Infor Birst Visualization Extensibility Library

V 1.0

Important Notices

Trademark Acknowledgements

Publication Information

Release: Infor Birst Visualization Extensibility Library V 1.0

Publication date: December 27, 2021

Contents

Overview

Infor Birst – Extending Insights

Infor Birst enables you to bring your data to life and broaden your insights by embedding customized HTML and JavaScript within your dashboards. The HTML can reference an external CSS file or use internal styling to personalize the visualization. Third-party libraries such as Google Visualization API or D3.js can be used to further extend and personalize the visualization experience.

Requirements

The intent of this guide is not to provide instruction on HTML or JavaScript, but to provide information and guidance on how to leverage the HTML editor within the Infor Birst product. The guide provides examples of how D3 visualizations from third-party libraries can be leveraged and integrated with your Infor Birst environment and BQL queries. The examples have been built against an internal Infor Birst space that is not accessible for public consumption. You need to type or copy and paste the provided HTML and modify it accordingly to accommodate your own Infor Birst environment and BQL.

Tips

- It is recommended that you have a basic knowledge of HTML and JavaScript.

  For more information, see this free HTML tutorial: https://www.w3schools.com/

  Note: Infor is not responsible for the content and instruction provided by this site.
- Knowledge of Infor Birst and an understanding of BQL is a must.
- The URL of the third-party code must be publicly accessible. If you do not see the website in the preview, the site may be configured to not allow embedding on other websites.
- Any URLs in the code must use HTTPS because Birst is hosted over HTTPS.
- Because the HTML is already inside Birst, a login token is not required in the code.
- Although you can define CSS directly inside the HTML Editor, the custom style must only affect the external visualization and not conflict with the overarching Dashboards 2.0 theme.
- If the JavaScript is lengthy, consider providing some of it in a JavaScript library and then include a JavaScript src source reference.
- You must have Edit permissions in the dashboard to access the HTML dashlet.
- See the Infor Birst Help documentation for more information: https://login.bws.birst.com/Help/Full/data/integration/embedding_api_for_javascript.htm

Installation

Getting Started

To successfully implement the example, you must have access to an Infor Birst environment and adequate permissions that enables access to the Edit functionality within a dashboard.

The example shows how you can integrate visualizations from third-party D3 libraries into your Infor Birst environment. To modify the example, you need to have the following information readily available:

- The Infor Birst environment — Mandatory
- Measure(s) — Mandatory
- Attribute(s) — Mandatory
- Filter(s) — Optional
- Sort(s) — Optional

Infor Birst	Description	Example
Environment	URL that launches the Infor Birst environment.	https://xxxxxx.xxxx.birst.com
Measures	A data point that contains numeric or quantitative values. In analytics it is used to do comparative analysis against performance and benchmarks.	[OrderDate: Revenue] [OrderDate: Sum: Revenue] [OrderDate: Avg: Revenue]
Attributes	In data analysis, an attribute is a characteristic or feature that is measured for each record. Often answer who, what, where, when, and why.	[Time.Year/Month] [Products.ProductName] [Sales.RepID]
Filters	Views used to segment data into smaller groups.	[Time.Year/Month] = ‘2021/06’

Infor Birst	Description	Example
Sorts	Control the sort order of the information for more impactful visualizations.	Attribute: [Products.ProductName] Measure: OrderDate: Revenue]

Note: The BQL code generated by Infor Birst provides you with some of the information in the table.

Infor Birst Environment

This script is contained within the example and allows for communication with the Infor Birst environment and the D3 visualization library.

This information can most often be found in your browser. Contact your company Infor Birst administrator for guidance.

```
<!-- Comment:  Set Infor Birst Environment -->
<script type="text/javascript"
src=https://app2102.bws.birst.com/js/birst_embed.js>
</script>
```

Understanding BQL

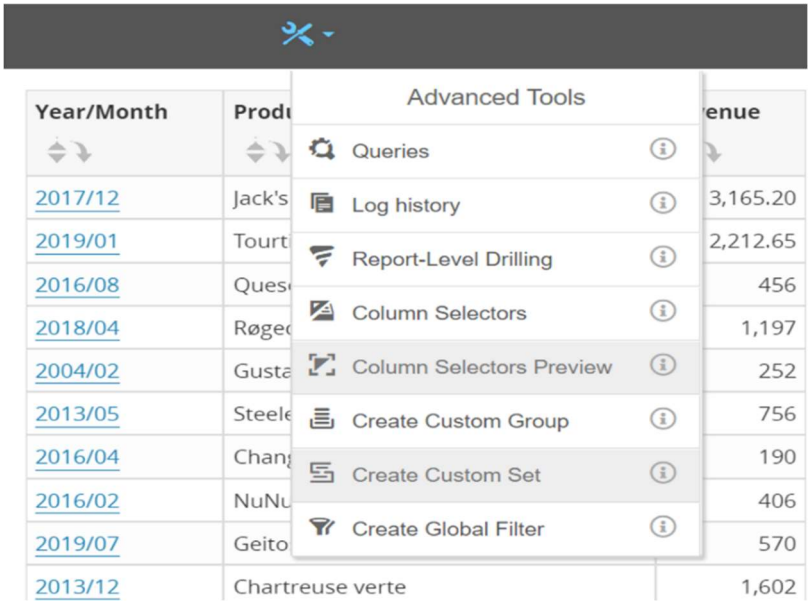
BQL is a powerful proprietary logical query language that supports important capabilities within the Infor Birst platform. Although it has a lot of similarities to SQL code, it provides extended functionality to allow for more control of the query engine and result set.

Because Visualizer, Designer, and Dashboards 2.0 automate most of the BQL, this is the best place to evaluate the BQL Code that is being generated. The code provides you with measures and attributes that are needed to populate the D3 visualization.

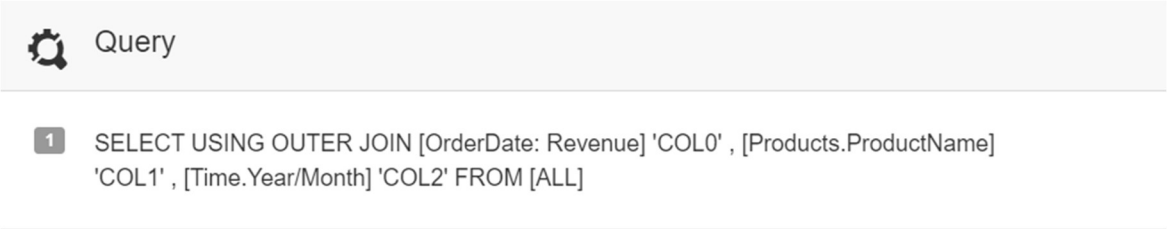
Step 1: Accessing the BQL Query - Visualizer

1. Launch Visualizer.
2. Build a query or use an existing one.
    - Main menu
    - Within dashboard

3. From the **Advanced Tools** menu, select **Queries**.



The Query View is displayed.



You can copy and paste from this logical query to use in the HTML example.

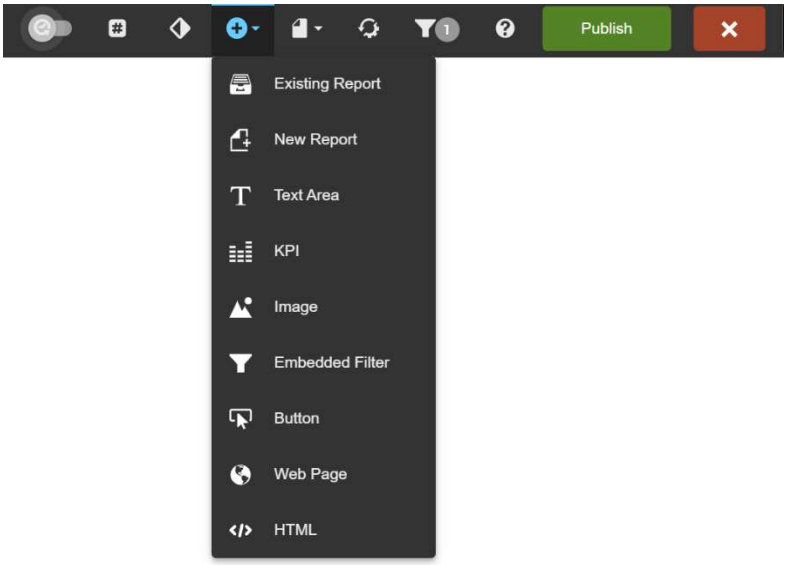
HTML Dashlet

A dashlet is an individual container that can be added to or removed from an Infor Birst dashboard. Dashlets have additional controls that can be controlled to personalize the experience for greater insights. To embed customized HTML and JavaScript we will be using the HTML dashlet.

Step 1: Accessing the HTML dashlet

1. Within a dashboard, click **Edit**.

2. From the **Add** menu, select **</> HTML**.



The HTML Editor is displayed. The HTML Editor allows you to specify HTML content and preview the results immediately.

3. In the HTML Editor, type or paste the HTML code. Click **Insert** to place the dashlet on the dashboard.

Example script:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: #EAEDED;}
h1   {color: #900C3F; font-family: verdana; font-size: 150%;}
p1   {color: #636466; font-family: verdana; font-size:  95%;}
</style>
</head>
<body>
<img src="https://www.infor.com/img/logo-infor.png" alt="Infor Logo"
width="50" height="50">
<h1>Infor Birst Embedded HTML </h1>
<p> Enter HTML to display in a dashboard
</p>
</body>
</html>
```

Example script with comments:

```
<!-- Comment:  !DOCTYPE html - Document Type
       Defined once  -->
<!DOCTYPE html>
<!-- Comment:  HTML - Root of the HTML document
               Start:  <html>   End:  </html>  -->
<html>
<!-- Comment:  Head - Container for metadata (data about data) and is placed
between the <html> tag and the <body> tag
```

```
               Metadata is not displayed
               Start:  <head>  End: </head> -->
<head>
<!-- Comment:  Style - Define style information (CSS) for a document
                Start:  <style>  End: </style> -->
<style>
body {background-color: #EAEDED;}
h1   {color: #900C3F; font-family: verdana; font-size: 150%;}
p1   {color: #636466; font-family: verdana; font-size:  95%;}
</style>
</head>
<!-- Comment:  Body tag - Defines the documents body
                Body element - contains all the contents of an HTML document
       Examples:  img = images
                  Requires 2 attributes:
                  src - Specifies the path to the image
                  alt - Specifies an alternate text for the image
                  to display; if the image can't be displayed,
                  h1 = header,
                  p = paragraph
                        Only 1 <body> element in an HTML document
                        Start:  <body>  End: </body> -->
<body>
<img src="https://www.infor.com/img/logo-infor.png" alt="Infor Logo"
width="50" height="50">
<h1>Infor Birst Embedded HTML </h1>
<p> Enter HTML to display in a dashboard
</p>
</body>
</html>
```

Step 2: Embedding D3 Visualization

This example uses the Dashboards 2.0 JavaScript API functionality to display a Decomposition Tree from a Birst query.

A Decomposition Tree allows the user to visualize data across multiple dimensions. It aggregates the data in the format of Parent-Child relationship and enables drilling down into the dimensions. The drill-down option on certain criteria makes it a valuable tool for performing root cause analysis.

The visualization requires two types of inputs:

- Measure: The metric you would like to analyze. This is a Measure in a BQL Query.
- Dimensions: These dimensions are setup in a Parent-Child relationship format. This is an Attribute in a BQL Query.

Accessing the example

1. Download the example from the product page.

2. Instructions for where to embed information that relates to your environment and query are contained within the example. The team has added variables to enable more ease in integration.

Understanding the HTML that is provided

The HTML contains various sections denoted by Comment tags. These tags provide documentation to enable easy integration of the sample HTML into your environment to achieve the desired results. Each section can be easily found by typing COMMENT: ** in the search field of your editor.

Note: Not all sections are the same for each HTML provided, but a similar format has been followed.

We will review each section in detail:

- Comment: ** SETTINGS
- Comment: ** COLORS
- Comment: ** LIBRARY
- Comment: ** BQL
- Comment: ** FILTER
- Comment: ** SORT
- Comment: ** DATASET
- Comment: ** SIZING
- Comment: ** TOOLTIPS
- Comment: ** ANIMATION

Comment: ** SETTINGS

This section defines the initial settings for the HTML such as the measures and attributes from the BQL as well as allows control of the labels that are used in the D3 visualization. Variables are used to allow for ease of integration.

```
// Comment: ** SETTINGS

        // Comment: Please enter the 'Attributes' and 'Measures' below. The
recommended method is to copy the BQL query to Notepad from the Visualizer by
creating a simple table which contains all the required 'Attributes' and
'Measures' to be used. The BQL query will be helpful to derive the 'Measure'
and 'Attribute' names for the inputs.
        // Comment: Please enter the 'Measure' from the BQL query. Also, enter
the 'Measure' name to be displayed in the visual.
        // Comment: Example: var Measure_1="[OrderDate: Revenue], " where
'Revenue' is the 'Measure' by 'OrderDate'. var MeasureName_1="Revenue" is the
title of the 'Measure' that will be displayed in the visual.
```

```
        var Measure_1="[OrderDate: Revenue], "
        var MeasureName_1="Revenue"
        var Measure_2="[OrderDate: Sales Cost], "
        var MeasureName_2="Cost of Sales"
        // Comment: Dimensions (Parent 'Attribute')
        // Comment: Please enter the 'Attribute' in the format [Table.Column
Name]. Also, enter the column name to be displayed in the chart.
        // Comment: Example: Parent = "[Customers.SalesRegion]" where
"Customers" is the table name and "SalesRegion" is the column name.
ParentName = "Sales Region" will be displayed as the title in the chart.
        var Parent="[Customers.SalesRegion]"
        var ParentName = "Sales Region"
        // Comment: Dimensions (Child 'Attribute')
        // Comment: Please enter the 'Attribute' in the format [Table.Column
Name]. Also, enter the Column Name to be displayed in the chart.
        // Comment: Example: Child_1="[Customers.Country]" where "Customers"
is the table name and "Country" is the column name. Child_1Name="Country"
will be displayed as the title in the chart.
        var Child_1="[Customers.Country]"
        var Child_1Name="Country"
        var Child_2="[Customers.City]"
        var Child_2Name="City"
        var Child_3="[Customers.CompanyName]"
        var Child_3Name="Company Name"
        var Child_4=""
        var Child_4Name=""  …
```

Comment: ** COLORS

This section defines the colors that are used for the visualization. Color plays a key role in the ability to quickly identify the meaning of the chart for better insights.

```
// Comment: ** COLORS
        // Please enter the Color Palette below if needed or leave as
'default'. You can enter the hexadecimal code or the name of the color.
        var Color_1="#0054B1"
        var Color_2="#82D4D4"
        var Color_3="#7928E1"
        var Color_4="#97979B"
        var Color_5="#FCC888"
              var Color_6=”#DF6F00” …
        var Color_6="#DF6F00"  …
```

Comment: ** LIBRARY

This section defines the libraries using links that are required to integrate the Infor Birst environment with the third-party libraries. Below in bold is the link to call the Infor Birst environment and the java script api as well as the D3 libraries.

```
<!-- Comment: ** LIBRARY

    <!-- Comment: load BQL library -->
    <script type="text/javascript"
src=https://app2102.bws.birst.com/js/birst_embed.js></script>
```

```
    <!-- Comment: load the d3.js library -->
    <script src=https://d3js.org/d3.v5.min.js></script>
    <!-- Comment: load underscore.js library for discovering the depth of the
nodes, used in placement of the labels. -->
    <script src=https://cdn.jsdelivr.net/npm/underscore@1.13.1/underscore-
umd-min.js></script>
```

Comment: ** BQL

This section defines the BQL query that is submitted to the Infor Birst query engine. The results are exposed to the D3 visualization. Variables are used for more seamless integration.

```
// Comment: ** BQL

// Comment: BQL queries stored in 'queries' variable --> Birst Query.
        var queries = [{
                "name": MeasureName_1,
                "query": "SELECT USING OUTER JOIN "+ Measure_1 +query_columns+
Parent +" FROM [ALL]"
                }, …
```

Comment: ** FILTER

This section defines the handling of the integration between the dashboard filters and the result set that is exposed to the D3 visualization.

```
// Comment: ** FILTER

        if(e.data.operation === "setFilters") {

        // Comment: Retrieve filter if present, and run the filtered query.
                        var pagefilter = e.data.filters[0].value[0];

// Comment: console.log(e.data.filters[0].value[0]);

        // Comment: Adding the WHERE clause as per Filters from the dashboard
        var query_clause = ` WHERE ( ( [Time.Year]='${pagefilter}' ) )`

        query  = queries[0]["query"] + query_clause;

        // Comment:Add buttons for each query/filter
                        generate_buttons(query_clause)

                        BirstConfig.getData(query)
                }

                if(e.data.operation === "executeQueryResult") {

        // Comment: Delete the tree if it already exists.
                        try {
```

```
        document.getElementById("tree_graph").remove()
                        } catch {}

// Comment: console.log(query); …
```

Comment: ** SORT

This section defines the attribute or measure that will be sorted. The order is also defined.

```
// Comment: ** SORT

// Comment: Setting the sorting of the data. The sorting is according to the
'Measure_Display' variable
        var Sort_Display = Metric_Layer

        // Comment: BQL Query for extraction of the data
        var BQL = "SELECT USING OUTER JOIN " + Metric_Layer + ", " +
Vertical_Axis + ", " + Metric_Measure + " FROM [ALL]";
        //
        var BQL_append = ")  ORDER BY "+ Sort_Display +" DESC,
"+Vertical_Axis+"ASC" ; …
```

Comment: ** DATASET

This section defines the transformation of the data results to be more optimal for the D3 visualization.

```
// Comment: ** DATASET

        // Comment: Process data retrieved from the database
        var columnheaders = e.data.result.columnNames.reverse(); // Comment:
Column names from the query.

// Comment: console.log(columnheaders)

        var dataset = e.data.result.rows // Results (data) from the Query.

// Comment: console.log(dataset)

        // Comment: Running the 'prepare_tree' function.
                prepare_tree(treeData, dataset);

        // Comment: GENERATE GRAPH

        // Comment: Set the dimensions and margins of the diagram
                var margin = {
                                top: 30,
                                right: 90,
                                bottom: 30,
```

```
                                left: 90
                }, …
```

Comment: ** SIZING

This section defines the sizing for the visualization, attributes, etc. Use this section to accommodate the proper device resolution and dashboard real estate.

```
// Comment: ** SIZING
        // Comment: Set the width and the height of the visual.

        width = window.innerWidth - 100 - margin.left - margin.right,
        height = window.innerHeight - 100 - margin.top - margin.bottom;

        // Comment: Append the svg object to the body of the page.
        // Comment: Append a 'group' element to 'svg'.
        // Comment: Move the 'group' element to the top left margin.
        var svg = d3.select("body").append("svg")
                .attr("width", width + margin.right + margin.left)
                .attr("height", height + margin.top + margin.bottom)
                .attr("id", "tree_graph")
                .append("g")
                .attr("transform", "translate(" +
                        margin.left + "," + margin.top + ")"); …
```

Comment: ** TOOLTIPS

This section defines tooltips that are passed to the visualization.

```
// Comment: ** TOOLTIPS
                nodeEnter.append("svg:title")
                        .text(function(d) {
                                return d.data.name
                        });

        // Comment: Add labels for the nodes
                nodeEnter.append('text')
                        .attr("dy", "-.35em")
                        .attr("x", function(d) {
                                return 4;
                        })
                        .attr("text-anchor", function(d) {
                                return "start";
                        })
                        .text(function(d) {
                                var name = d.data.name
                                // Slice the name if needed
                                if(name.length >= 27) {
                                        return name.slice(0, 25) + "..."
                                };
                                return name; …
```

Comment: ** ANIMATION

This section defines any animation settings that may be required or optional within the visualization.

```
// Comment: ** ANIMATION
                var nodeUpdate = nodeEnter.merge(node);
        // Comment: Transition to the proper position for the node
                nodeUpdate.transition()
                        .duration(duration)
                        .attr("transform", function(d) {
                                return "translate(" + d.y + "," + d.x + ")";
                                        }); …
```

Limitations

To be successful in extending your insights and leveraging embedded HTML and third-party libraries, knowledge of Infor Birst and HTML and the various components such as CSS and JavaScript is a must.