/doc/automation/index.html](http[s]://hostname:12331/webquery/web_resource/doc/automation/index.html)

where:

hostname

Is the name of the machine on which the Db2 Web Query Client is installed.

Class: *ibaObject*

new ibaObject(element)

Is the base automation object. Automation objects are created internally by the system. You never directly instantiate an automation object.

where:

element

Is the node that is being automated.

Available methods include the following:

❑ **classId()**

Returns a string, which is the unique, automatically generated class ID of the object being automated. It can be used to filter enumerations.

❑ **customClasses(classes, remove)**

Returns a string. Sets or gets custom classes.

where:

classes

String.

Optional. A space-separated list of classes to be added or removed.

remove

Boolean.

Optional. If true, remove the classes specified in *classes*, otherwise add them.

❑ **element()**

Returns an element, the DOM node that is being automated by this object.

Class: *ibaPage*

```
new ibaPage(element)
```

Is the page automation object. Automation objects are created internally by the system. You never directly instantiate an automation object. The event `iba_pageloading` will be triggered on the global window object. `event.data` will be the `ibaPage` object.

where:

element

Is the node that is being automated.

The following triggers can be used with the page object:

- ❑ **event:iba_pageloaded.** The page is loaded in the running state.
- ❑ **event:iba_beforecontentdescribe.** Before the page content filtering information has been retrieved.
- ❑ **event:iba_contentdescribed.** After the page content filtering information has been retrieved, but before it has been processed.
- ❑ **event:iba_beforecontentload.** Before the content of a panel is about to load.

These are used in the following JavaScript syntax example:

```
window.addEventListener("iba_pageloading", function (e){
    var page = e.data;
    page.element().addEventListener("iba_pageloaded", function(e){
        var page = e.data;
    });
    page.element().addEventListener("iba_beforecontentdescribe", function (e){
        var describeInfo = e.data;
        var path = describeInfo.path; // The path to the content being described
        // e.preventDefault() will stop the describe from happening
    });
    page.element().addEventListener("iba_contentdescribed", function (e){
        var describeInfo = e.data;
        var path = describeInfo.path; // The path to the content being described
        var wfDescribe = describeInfo.wfDescribe; // The filtering information of the
content being described
    });
    page.element().addEventListener("iba_beforecontentload", function (e){
        var loadInfo = e.data;
        var path = loadInfo.path; // The path to the content being described
        var defaultValues = loadInfo.defaultValues; // If default values are used to
run the content
        // e.preventDefault() will stop the page from retrieving the content
    });
});
```

If you add multiple pages with custom JavaScript to the same portal, the `iba_pageloading` event listener, for the global window object, will execute whenever any page in the portal loads. To execute your code only when a specific page loads, remove the event listener after the code executes for the desired page using the `window.removeEventListener` method.

The `iba_pageloading` event listener can be used to access the page being loaded, since it receives a page object as a parameter. The page object can be stored or used to assign additional event handlers to the page as shown below:

```
function onPageLoading(e) {
    var page = e.data;
    page.element().addEventListener("iba_pageloaded", function(e){
        on load code
    });
    page.element().addEventListener("click", function(e)
    {
        on click code
    });
    window.removeEventListener("iba_pageloading", onPageLoading);
}
```

where:

`function onPageLoading`

Is an `iba_pageloading` event listener function.

`var page = e.data`

Accesses the page object via the event parameter.

`on load code`

Code to be executed when the page loads.

`on click code`

Code to be executed when the page is clicked.

`window.removeEventListener("iba_pageloading", onPageLoading);`

Removes the `iba_pageloading` listener.

You can use the `context` member object to supply the context information for the page.

Available methods include the following:

❏ `addButton(options, sibling, before)`

Returns an element, adding a button to the title bar of the page.

where:

options

Object.

One or more button options, which can include the following:

- ❑ **icon.** A URL to an image file in .jpeg, .png, or .gif format.
- ❑ **glyph.** A glyph or ligature, such as a Material icon.
- ❑ **glyphClasses.** Glyph classes. For example, 'material-icons'.
- ❑ **class.** Additional CSS classes.
- ❑ **tooltip.** A tooltip for the button.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add button before or after the specified sibling.

❑ **addSection(options, sibling, before)**

Returns an element, adding a section to the page.

where:

options

Object.

One or more section options, which can include the following:

- ❑ **collapsible.** A Boolean value. When true, the section is collapsible.
- ❑ **collapsed.** A Boolean value. When true, the section is created in a collapsed state.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add the section before or after the specified sibling.

❑ **buttons(selector)**

Returns the page title bar buttons as an array.

where:

selector

Selector, element, or jQuery.

Is the buttons selector. All is the default.

❑ **classId()**

Returns a string, which is the unique, automatically generated class ID of the object being automated. It can be used to filter enumerations.

❑ **containers(selector)**

Returns the page containers as an array.

where:

selector

Selector, element, or jQuery.

Is the containers selector. All is the default.

❑ **customClasses(classes, remove)**

Returns a string. Sets or gets custom classes.

where:

classes

String.

Optional. A space-separated list of classes to be added or removed.

remove

Boolean.

Optional. If true, remove the classes specified in *classes*, otherwise add them.

❑ **element()**

Returns an element, the DOM node that is being automated by this object.

❑ **refreshFilters()**

Use to redescribe the existing content and recreate filter panels.

❑ **removeButton(*button*)**

Removes one or more buttons from the title bar of the page.

where:

button

Selector, element, or jQuery.

Optional. Is a button selector. All is the default.

❑ **removeSection(*selector*)**

Removes one or more sections from the page.

where:

selector

Selector, element, or jQuery.

Optional. Is a section selector. All is the default.

❑ **sections(*selector*)**

Returns the page sections as an array.

where:

selector

Selector, element, or jQuery.

Is the section selector. All is the default.

❑ **title(*title*)**

Sets the title of the page. If *title* is not passed, the title of the container is returned as a string or jQuery.

where:

title

String.

Optional. Is the new title of the page.

Class: *ibaSection*

The *ibaSection* class extends *ibaObject*.

new ibaSection(element)

Is the page section automation object. Automation objects are created internally by the system. You never directly instantiate an automation object.

where:

element

Is the DOM node that is being automated.

Available methods include the following:

❑ ***addContainer(type, title, col, row, colSpan, rowSpan)***

Adds a new container to the section.

where:

type

String.

Is one of the following container types:

❑ 'pane'

❑ 'tab'

❑ 'accordion'

❑ 'carousel'

title

String.

Is the container title.

col

Integer.

Optional. The column of the section to insert to. By default, the container is added to the first available column.

row

Integer.

Optional. The row of the section to insert to. By default, the container is added to the first available row.

colSpan

Integer.

Optional. The width of the container in columns. 3 is the default.

rowSpan

Integer.

Optional. The height of the container in rows. 5 is the default.

❑ **classId()**

Returns a string, which is the unique, automatically generated class ID of the object being automated. It can be used to filter enumerations.

❑ **collapsed(*collapsed*)**

Makes the section collapsed, or returns the current collapsed state or a chainable automation object.

where:

collapsed

Boolean.

Optional. If true, collapse the section. Otherwise, expand.

❑ **collapsible(*collapsible*)**

Makes the section collapsible, or returns the current collapsible state or a chainable automation object.

where:

collapsible

Boolean.

Optional. If true, make the section collapsible.

❑ **containers(*selector*)**

Returns the containers in the section as an array.

where:

selector

Selector, element, or jQuery.

An optional containers selector. All is the default.

❑ **customClasses(*classes*, *remove*)**

Returns a string. Sets or gets custom classes.

where:

classes

String.

Optional. A space-separated list of classes to be added or removed.

remove

Boolean.

Optional. If true, remove the classes specified in *classes*, otherwise add them.

❑ **element()**

Returns an element, the DOM node that is being automated by this object.

Class: *ibaContainer*

`new ibaContainer(element)`

Is the container automation object. Automation objects are created internally by the system. You never directly instantiate an automation object.

where:

element

Is the DOM node that is being automated.

Available methods include the following:

❑ **addButton(*options*, *sibling*, *before*)**

Returns an element, adding a button to the title bar of the container.

where:

options

Object.

One or more button options, which can include the following:

- ❑ **icon.** A URL to an image file in .jpeg, .png, or .gif format.
- ❑ **glyph.** A glyph or ligature, such as a Material icon.
- ❑ **glyphClasses.** Glyph classes. For example, 'material-icons'.
- ❑ **class.** Additional CSS classes.
- ❑ **tooltip.** A tooltip for the button.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add button before or after the specified sibling.

❑ **addContent(*path*, *description*, *replaceExisting*)**

Returns elements. Adds new content by replacing the content in the current content panel or adding a sub-panel.

where:

path

String.

Path of the content being added.

description

String.

Description of the content being added.

replaceExisting

Boolean.

Optional. When true, replace the existing content. When false, which is the default, add new content.

❑ **addMenuItem(*options*, *sibling*, *before*)**

Returns an element, adding a menu item to the container menu.

where:

options

Object.

One or more menu item options, which can include the following:

- ☐ **text.** The menu item text.
- ☐ **icon.** A URL to an image file in .jpeg, .png, or .gif format.
- ☐ **glyph.** A glyph or ligature, such as a Material icon.
- ☐ **glyphClasses.** Glyph classes. For example, 'material-icons'.
- ☐ **class.** Additional CSS classes.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add the menu item before or after the specified sibling.

☐ **addMenuSeparator(*options*, *sibling*, *before*)**

Returns an element, adding a separator to the container menu.

where:

options

Object.

Menu separator options. You can use the class property to specify additional CSS classes.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add menu separator before or after the specified sibling.

❑ **buttons(*selector*)**

Returns the container title bar buttons as an array.

where:

selector

Selector, element, or jQuery.

Is the buttons selector. All is the default.

❑ **classId()**

Returns a string, which is the unique, automatically generated class ID of the object being automated. It can be used to filter enumerations.

❑ **contents(*selector*)**

Returns the content panels in the container as an array.

where:

selector

Selector, element, or jQuery.

Is the content selector. All is the default.

❑ **customClasses(*classes*, *remove*)**

Returns a string. Sets or gets custom classes.

where:

classes

String.

Optional. A space-separated list of classes to be added or removed.

remove

Boolean.

Optional. If true, remove the classes specified in *classes*, otherwise add them.

❑ **element()**

Returns an element, the DOM node that is being automated by this object.

❑ **removeButton(*button*)**

Removes one or more buttons from the title bar of the container.

where:

button

Selector, element, or jQuery.

Optional. Is a button selector. All is the default.

❑ **removeContent(selector)**

Removes content from the container.

where:

selector

Selector, element, or jQuery.

Optional. Is a content selector. All is the default.

❑ **removeMenuItem(selector)**

Removes one or more menu items from the container menu.

where:

selector

Selector, element, or jQuery.

Optional. Is a menu item selector. All is the default.

❑ **removeMenuSeparator(selector)**

Removes one or more separators from the container.

where:

selector

Selector, element, or jQuery.

Optional. Is a menu separator selector. All is the default.

❑ **title(title)**

Sets the title of the container. If *title* is not passed, the title of the container is returned as a string or jQuery.

where:

title

String.

Optional. Is the new title of the container.

Class: ibaAccordionContainer

The ibaAccordionContainer class extends ibaContainer.

```
new ibaAccordionContainer(element)
```

Is the accordion container automation object. Automation objects are created internally by the system. You never directly instantiate an automation object.

where:

element

Is the DOM node that is being automated.

Available methods include the following:

❑ **addButton(*options*, *sibling*, *before*)**

Returns an element, adding a button to the title bar of the accordion container.

where:

options

Object.

One or more button options, which can include the following:

- ❑ **icon.** A URL to an image file in .jpeg, .png, or .gif format.
- ❑ **glyph.** A glyph or ligature, such as a Material icon.
- ❑ **glyphClasses.** Glyph classes. For example, 'material-icons'.
- ❑ **class.** Additional CSS classes.
- ❑ **tooltip.** A tooltip for the button.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add button before or after the specified sibling.

❑ **addContent(*path*, *description*, *replaceExisting*)**

Returns elements. Adds new content by replacing the content in the current content panel or adding a new accordion tab.

where:

path

String.

Path of the content being added.

description

String.

Description of the content being added.

replaceExisting

Boolean.

Optional. When true, replace the existing content. When false, which is the default, add new content.

❑ **addMenuItem(*options*, *sibling*, *before*)**

Returns an element, adding a menu item to the accordion container menu.

where:

options

Object.

One or more menu item options, which can include the following:

❑ **text.** The menu item text.

❑ **icon.** A URL to an image file in .jpeg, .png, or .gif format.

❑ **glyph.** A glyph or ligature, such as a Material icon.

❑ **glyphClasses.** Glyph classes. For example, 'material-icons'.

❑ **class.** Additional CSS classes.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add the menu item before or after the specified sibling.

❑ **addMenuSeparator(*options*, *sibling*, *before*)**

Returns an element, adding a separator to the accordion container menu.

where:

options

Object.

Menu separator options. You can use the class property to specify additional CSS classes.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add menu separator before or after the specified sibling.

❑ **buttons(*selector*)**

Returns the container title bar buttons as an array.

where:

selector

Selector, element, or jQuery.

Is the buttons selector. All is the default.

❑ **classId()**

Returns a string, which is the unique, automatically generated class ID of the object being automated. It can be used to filter enumerations.

❑ **contents(*selector*)**

Returns the content panels in the container as an array.

where:

selector

Selector, element, or jQuery.

Is the content selector. All is the default.

❑ **customClasses(*classes*, *remove*)**

Returns a string. Sets or gets custom classes.

where:

classes

String.

Optional. A space-separated list of classes to be added or removed.

remove

Boolean.

Optional. If true, remove the classes specified in *classes*, otherwise add them.

❑ **element()**

Returns an element, the DOM node that is being automated by this object.

❑ **removeButton(*button*)**

Removes one or more buttons from the title bar of the container.

where:

button

Selector, element, or jQuery.

Optional. Is a button selector. All is the default.

❑ **removeContent(*selector*)**

Removes content from the container.

where:

selector

Selector, element, or jQuery.

Optional. Is a content selector. All is the default.

❏ **removeMenuItem(selector)**

Removes one or more menu items from the container menu.

where:

selector

Selector, element, or jQuery.

Optional. Is a menu item selector. All is the default.

❏ **removeMenuSeparator(selector)**

Removes one or more separators from the container.

where:

selector

Selector, element, or jQuery.

Optional. Is a menu separator selector. All is the default.

❏ **selectContent(selector)**

Returns an element. Selects a content pane in the accordion container.

where:

selector

Selector, element, or jQuery.

Optional. Is a content selector. All is the default.

❏ **title(title)**

Sets the title of the container. If *title* is not passed, the title of the container is returned as a string or jQuery.

where:

title

String.

Optional. Is the new title of the container.

Class: *ibaCarouselContainer*

The *ibaCarouselContainer* class extends *ibaContainer*.

new ibaCarouselContainer(element)

Is the carousel container automation object. Automation objects are created internally by the system. You never directly instantiate an automation object.

where:

element

Is the DOM node that is being automated.

Available methods include the following:

❑ **addButton(options, sibling, before)**

Returns an element, adding a button to the title bar of the carousel container.

where:

options

Object.

One or more button options, which can include the following:

- ❑ **icon.** A URL to an image file in .jpeg, .png, or .gif format.
- ❑ **glyph.** A glyph or ligature, such as a Material icon.
- ❑ **glyphClasses.** Glyph classes. For example, 'material-icons'.
- ❑ **class.** Additional CSS classes.
- ❑ **tooltip.** A tooltip for the button.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add button before or after the specified sibling.

❑ **addContent(path, description, replaceExisting)**

Returns elements. Adds new content by replacing the content in the current content panel or adding a new carousel slide.

where:

path

String.

Path of the content being added.

description

String.

Description of the content being added.

replaceExisting

Boolean.

Optional. When true, replace the existing content. When false, which is the default, add new content.

❑ **addMenuItem(*options*, *sibling*, *before*)**

Returns an element, adding a menu item to the carousel container menu.

where:

options

Object.

One or more menu item options, which can include the following:

- ❑ **text.** The menu item text.
- ❑ **icon.** A URL to an image file in .jpeg, .png, or .gif format.
- ❑ **glyph.** A glyph or ligature, such as a Material icon.
- ❑ **glyphClasses.** Glyph classes. For example, 'material-icons'.
- ❑ **class.** Additional CSS classes.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add the menu item before or after the specified sibling.

❑ **addMenuSeparator(*options*, *sibling*, *before*)**

Returns an element, adding a separator to the carousel container menu.

where:

options

Object.

Menu separator options. You can use the class property to specify additional CSS classes.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add menu separator before or after the specified sibling.

❑ **buttons(*selector*)**

Returns the container title bar buttons as an array.

where:

selector

Selector, element, or jQuery.

Is the buttons selector. All is the default.

❑ **classId()**

Returns a string, which is the unique, automatically generated class ID of the object being automated. It can be used to filter enumerations.

❑ **contents(*selector*)**

Returns the content panels in the container as an array.

where:

selector

Selector, element, or jQuery.

Is the content selector. All is the default.

❑ **customClasses(*classes*, *remove*)**

Returns a string. Sets or gets custom classes.

where:

classes

String.

Optional. A space-separated list of classes to be added or removed.

remove

Boolean.

Optional. If true, remove the classes specified in *classes*, otherwise add them.

❑ **element()**

Returns an element, the DOM node that is being automated by this object.

❑ **removeButton(*button*)**

Removes one or more buttons from the title bar of the container.

where:

button

Selector, element, or jQuery.

Optional. Is a button selector. All is the default.

❑ **removeContent(*selector*)**

Removes content from the container.

where:

selector

Selector, element, or jQuery.

Optional. Is a content selector. All is the default.

❏ removeMenuItem(selector)

Removes one or more menu items from the container menu.

where:

selector

Selector, element, or jQuery.

Optional. Is a menu item selector. All is the default.

❏ removeMenuSeparator(selector)

Removes one or more separators from the container.

where:

selector

Selector, element, or jQuery.

Optional. Is a menu separator selector. All is the default.

❏ selectContent(selector)

Returns an element. Selects a content pane in the carousel container.

where:

selector

Selector, element, or jQuery.

Optional. Is a content selector. All is the default.

❏ title(title)

Sets the title of the container. If *title* is not passed, the title of the container is returned as a string or jQuery.

where:

title

String.

Optional. Is the new title of the container.

Class: ibaPanelContainer

The ibaPanelContainer class extends ibaContainer.

`new ibaPanelContainer(element)`

Is the panel container automation object. Automation objects are created internally by the system. You never directly instantiate an automation object.

where:

element

Is the DOM node that is being automated.

Available methods include the following:

❑ **addButton(*options*, *sibling*, *before*)**

Returns an element, adding a button to the title bar of the panel container.

where:

options

Object.

One or more button options, which can include the following:

- ❑ **icon.** A URL to an image file in .jpeg, .png, or .gif format.
- ❑ **glyph.** A glyph or ligature, such as a Material icon.
- ❑ **glyphClasses.** Glyph classes. For example, 'material-icons'.
- ❑ **class.** Additional CSS classes.
- ❑ **tooltip.** A tooltip for the button.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add button before or after the specified sibling.

❑ **addContent(*path*, *description*, *replaceExisting*)**

Returns elements. Adds new content by replacing the content in the current content panel or adding a new panel.

where:

path

String.

Path of the content being added.

description

String.

Description of the content being added.

replaceExisting

Boolean.

Optional. When true, replace the existing content. When false, which is the default, add new content.

❑ **addMenuItem(*options*, *sibling*, *before*)**

Returns an element, adding a menu item to the container menu.

where:

options

Object.

One or more menu item options, which can include the following:

❑ **text.** The menu item text.

❑ **icon.** A URL to an image file in .jpeg, .png, or .gif format.

❑ **glyph.** A glyph or ligature, such as a Material icon.

❑ **glyphClasses.** Glyph classes. For example, 'material-icons'.

❑ **class.** Additional CSS classes.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add the menu item before or after the specified sibling.

❏ **addMenuSeparator(*options*, *sibling*, *before*)**

Returns an element, adding a separator to the container menu.

where:

options

Object.

Menu separator options. You can use the class property to specify additional CSS classes.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add menu separator before or after the specified sibling.

❏ **buttons(*selector*)**

Returns the container title bar buttons as an array.

where:

selector

Selector, element, or jQuery.

Is the buttons selector. All is the default.

❏ **classId()**

Returns a string, which is the unique, automatically generated class ID of the object being automated. It can be used to filter enumerations.

❏ **contents(*selector*)**

Returns the content panels in the container as an array.

where:

selector

Selector, element, or jQuery.

Is the content selector. All is the default.

❑ **customClasses(*classes*, *remove*)**

Returns a string. Sets or gets custom classes.

where:

classes

String.

Optional. A space-separated list of classes to be added or removed.

remove

Boolean.

Optional. If true, remove the classes specified in *classes*, otherwise add them.

❑ **element()**

Returns an element, the DOM node that is being automated by this object.

❑ **removeButton(*button*)**

Removes one or more buttons from the title bar of the container.

where:

button

Selector, element, or jQuery.

Optional. Is a button selector. All is the default.

❑ **removeContent(*selector*)**

Removes content from the container.

where:

selector

Selector, element, or jQuery.

Optional. Is a content selector. All is the default.

❑ **removeMenuItem(*selector*)**

Removes one or more menu items from the container menu.

where:

selector

Selector, element, or jQuery.

Optional. Is a menu item selector. All is the default.

❏ **removeMenuSeparator(*selector*)**

Removes one or more separators from the container.

where:

selector

Selector, element, or jQuery.

Optional. Is a menu separator selector. All is the default.

❏ **title(*title*)**

Sets the title of the container. If *title* is not passed, the title of the container is returned as a string or jQuery.

where:

title

String.

Optional. Is the new title of the container.

Class: ibaTabContainer

The ibaTabContainer class extends ibaContainer.

`new ibaTabContainer(element)`

Is the carousel container automation object. Automation objects are created internally by the system. You never directly instantiate an automation object.

where:

element

Is the DOM node that is being automated.

Available methods include the following:

❏ **addButton(*options*, *sibling*, *before*)**

Returns an element, adding a button to the title bar of the tab container.

where:

options

Object.

One or more button options, which can include the following:

- ❑ **icon.** A URL to an image file in .jpeg, .png, or .gif format.
- ❑ **glyph.** A glyph or ligature, such as a Material icon.
- ❑ **glyphClasses.** Glyph classes. For example, 'material-icons'.
- ❑ **class.** Additional CSS classes.
- ❑ **tooltip.** A tooltip for the button.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add button before or after the specified sibling.

❑ **addContent(*path*, *description*, *replaceExisting*)**

Returns elements. Adds new content by replacing the content in the current content panel or adding a new tab.

where:

path

String.

Path of the content being added.

description

String.

Description of the content being added.

replaceExisting

Boolean.

Optional. When true, replace the existing content. When false, which is the default, add new content.

❑ **addMenuItem(*options*, *sibling*, *before*)**

Returns an element, adding a menu item to the tab container menu.

where:

options

Object.

One or more menu item options, which can include the following:

- ☐ **text.** The menu item text.
- ☐ **icon.** A URL to an image file in .jpeg, .png, or .gif format.
- ☐ **glyph.** A glyph or ligature, such as a Material icon.
- ☐ **glyphClasses.** Glyph classes. For example, 'material-icons'.
- ☐ **class.** Additional CSS classes.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add the menu item before or after the specified sibling.

☐ **addMenuSeparator(*options*, *sibling*, *before*)**

Returns an element, adding a separator to the tab container menu.

where:

options

Object.

Menu separator options. You can use the class property to specify additional CSS classes.

sibling

Selector, element, or jQuery.

Optional. A sibling to add before or after. The default is Last.

before

Boolean.

Add menu separator before or after the specified sibling.

❏ buttons(selector)

Returns the container title bar buttons as an array.

where:

selector

Selector, element, or jQuery.

Is the buttons selector. All is the default.

❏ classId()

Returns a string, which is the unique, automatically generated class ID of the object being automated. It can be used to filter enumerations.

❏ contents(selector)

Returns the content panels in the container as an array.

where:

selector

Selector, element, or jQuery.

Is the content selector. All is the default.

❏ customClasses(classes, remove)

Returns a string. Sets or gets custom classes.

where:

classes

String.

Optional. A space-separated list of classes to be added or removed.

remove

Boolean.

Optional. If true, remove the classes specified in *classes*, otherwise add them.

❏ element()

Returns an element, the DOM node that is being automated by this object.

❏ removeButton(button)

Removes one or more buttons from the title bar of the container.

where:

button

Selector, element, or jQuery.

Optional. Is a button selector. All is the default.

❑ **removeContent(selector)**

Removes content from the container.

where:

selector

Selector, element, or jQuery.

Optional. Is a content selector. All is the default.

❑ **removeMenuItem(selector)**

Removes one or more menu items from the container menu.

where:

selector

Selector, element, or jQuery.

Optional. Is a menu item selector. All is the default.

❑ **removeMenuSeparator(selector)**

Removes one or more separators from the container.

where:

selector

Selector, element, or jQuery.

Optional. Is a menu separator selector. All is the default.

❑ **selectContent(selector)**

Returns an element. Selects a content pane in the tab container.

where:

selector

Selector, element, or jQuery.

Optional. Is a content selector. All is the default.

❏ title(*title*)

Sets the title of the container. If *title* is not passed, the title of the container is returned as a string or jQuery.

where:

title

String.

Optional. Is the new title of the container.

Class: *ibaContent*

The *ibaContent* class extends *ibaObject*.

`new ibaContent(element)`

Is the content panel automation object. Automation objects are created internally by the system. You never directly instantiate an automation object.

where:

element

Is the DOM node that is being automated.

Available methods include the following:

❏ classId()

Returns a string, which is the unique, automatically generated class ID of the object being automated. It can be used to filter enumerations.

❏ customClasses(*classes*, *remove*)

Returns a string. Sets or gets custom classes.

where:

classes

String.

Optional. A space-separated list of classes to be added or removed.

remove

Boolean.

Optional. If true, remove the classes specified in *classes*, otherwise add them.

❏ **element()**

Returns an element, which is the DOM node that is being automated by this object.

❏ **path()**

Returns the path of the content as a string.

Adding Panels to a Responsive Container

You can utilize responsive panel in panel behavior by adding items to a panel in panel container. When you run the page, top-level containers in the page, including the panel group container, fold responsively based on the size of the browser, and the panels within the panel group container fold based on its current size, independent of the rest of the page. Additionally, items in the panel group container are kept together when folding occurs, making the panel group container a good way to group together closely related content that you want to be visually separated from the other items on the page. This is particularly true when related items are arranged vertically, as items typically wrap based on rows.

To create a panel group container, click the *Container* tab on the sidebar, select the *Panel group* container, and drag it onto the canvas. You can then resize it like any other container.

To add a panel to a panel group container, drag a new content item or another container from the *Container* tab on the sidebar into it. You can then resize the new container inside the panel group container. You can also add an existing container on the page to a panel group by holding the Ctrl key and dragging the container into it.

Once you have created it, you can right-click an area of blank space within the panel group to duplicate it, delete the panel group and its contents, or remove the panel group but keep its constituent containers on the page.

You can also select the panel group and use the Properties panel to add a toolbar and title to the panel group container. Properties applied to the entire page or to the panel group container, such as margins or styling, affect all panels within it, unless specifically overridden.

Linking to External Content From a Page

The link tile widget provides additional innovative ways of using your content and incorporating it into your page. The link tile widget layers content in such a way that one content item becomes a click-through tile that opens another content item. This feature is especially useful when you need to incorporate a large or Insight-enabled item into a page and display it on all devices. The link tile widget can be used in pages assembled from existing content.

To add a link tile to a page, drag the link tile object from the Resources panel with *Container* selected on the sidebar. You configure a link tile using the Settings tab of the Properties panel, as shown in the following image.

The image shows a configuration panel for a 'Link Tile' widget. At the top, there are three device icons: a desktop monitor, a tablet, and a smartphone. Below these is the title 'Link Tile' with an upward-pointing arrow. The panel is divided into sections: 'Background' with a 'Not selected' dropdown and a menu icon; a 'Tooltip' checkbox; an empty text input field; 'Content' with a 'Not selected' dropdown and a menu icon; an 'Attach all parameters' checkbox; and 'Target' with a 'Viewer' dropdown menu.

When adding a link tile widget to a page, you configure the content that you want to display on the page using the Background property, the item that you want to link to using the Content property, and the location in which to open the linked content using the Target property. You can also add a tooltip to the link tile container to provide a description of the target content, providing information about what happens when a user clicks the link tile.

If the link tile background content is a procedure, filters can be created for and applied to it just like any other content item. The background behaves as a static image, so interactive features, such as chart tooltips or hyperlinks, are not enabled. Link tile target content can include Db2 Web Query charts, reports, and pages, URLs, ReportCaster schedules, and more.

Note: Images, such as .jpeg, .png, or .gif files, used as the link tile background, are scaled to completely fill the link tile container. If the image is much smaller than the link tile container, it may become pixelated when it is resized to fill the frame. As a best practice, either use an image that is about the same size as the link tile container, or use an image in SVG format, which is designed to display well at any size or resolution.

Additionally, the filter selections in the page can be passed to the link tile target content if it is a procedure or a page by enabling the *Attach all parameters* option. When enabled, the link tile target is automatically filtered corresponding to the source page.

Procedure: How to Use the Link Tile Widget

1. Assemble a new page from existing content, or open an existing page in Db2 Web Query Designer.
2. On the sidebar, click the *Container* tab.
3. Drag a link tile container from the Resources panel onto the page
The link tile container initially appears blank.
4. With the link tile selected on the page, expand the Link Tile area of the Settings tab on the Properties panel.

Note: The link tile container does not include a title bar by default, so clicking the link tile on the canvas may only allow you to select the link tile background content. To select the link tile container more easily, use the outline.

5. In the Link Tile section of the *Settings* tab, click the ellipses button next to the Background option and navigate to the item from your repository that you would like to display in the link tile.

You can select a FOCEXEC procedure such as a chart or report, a page, an image, a URL, an HTML page, and more. The item behaves as an image, so interactive features such as tooltips are not enabled.

6. Click the ellipses icon next to the Content option and select the target item. This content will run when users click on the link tile panel.
7. Optionally, select the *Tooltip* check box and type a description into the text box to provide information about the link tile target content.
8. Select the *Attach all parameters* check box if parameter filter selections from the page should be passed to the target content, or leave it unselected to open the target content without passing filter selections.
9. Determine how the target item will display. Select *Viewer* to run the target content item in an overlay on the same browser tab as the link tile, or select *New window* to run the item in a new browser tab.

10. Run the visualization using the *Run in new window* button to test the link tile behavior.

When you click the link tile, your target content opens as configured.

Enabling Container Customization

When creating a page with external content, you can enable users to select content to display at run time, which can make portal pages more engaging, useful, and interactive. With this feature you can also control which items users can access, by assigning a specific path to the Add content button in the Properties panel. This feature can be used with panel, tab, carousel, and accordion containers. If you enable this feature on a multi-content container, the Add content button automatically appears on every new tab, slide, or area.

Procedure: How to Configure the Container Customization Properties

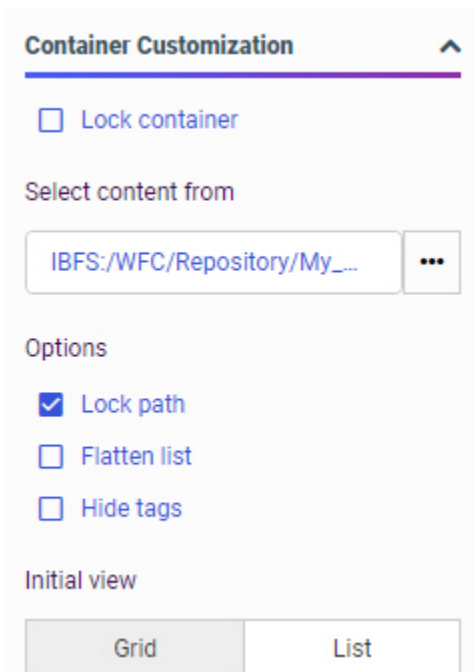
1. In a visualization using external content, drag an empty panel or a content item onto the canvas.
2. With the panel or content item selected, in the Properties panel, on the Settings tab, disable the *Lock container* option.

If you used an empty panel, the *Add content* icon displays in the middle of the selected panel.

3. If you want to direct your selection to a specific area in the repository, click the ellipses next to the *Select content from* property to select a default workspace or folder that you want to display for content selection at run time.
4. Keep the *Lock path* option selected, if you want to limit your selection to the path specified in the previous step. Otherwise, users can navigate to other workspaces to access content there in addition to the default path.
5. Optionally, select the *Flatten list* option, if you want to hide the folder hierarchy inside your selection area and only display items.
6. Optionally, select the *Hide tags* option, if you want to hide all tags from your selection area.
7. Configure the *Initial View* option, choosing either Grid or List, to set the initial appearance of the window where users select the content to display.

Note: Users can change the view at run time inside the Select Item dialog box.

The following image shows an example of the configured Container Customization properties.



The image shows a configuration panel titled "Container Customization" with an upward arrow icon. It contains several sections: "Lock container" with an unchecked checkbox; "Select content from" with a text input field containing "IBFS:/WFC/Repository/My_..." and a dropdown arrow; "Options" with three checkboxes: "Lock path" (checked), "Flatten list" (unchecked), and "Hide tags" (unchecked); and "Initial view" with two buttons, "Grid" (selected) and "List".

Container Customization ^

☐ Lock container

Select content from

IBFS:/WFC/Repository/My_... ⋮

Options

☒ Lock path

☐ Flatten list

☐ Hide tags

Initial view

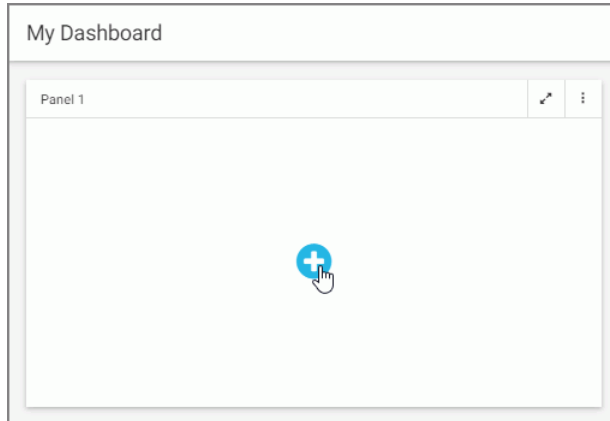
Grid List

8. Save the page.

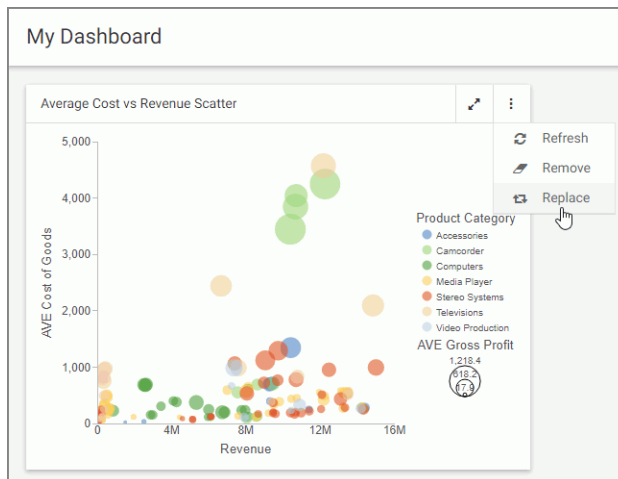
Procedure: How to Select Content at Run Time

1. Run the page that features interactive panels.

- Click the *Add content* button, as shown in the following image.



If you enabled container customization for a container with content in it, click the Options menu and then click *Replace*, as shown in the following image.



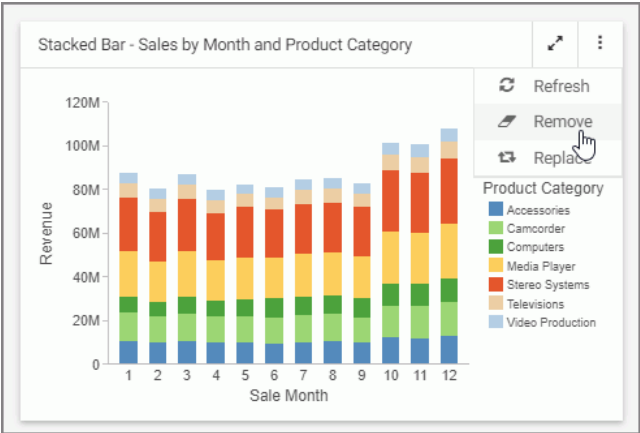
The Select Item dialog box opens.

If the *Lock path* option is disabled, you can navigate between workspaces using the breadcrumb trail.

If the *Flatten list* option is disabled, you can navigate between the folders within the selection area hierarchy.

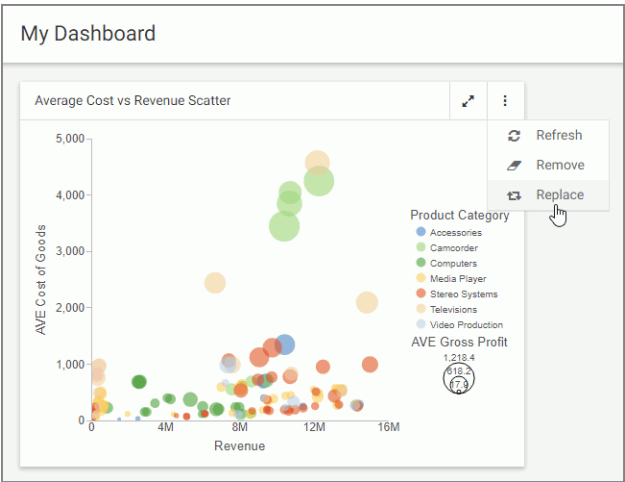
You can also use the search function, toggle between the Grid and List views, and refresh your selection area.

- 3. Make a selection and click *Add*.
The selected item displays inside the panel.
- 4. To go back to the initial view of the interactive panel, click *Options*, and then click *Remove*, as shown in the following image.



The item is removed from the panel, and the *Add content* button displays.

- 5. To replace an item with another item, click *Options*, and then click *Replace*, as shown in the following image.



The Select Item dialog box opens, and you can make a new selection.


Note: When the page is used in a portal, content customization changes that the user makes at run time to unlocked containers are remembered if the user has the Save Portal Customization privilege. If the user does not have this privilege or if the content customization changes are made to a standalone page, these changes are not remembered by Db2 Web Query.

Providing Access to Content Items and Db2 Web Query for i Tools

When assembling pages from existing content in Db2 Web Query Designer, two preset containers are available that allow you to provide access to content saved to your Repository. These are the Workbench template and the Explorer widget. In addition to running content, the Explorer widget allows users to create new content using the same options that are available on the Db2 Web Query Hub and Home Page.

The Workbench template provides a resource tree that is linked to another container on the page, using the CSS class *targetContainerForWorkBenchTree*. You can select the Workbench template from the Choose Template dialog box when you start assembling a page from existing content. It is not recommended to delete either of the containers that are added as part of the template, but if you right-click the target container and point to *Convert to*, you can change it from a tab container to a basic panel, or to an accordion or carousel container. You can also change the theme used in the page and the styles used by each of the containers. Content that you run in the workbench target container inherits the theme used in the page automatically.

When you run the page, you can run content items from your Repository, listed in the resource tree, in the section of the target container that is currently visible, by double-clicking them or by right-clicking them and clicking *Run*. To run an item in a new section of a tab, accordion, or carousel container, right-click it and click *Run in New Area*. This option is not available if you change the target container to a basic panel. Running items in a new area allows you to see the items that you previously ran by returning to the tab, accordion panel, or carousel slide where you ran them. If the items that you run include parameter filters, you can access filter

controls in a modal window by clicking *Show filters*  on the page toolbar. Note that if links within content items, such as drilldowns, links generated by Auto Linking, or hyperlinks in a web page, are set to run in a new browser tab or window, they will continue to do so when you access them in the workbench output container. Only content items accessed from the workbench resource tree run in the target container.

Another way to provide access to content in your repository from an assembled page is by using the Explorer widget. You can use the Explorer widget to provide access to the Workspaces area of the Hub or Home Page. This allows anyone accessing the page, or a portal containing the page, to open and run content items and create new content, depending on their privileges. To add the Explorer Widget to a page, select the *Container* tab on the sidebar and drag it onto the canvas. When you run a page with the Explorer widget, you can use it to navigate through workspaces, folders, and content items from the Resources tree and Content area, and create new content using options on the Action bar. When an administrator uses the Redirect option in the Administration Console to enable a custom welcome page with the Explorer widget, you can provide an enhanced home page combining all of the options available from the Workspaces view with custom content of your choice.

By default, the Workbench template and Explorer widget fill the entire page in order to provide ample space to display any content items and content creation options. As a best practice, leave the Workbench template containers or Explorer widget as the only items on a page, and add that page to a portal with other pages whose contents you want to display.

Note that if you created pages with the Explorer widget in a prior release, some Hub or Home Page features may not be available. Open the page in Db2 Web Query Designer, and replace the old Explorer widget with the new one to use the updated version.

Procedure: How to Use the Explorer Widget

You can add the Explorer widget to a page assembled from existing content.

1. On the Db2 Web Query Hub or Db2 Web Query Home Page, click the plus menu and then click *Assemble Visualizations*.

Db2 Web Query Designer opens.

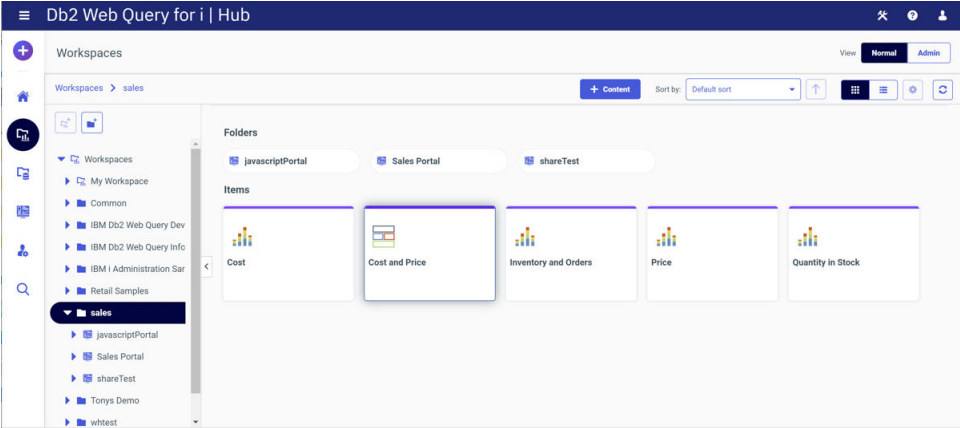
2. In the Choose Template dialog box, select the *Blank* template.
3. From the *Container* tab on the sidebar, drag the *Explorer* widget onto the canvas.

The widget displays on the page and incorporates the full display of the Workspaces view of the Hub or Home Page.

4. Click the maximize button on the Explorer widget container to fill your page.
5. Optimize the look of the page by removing the page and container titles and toolbars. Select the Explorer widget container and, on the *Settings* tab, clear the *Show title* and *Show toolbar* check boxes.

Optionally, select the entire page and clear the *Show page heading* and *Show page toolbar* check boxes to show only the Explorer widget on the page.

An example of a completed page is shown in the following image.

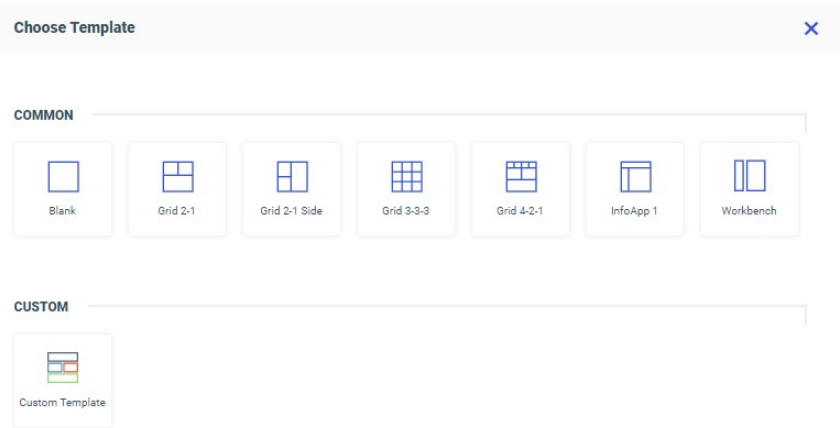


6. Save your page.

The page can now be added to a portal as described in *Defining a Portal Structure* or used as a standalone item.

Creating Custom Templates

When you assemble a page from existing content in Db2 Web Query Designer, the first step is to select a page template. Db2 Web Query Designer provides a selection of preset templates, as shown in the following image.



Select the Blank template to start with a blank canvas to which you can add, reposition, and resize containers on your own. The Grid 2-1, Grid 2-1 Side, Grid 3-3-3, and Grid 4-2-1 templates provide a set of containers that are prearranged on the page. The InfoApp 1 template provides a collapsible filter panel and movable filter grid to which you can add filter controls, which are generated when you add parameterized content to the page. The Workbench template provides a resource tree that you can use to navigate to content items in your repository, which you can then run in the target container on the right.

In addition to the default templates that Db2 Web Query Designer offers, you can create custom templates that reflect your unique layout. They are a great way to add structural variation or highly customized layouts to your page. Custom templates are also useful for organizations that have a set style for all of their pages. You must have administrator access to Db2 Web Query to deploy custom templates.

Procedure: How to Create a Custom Template for Db2 Web Query Designer Pages

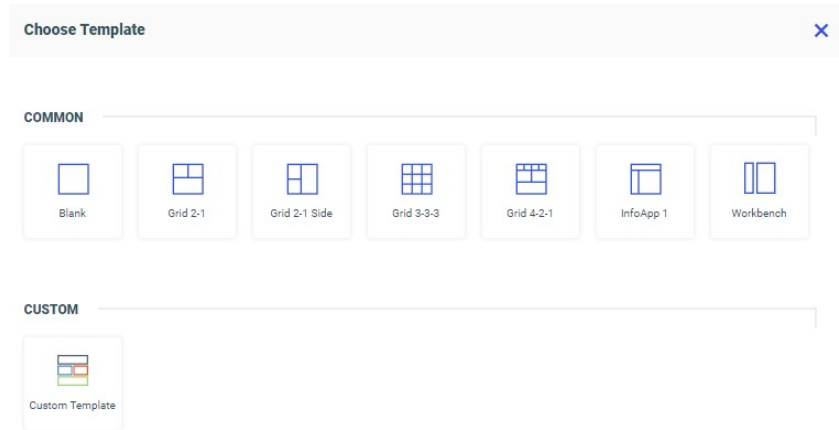
1. Create a new page. On the Db2 Web Query Hub or the Db2 Web Query Home Page, click the plus button.
2. Click *Assemble Visualizations*.

Db2 Web Query Designer opens, and you are prompted to select a template.
3. Select one of the default templates that you want to use as the starting point for your custom template, to click *Blank* to build a new template from scratch.
4. Modify the page layout to reflect your preferences. For example, insert new sections, customize the sizing of each row in the template.
5. Save your new page to any folder and exit Db2 Web Query Designer.
6. Optionally, apply a custom thumbnail to your template page. If you do not assign a custom thumbnail, the default thumbnail Db2 Web Query associates with pages will display.
7. In the Workspace area of the Hub or Home Page, expand the *Global Resources* folder, and then expand the *Page Templates* folder.

Two folders display inside the Page Templates folder: Standard and Custom.

8. Copy your new template page from its initial directory and paste it inside the *Custom* folder.

The next time you launch Db2 Web Query Designer using the Assemble Visualizations option, your custom template appears, as shown in the following image.



Once a custom template is created, you can configure access to it by using security rules to hide it from specific groups of users.

Passing Parameter Values to a Page

When a page includes content with dynamic parameter filters, you can pass parameter values to it from external content to set the filter values when the page initially loads. For example, you can create a chart or report with drill-down links that points to the page, and configure the drilldowns in such a way that the value the user clicks in the chart or report is used as a filter value in the page. This allows users to navigate between content items and determine the scope of the information that they want to see.

If the page is the target of a link tile in another page, parameter values in the source page are passed to the target page automatically if the Attach all parameters option is enabled and the same parameter filters are present. This allows users to see the same set of values in the target page without having to set filter values again.

Parameter values can also be passed to a page using parameter name-value pairs in the run-time URL. The run-time URL uses the following format:

```
http[s]://hostname:12331/webquery/run/path_to_item[?
&variable_name1=variable_value1&variable_name2=variable_value2]
```

where:

hostname

Is the name of the system where Db2 Web Query is installed.

path_to_item

Is the IBFS path of a page in your Db2 Web Query repository. You can find the path of a page by right-clicking it on the Db2 Web Query Hub or Db2 Web Query Home Page and clicking *Properties*. The file path is listed as the Path property.

Slashes (/) between folder names should be retained literally instead of encoded in the URL, and the colon (:) should be removed after *IBFS*. For example:

```
http[s]://hostname:12331/webquery/run/IBFS/WFC/Repository/My_Workspace/  
~user/example_page
```

&variable_name1=variable_value1&variable_name2=variable_value2

Are one or more parameter name-value pairs. To pass multiple values to a multiselect parameter in the page, separately list the same parameter name multiple times with each distinct value. For example, the URL,

```
http[s]://hostname:12331/webquery/run/IBFS/IBFS/WFC/Repository/  
My_Workspace/~user/example_page?&TIME_YEAR=2018&TIME_YEAR=2019
```

passes 2018 and 2019 to the *&TIME_YEAR* parameter.

Note: If multiple filters in a page use the same ampers variable name, for example, if they come from different data sources, the value or values defined in the URL are passed to each of these filters.

Procedure: How to Pass Parameter Values to a Page Using a URL Drilldown

You can drill down from a chart or report procedure to a page created in Db2 Web Query Designer. Drill-down parameters can be passed to the page to apply their values to any page filters.

1. Create and save a page that includes prompted filters. These filters can be created by adding parameterized external content to an assembled page and clicking *Add all filters to page* in the Filters tab of the sidebar, or can be created in a page with new content by dragging a field to the Filter toolbar.

The filters in this page will be populated with values passed from a drill-down report.

For information on creating a page, see [Creating Pages](#) on page 6.

2. Create a new report in InfoAssist.

On the Hub or Home Page, in the Workspaces view, click the *INFOASSIST* tab on the Action bar and click *Report* or *Chart*.

InfoAssist opens and you are prompted to select a data source.

3. Select the same data source that was used to filter the page, and click *Open*.

The report or chart canvas appears.

4. Create content that is sorted by the same fields that are used as filters in the page. For example, if the page is filtered by Customer Business Region and Sale Year, create a report with those fields in the row bucket. Add other measures and dimensions to your content based on your needs and preferences.
5. Select an element of the chart or report to which to add drill-down links.

If you are passing multiple parameter values through the drilldown, selecting a section of the chart or report that represents multiple sort values may help clarify to the user which values will be passed.

For example, in the report shown in the following image, the Sale Year field is the secondary sort field while Customer Business Region is the primary sort field, so each row in the Sale Year column represents a sale year and a business region.

Customer Business Region	Sale Year	Quantity Sold	Revenue
EMEA	2014	115,367	\$32,608,030.34
	2015	132,574	\$39,630,841.14
	2016	193,158	\$57,591,488.69
	2017	255,468	\$76,647,406.85
	2018	464,797	\$140,440,022.62
	2019	699,498	\$215,398,344.26
North America	2014	46,386	\$13,060,621.95
	2015	61,246	\$18,287,269.34
	2016	92,483	\$27,473,407.52
	2017	147,038	\$43,907,888.55
	2018	426,176	\$128,662,736.78
	2019	694,386	\$212,755,358.69
Oceania	2015	56	\$14,736.30
	2016	165	\$48,743.31
	2017	374	\$104,323.54
	2018	1,321	\$424,709.91
	2019	6,481	\$1,970,040.66

In this case, adding the drill-down links to the Sale Year column would allow you to choose from all of the available Customer Business Region and Sale Year values, whereas adding the links to the Customer Business Region column would only allow you to pass parameter values for the first sale year in each business region group. As an alternative to creating drilldowns on the lower level sort field, you could ensure that all parameter values are passed from a report by using repeating sort values. On the *Format* tab, in the *Features* group, click *Repeat Sort Value*.

6. On the *Field* tab, in the *Links* group, click *Drill Down*.

The Drill Down dialog box opens.

7. Change the drilldown type to a URL drilldown by selecting the *Web Page* radio button.

8. In the URL text box, type the run-time URL for the target page, without the parameters, using the format described above.

The URL format should resemble the following:

`http[s]://hostname:12331/webquery/run/path_to_page`

You can get the path to the page by right-clicking it on the Hub or Home Page and clicking *Properties*. It displays as the Path property. When you add it to the run-time URL for the drilldown, remember to remove the colon (:) after *IBFS*.

9. In the Description text box, type the name of the drilldown. This name displays in the tooltip from your content.
10. In the Target drop-down menu, select the location where the target page should run. It can be in a new tab, the same window where the report was run, or a user-specified location.
11. In the Parameters section, click *Add Parameter* to create a parameter to pass to each filter on the target page that you wish to populate.
 - a. In the Name text box, type the name of the parameter. Typically, this matches the field name. You can check the parameter names used in an assembled page by clicking the *Info* button.
 - b. From the Type menu, choose whether the value to pass should be the field value associated with the user drill-down link selection, or a constant value.
 - c. If you selected *Field*, use the Value menu to select a field from your content. If you selected *Constant*, type a constant value to pass to the page.
12. Repeat step 11 for all parameters whose values you want to pass to the target page, then click *OK* to create the drilldown.


In a report, drill-down links are added for each value of the field where the drilldown was created. In a chart, you can access drill-down links in a tooltip at run-time.

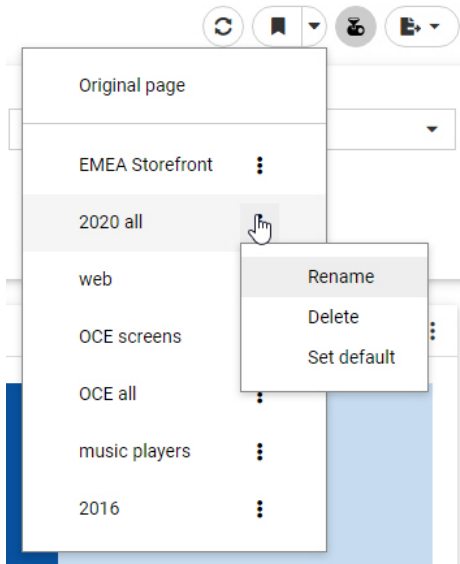
13. Run the chart or report.
14. At run-time, drill down to the target page.
 - ☐ If you created a chart, point to a section of the chart and click the drill-down link in the tooltip.
 - ☐ If you created a report, click a drill-down link. If that field of the report has multiple drilldowns associated with it, select the drilldown that you just created.
15. The page opens in a new browser tab or window. Notice that the value or values represented by your selection in the parent content item are applied as filter values in the page.

Bookmarking Control Selections in a Page

Bookmarking allows users to save filter control selections and other run-time content customizations that they make in a page so that they can easily reapply them when they run the page again later. This is especially useful when there are many filter controls on a page that a user may want to apply each time they access it. Bookmarks are personal to each user and can easily be created and deleted as needed.

Bookmarking is enabled automatically once the page has been saved, and is available in authored pages created with new content and in assembled pages built from existing content. To add a bookmark, make some filter selections when running a page, and then click the

Bookmarks button , which appears on the page toolbar at run time. You are prompted to type a name for the bookmark. Click *OK* to apply it. A maximum of 46 characters display for each bookmark name. Once applied, the bookmark is added to the list, which you can open by clicking the drop-down menu arrow next to the Bookmarks button. Select a bookmark from the list to refresh the page and apply the associated set of filter values. The bookmark list also includes the *Original page* option, which reloads the page with the default filter values selected. From the bookmark list, you can also use the menu for each existing bookmark to rename or delete the bookmark, or make it your personal default set of filter values when you load the page, as shown in the following image.



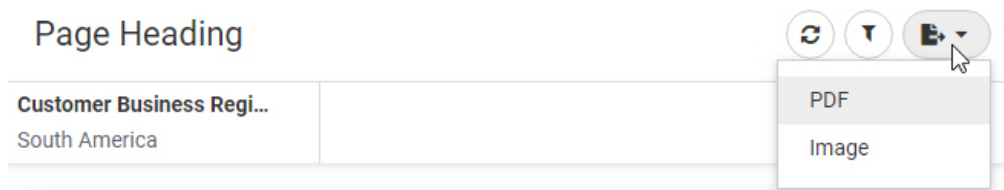
Each user can create a maximum of ten personal bookmarks for a page. If you reach this limit on a page, delete an existing bookmark to add a new one. The Original page bookmark is always available and cannot be deleted.

Bookmarks also save certain user customizations made in the page. For example, if you use on-chart filtering in an authored page at run-time to create a new filter, you can create a bookmark to save it. In authored pages, you can also change the filter condition from included values to excluded values and change the aggregation operation, and in assembled pages, you can change the content that displays in unlocked containers. All of these changes are saved when you create a new bookmark.

Exporting a Page as a PDF or Image

You can instantly download a snapshot of your page as a PDF document or .png image using the Export to file menu on the page toolbar. This allows you and anyone running the page to quickly capture its current state, with filter selections applied, in a format that can be easily reused, distributed, added to PowerPoint presentations, and more. The ability to capture a snapshot at design time allows you to quickly capture multiple visual representations of your data as you develop your content, while the ability to capture a snapshot at run time allows you or a user running the page to save images of it with the currently selected filter values.

When designing or running the page, the Export to file menu appears on the page toolbar by default. Click it to display options to download a snapshot of the page as a PDF or image, as shown in the following image.



The .pdf or .png file is downloaded using your browser. The file can be opened immediately and saved to distribute and reuse later.

The snapshot that is produced is a run-time representation of your content with styling and filtering, including on-chart filtering, applied. Since the exported file is an image, it is not interactive, so tooltips, scrolling, and other behavior are unavailable.

You can change the export options that are visible on the page toolbar. Select the entire page, then open the Settings tab. The *Show export* option and *PDF* and *Image* options are selected by default. Clear the *PDF* or *Image* check box to change the format available when exporting the page. If only one format is enabled, the Export to file option on the page toolbar changes from a menu to a button. You can also clear the *Show export* check box to remove the export option from the page.

The following considerations may affect page export behavior:

- ☐ If a container is maximized when you export the page, then only that container displays in the exported image or PDF file.
- ☐ Content in containers on a page may not appear correctly when exported from Internet Explorer.
- ☐ When exporting a page from Microsoft Edge, ensure that the containers on the page are large enough to display their contents without scroll bars. Otherwise, the content may overlap.
- ☐ XML files and procedures with an XML output format added to a page produce an error when exporting.
- ☐ Certain chart extensions, for example, the Sparkline KPI, KPI with Sparkline Large, and KPI Table extensions, may not render correctly in an exported page.
- ☐ If a container on a page contains a URL that does not have the same protocol, host name, and port number as that of the environment from which the page is run, the export cannot be performed due to same-origin policy restrictions.
- ☐ If a page has too many rows to fit onto a single page of a PDF document, then containers and their contents may be split by page breaks in the exported file.
- ☐ PDF documents in containers on a page, including charts and reports with a PDF output format, do not display when the page is exported.

Legal and Third-Party Notices

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, FOCUS, iWay, Omni-Gen, Omni-HealthData, and WebFOCUS are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle Corporation and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2021. TIBCO Software Inc. All Rights Reserved.