Widget Examples

- Overview
- Dashboard Widget
- Sub Tab Widget
- Text Widget
- Export Button Widget
- Filter Widget
    - Dates
    - Buttons
- Report Widget
    - Chart
    - Table
- Modal Window Widget
- Custom HTML Widget
    - Text & Report Data

Overview

The following sections outline sample dashboard content developed with DashXML and Yellowfin content. It will explore each of the avilalbe widget types, how they were set up in the XML, and how they were styled through custom CSS.

Note: this is merely a guide in order to walk you through using the DashXML framework for your own dashboard requirements.

The main XML widget types consist of the following:

```
<application-definition>
        <sourcefilter></sourcefilter>
        <dashboard>
                <text></text>
                <exportlist>
                        <export><export>
                </exportlist>
                <filter>
                        <options>
                                <summary><summary>
                        </options>
                </filter>
                <report>
                        <modal>
                                <subcontent></subcontent>
                        </modal>
                        <filterMap></filterMap>
                </report>
                <modal>
                        <subcontent></subcontent>
                </modal>
                <customhtml>
                        <replacements></replacements>
                </customhtml>
                <subtab>
                        <options>
                                <subcontent></subcontent>
                                <summary></summary>
                        </options>
                <subtab>
        </dashboard>
</application-definition>
```

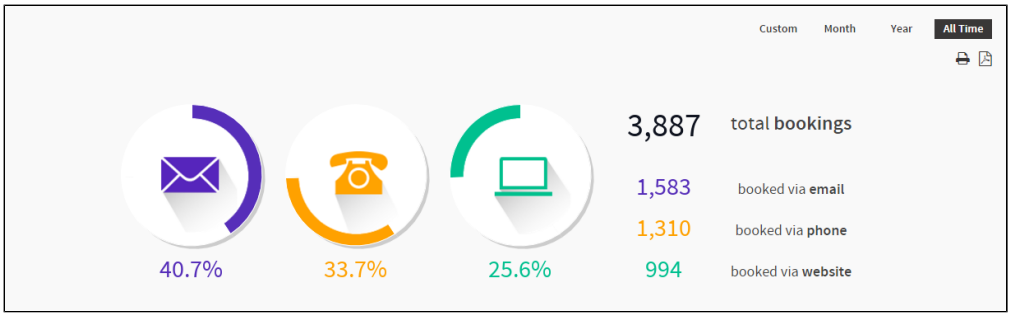
> *The full XML structure is outlined on the Widget XML page.*

Dashboard Widget

top

Dashboard Widgets are used to define the contents of each page in order to produce a dashboard. The dashboard will hold all other widgets to be displayed, called, or used on a page. It can contain as many reports as required, each styled be its own CSS class, as well as allowing for styles to be applied to the dashboard as a whole.

Example Output



Example Elements

`<dashboard>`

Child	Value	Description
`id`	1	This is a unique number used to identify the tab within the XML. It's also used in the URL in order to identify the tab. For example, if the id is 1 then the URL might look like: `http://localhost:7171/DashboardPortal/dashboard.jsp?id=1`
`filter`		This contains the details of filters used on the tab. In this scenario we had date filters controlling the dashboard content. See Dates for more detail.
`exportlist`		This contains the details of any export buttons to be used on the dashboard or individual reports. See Export Button Widget for more detail
`report`		This contains information around a single report to be displayed on the tab. See Chart for more detail.

Example XML

```
<application-definition>
        <dashboard>
        <id>1</id>

                <!-- Year Filters -->
        <filter>
            <sync>true</sync>
            <datatype>DATE</datatype>
            <name>dateFilter</name>
            <operator>BETWEEN</operator>
            <styleClass>topFilter</styleClass>
                        <options>
                                <title>All Time</title>
                                <value>1900-01-01\|2100-01-01</value>
                                <defaultOption>true</defaultOption>
                        </options>
                        <options>
                                <title>Year</title>
                                <typeCode>PREDEF</typeCode>
                                <value>YEARTODATE</value>
                        </options>
                        <options>
                                <title>Month</title>
                                <typeCode>PREDEF</typeCode>
                                <value>MONTHTODATE</value>
                        </options>
                        <options>
                                <title>Custom</title>
                                <prompt>true</prompt>
                        </options>
        </filter>

                <!-- Export Buttons (Print, PDF, XLS) -->
                <exportlist>
                        <export>
                                <styleClass>printExport</styleClass>
                                <exporttype>PRINT</exporttype>
                        </export>
                        <export>
                                <styleClass>pdfExport</styleClass>
                                <exporttype>PDF</exporttype>
                        </export>
                </exportlist>

                <!-- Canvas Chart -->
                <report>
                        <display>CHART</display>
                        <uuid>898e21f9-5acd-4b03-943c-25feca2c23c2</uuid>
                        <styleClass>canvasChart</styleClass>
                        <filterMap>
                                <filterUUID>071367c9-9cad-46c6-94d8-f4821c61d90e</filterUUID>
                                <mapToFilter>dateFilter</mapToFilter>
                        </filterMap>
                </report>

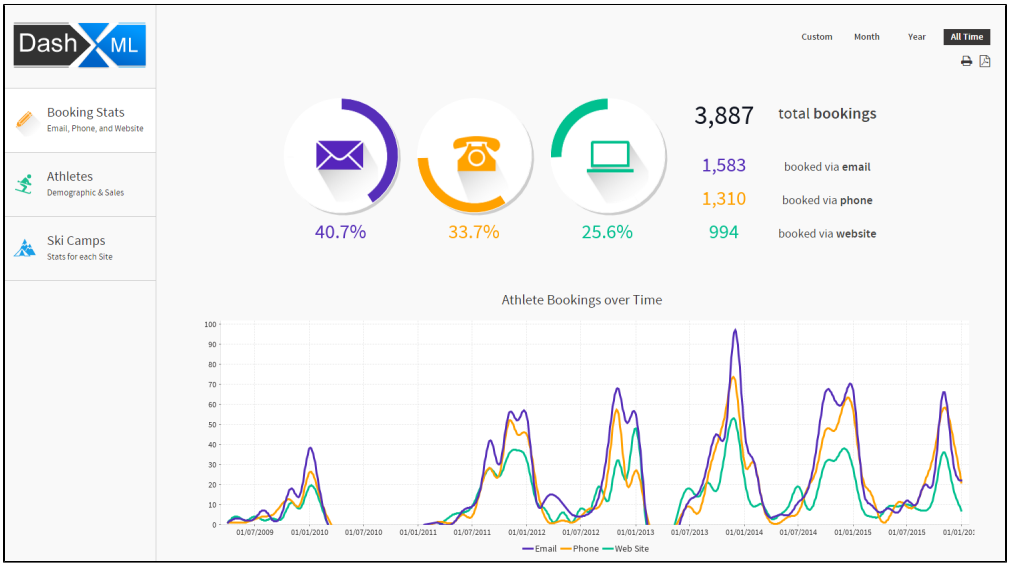
        </dashboard>
        <!-- End of the Example -->
</application-definition>
```

Sub Tab Widget

top

Sub Tab Widgets are used for toggling between different sections of content, within a single dashboard definition - much like how sub tabs work within the standard Yellowfin interface. A button or drop down list allows the user to switch between tabs to display different content on the screen.

Example Output



Example Elements

<subtab>

Child	Value	Description
`styleClass`		Contains a CSS class name used to apply styling to the component.
`name`		This is a name of the sub tab for referencing within the XML.
`options`		This contains nodes to define the contents of the display and reports within the sub tab. See `<options>` node description.

<options>

For within a sub tab.

Child	Value	Description
`title`	Booking Stats	This is the text label for the sub tab selector (how you toggle between the sub tab sets of reports).
`description`	Email, Phone, and Website	This is the descriptive text displayed below the sub tab title in the sub tab selector.
`icon`	<![CDATA[<div class="subtabBookingIcon"></div>]]>	This contains a reference to include an icon within the sub tab button. For example: `<![CDATA[<div class="subtabBookingIcon"></div>]]>`

Example XML

```
<!-- Start of Sub Tabs -->
<subtab>
        <styleClass>subTabs</styleClass>
        <name>Ski Team</name>

        <!-- Booking Stats Tab -->
        <options>
                <prompt>false</prompt>
                <title>Booking Stats</title>
                <description>Email, Phone, and Website</description>
                <icon><![CDATA[<div class="subtabBookingIcon"></div>]]></icon>
                <subcontent>
                        <styleClass>subTabOne</styleClass>
                        <name>Booking Stats</name>

                        <!-- Canvas Chart -->
                        <report>
                                <display>CHART</display>
                                <uuid>898e21f9-5acd-4b03-943c-25feca2c23c2</uuid>
                                <styleClass>canvasChart</styleClass>
                                <filterMap>
                                        <filterUUID>071367c9-9cad-46c6-94d8-f4821c61d90e</filterUUID>
                                        <mapToFilter>dateFilter</mapToFilter>
                                </filterMap>
                        </report>
                        <!-- Line Chart -->
                        <report>
                                <display>CHART</display>
                                <title>Athlete Bookings over Time</title>
                                <uuid>d1f779fb-7dbf-42e2-8e6a-bc7dc84ae04f</uuid>
                                <styleClass>fullWidthChart</styleClass>
                                <filterMap>
                                        <filterUUID>a0bfc7f6-1550-4f4d-a3c5-031b20c1a942</filterUUID>
                                        <mapToFilter>dateFilter</mapToFilter>
                                </filterMap>
                        </report>
                </subcontent>
        </options>

        <!-- Athletes Tab -->
        <options>
                <prompt>false</prompt>
                <title>Athletes</title>
                <description>Demographic &amp;amp; Sales</description>
                <icon><![CDATA[<div class="subtabAthletesIcon"></div>]]></icon>
                <subcontent>
                        <!-- Content Removed For Simplification -->

                </subcontent>
        </options>

        <!-- Ski Camps Tab -->
        <options>
                <prompt>false</prompt>
                <title>Ski Camps</title>
                <description>Stats for each Site</description>
                <icon><![CDATA[<div class="subtabSkiCampsIcon"></div>]]></icon>
                <subcontent>
                <!-- Content Removed For Simplification -->

                </subcontent>
        </options>
</subtab>
```

Example CSS

```
/* sub tabs */
.subTabs {
        clear: both;
```

```
}
/* Sub Tab Button Selectors */
.subTabSelection {
        width: 228px;
        display: table;
        border-collapse: separate;
        table-layout: fixed;
        overflow: hidden;
        position: fixed;
        left: 0px;
        top: 127px;
        border-spacing: 0px 1px;
        background-color: #CCCCCC;
}
.subTabSelection .subTabSelector {
        display: table-row;
        background: #F9F9F9;
        cursor: pointer;
        position: relative;
}
.subTabSelection .subTabMain {
        height: 55px;
        overflow: hidden;
        position: relative;
        padding: 20px 10px;
}
.subTabSelection  .selected {
        background: #FFFFFF;
}
.subTabSelection  .selected  div{
        color: #393737;
}
.subTabSelection .subTabTitle {
        color: #393737;
        font-size: 20px;
        font-family: sourceSansPro, sans-serif;
        padding-left: 55px;
        padding-top: 3px
}
.subTabSelection .subTabDesc{
        color: #393737;
        font-size: 13px;
        line-height: 22px;
        padding-left: 55px;
}
.subTabSelection .subTabSummary {
        color: #393737;
        font-size: 18px;
        font-weight: bold;
}
.subTabSelection .subTabIcon {
        position: absolute;
        top: 20px;
        left: 5px;
        width: 55px;
        text-align: center;
}
.subTabSelection .subtabBookingIcon{
        background-image: url("../images/booking.png");
        height: 55px;
        width: 55px;
}

.subTabSelection .subtabAthletesIcon{
        background-image: url("../images/athlete.png");
        height: 55px;
        width: 55px;
}

.subTabSelection .subtabSkiCampsIcon{
        background-image: url("../images/camp.png");
```

```
        height: 55px;
        width: 55px;
}
.subTabSelector.selected .subtabBookingIcon{
        background-image: url("../images/bookingSelected.png");
        height: 55px;
        width: 55px;
}

.subTabSelector.selected .subtabAthletesIcon{
        background-image: url("../images/athleteSelected.png");
        height: 55px;
        width: 55px;
}

.subTabSelector.selected .subtabSkiCampsIcon{
        background-image: url("../images/campSelected.png");
        height: 55px;
        width: 55px;
}
```

Text Widget

top
The Dashboard Title can be implemented using the `<text>` XML Widget. This will simply need to contain the text to be displayed as the title, and a css class name to be used for styling that text.

Example Output



Example Elements

`<text>`

Child	Value	Description
styleClass	dashboardTitle	Contains a CSS class name used to apply styling to the component.
text	Ski Team Dashboard	This contains text to be displayed on the dashboard as titles, descriptions, or for other uses.

Example XML

```
<!-- Dashboard Title Text -->
<text>
    <styleClass>dashboardTitle</styleClass>
    <text>Ski Team Dashboard</text>
</text>
```

Example CSS

```
.dashboardTitle {
    color: #393737;
    font-size: 40px;
    float: left;
}
```

Export Button Widget

top
The Export Buttons allow you to export the dashboard and reports to various formats using the `<exportlist>` XML Widget. This will need to define which export formats you wish to allow, and which report to export (if you are using XLS).

Example Output



Example Elements

`<exportlist>`

Child	Value	Description
`export`		This contains nodes to define the export buttons to be displayed on the dashboard tab. In this example, we've displayed the Print, PDF, and XLS buttons. See `<filterMap>` node description.

`<export>` (Print)

Child	Value	Description
`styleClass`	printExport	Contains a CSS class name used to apply styling to the component. In this particular example, we've used a unique class per export type, as each button has a different icon.
`exporttype`	PRINT	This specifies that the browser print function will be called to print the entire dashboard.

`<export>` (PDF)

Child	Value	Description
`styleClass`	pdfExport	Contains a CSS class name used to apply styling to the component. In this particular example, we've used a unique class per export type, as each button has a different icon.
`exporttype`	PDF	This specifies that the current dashboard will be exported to PDF format.

`<export>` (XLS)

Child	Value	Description
`styleClass`	xlsExport	Contains a CSS class name used to apply styling to the component. In this particular example, we've used a unique class per export type, as each button has a different icon.
`exporttype`	XLS	This specifies that a single report will be exported to XLS format.
`reportuuid`	39f8f49f-124b-4b36-a7f6-2f1ed9724c53	This is the UUID of the report to be exported to XLS.

Example XML

```
<!-- Export Buttons (Print, PDF, XLS) -->
<exportlist>
    <export>
        <styleClass>printExport</styleClass>
        <exporttype>PRINT</exporttype>
    </export>
    <export>
        <styleClass>pdfExport</styleClass>
        <exporttype>PDF</exporttype>
    </export>
    <export>
        <styleClass>xlsExport</styleClass>
        <exporttype>XLS</exporttype>
        <reportuuid>39f8f49f-124b-4b36-a7f6-2f1ed9724c53</reportuuid>
    </export>
</exportlist>
```

Example CSS

```
/* Export Buttons */
.exportList {
    display: table;
    padding: 0px 15px;
    margin: 20px 0px 20px 0px;
    font-family: FontAwesome;
    font-style: normal;
    font-weight: normal;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
    font-size: 18px;
    color: #364149;
}
.exportLink {
    padding-right: 10px;
    cursor: pointer;
    display: table-cell;
}
.printExport:before {
    content: "\f02f";
}
.pdfExport:before {
    content: "\f1c1";
}
.xlsExport:before {
    content: "\f1c3";
}
```

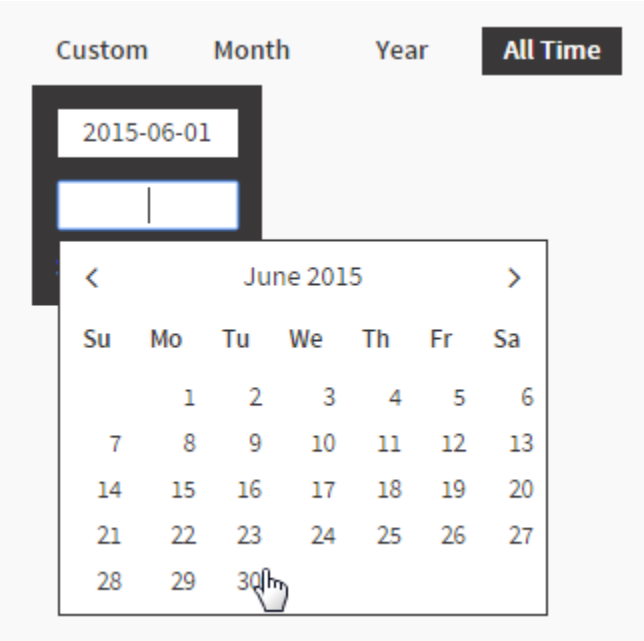
Filter Widget

top

Dates

These can be implemented using the `<filter>` XML widget. Each filter widget requires at least one option to be defined, allowing the user to select a value to apply. This particular filter has four options; Custom, Year, Month, and All Time. The Year, Month, and All Time filters have been set up using pre-defined date ranges, while the Custom option was specified as user prompt in order to allow users to specify a custom range.

Example Output



Example Elements

`<filter>`

Child	Value	Description
`sync`	true	Setting this to true allows the values of this filter to be passed to other dashboards.
`datatype`	DATE	This specifies that the filter options need to return a date type in order to be applied.
`name`	dateFilter	This contains the name of the filter, to be used to identify it for the `<filterMap>` node used on reports receiving the values.
`operator`	BETWEEN	This specifies the operator for the filter, and must be compatible with the operators used in reports you wish to pass the filter values to. BETWEEN means two values are expected. This option should be used for operators defined in the report including, but not limited to; Between, Not Between.
`styleClass`	topFilter	This contains a CSS class name used to apply styling to the component.
`options`		This contains nodes to define the contents of the display and possible values. Each option to be presented to the user will need to have its own <option> node. In this example we've used a Year, Month, and Custom option. See `<options>` node description.

`<options>` (Custom)

Child	Value	Description
`title`	Custom	This is the label displayed on the filter. A text name describing the value that will be applied.
`prompt`	true	This specifies that the option doesn't have a value predefined, but rather will ask the user to provide it. In this example, as we are using a prompt, there is no need for a `<value>` node. As the filter is defined as a date, the user will be presented with a calendar picker (which can be styled using css).

`<options>` (Year)

Child	Value	Description
`title`	Year	This is the label displayed on the filter. A text name describing the value that will be applied.

typecode	PREDEF	This defines that the option will use a predefined filter, such as a predefined date range. You will then have to specify which definition to use in the <value> node.
value	YEARTODATE	YEARTODATE uses the predefined date range that is the first date of the current year, to the current date.

<options> (Month)

Child	Value	Description
title	Month	This is the label displayed on the filter. A text name describing the value that will be applied.
typecode	PREDEF	This defines that the option will use a predefined filter, such as a predefined date range. You will then have to specify which definition to use in the <value> node.
value	MONTHTODATE	MONTHTODATE uses the predefined date range that is the first date of the current month, to the current date.

<options> (All Time)

Child	Value	Description
title	All Time	This is the label displayed on the filter. A text name describing the value that will be applied.
value	1900-01-01|2100-01-01	This defines that the date range will be from 1st Jan 1900 to 1st Jan 2100, effectively covering all the data in this dataset.
defaultOption	true	This specifies this option as the default selection for the dashboard. Only one <option> node can be the default, the rest should be false or omit this node.

Example XML

```
<!-- Year Filters -->
<filter>
        <sync>true</sync>
        <datatype>DATE</datatype>
        <name>dateFilter</name>
        <operator>BETWEEN</operator>
        <styleClass>topFilter</styleClass>
        <options>
                <title>All Time</title>
                <value>1900-01-01\|2100-01-01</value>
                <defaultOption>true</defaultOption>
        </options>
        <options>
                <title>Year</title>
                <typeCode>PREDEF</typeCode>
                <value>YEARTODATE</value>
        </options>
        <options>
                <title>Month</title>
                <typeCode>PREDEF</typeCode>
                <value>MONTHTODATE</value>
        </options>
        <options>
                <title>Custom</title>
                <prompt>true</prompt>
        </options>
</filter>
```

Example CSS

```
/* Date Filters */
.topFilter {
    height: 24px;
```

```
    line-height: 24px;
    padding-top: 14px;
}

.topFilter .filterOption {
    text-align: center;
    float: right;
    color: #393737;
    cursor: pointer;
    width: 70px;
    margin-left: 5px;
    font-weight: bold;
}

.topFilter .selected {
    background-color: #393737;
    color: #FFFFFF;
}

.topFilter .filterOption .customEntry {
    background-color : #393737;
    color: #FFFFFF;
    position: absolute;
    overflow: hidden;
    z-index: 100;
    margin-top: 5px;
    width: 115px;
}

.topFilter .filterOption .customEntry  input{
    background-color : #FFFFFF;
    border: 0px;
    font-family: sourceSansPro, sans-serif;
    font-size: 14px;
    color: #393737;
    height: 24px;
    width: 70px;
    line-height: 24px;
    margin: 12px 12px 0px 12px;
    padding: 0px 10px;
    text-align: center;
}


/* Date Entry */
.submitCustom {
    float: left;
    line-height: normal;
    margin: 10px 3px 10px 12px;
}

.cancelCustom {
    float: right;
    line-height: normal;
    margin: 10px 12px 10px 3px;
}

/* Date Picker */
.ui-datepicker {
    background-color : #FFFFFF;
    border: 1px solid #393737;
    color: #393737;
    font-family: sourceSansPro, sans-serif;
    font-size: 14px;
    margin-top: 5px;
}

.ui-datepicker a{
    color: #393737;
}
```

```
.ui-datepicker .ui-datepicker-prev {
    left: 10px;
    top: 6px;
}

.ui-datepicker .ui-datepicker-next {
    right: 10px;
    top: 6px;
}

.ui-datepicker .ui-datepicker-prev:before {
    content: "\f104";
    color: #393737;
    display: inline-flex;
    font-family: FontAwesome;
    font-style: normal;
    font-weight: normal;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
    font-size: 18px;
    cursor: pointer;
}

.ui-datepicker .ui-datepicker-next:before {
    content: "\f105";
    color: #393737;
    display: inline-flex;
    font-family: FontAwesome;
    font-style: normal;
    font-weight: normal;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
    font-size: 18px;
    cursor: pointer;
}

.ui-datepicker .ui-datepicker-prev-hover:before {
    content: "\f104";
    color: #393737;
    display: inline-flex;
    font-family: FontAwesome;
    font-style: normal;
    font-weight: normal;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
    font-size: 18px;
    cursor: pointer;
}
.ui-datepicker .ui-datepicker-next-hover:before {
    content: "\f105";
    color: #393737;
    display: inline-flex;
    font-family: FontAwesome;
    font-style: normal;
    font-weight: normal;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
    font-size: 18px;
    cursor: pointer;
    text-align: right;
}

.ui-datepicker .ui-datepicker-prev-hover {
    left: 10px;
    top: 6px;
}

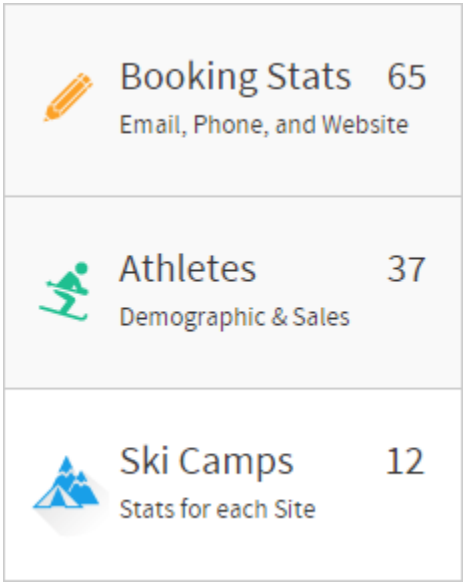
.ui-datepicker .ui-datepicker-next {
    text-align: right;
}
```

```
.ui-datepicker .ui-datepicker-next-hover {
    right: 10px;
    top: 6px;
    text-align: right;
}
```

Buttons

The filter widget displays possible filter value options that the user can select from. Each option has the capacity to display a title, description, icon, and summary value – all except the title are optional.

Example Output



Example Elements

`<report>`

Child	Value	Description
`display`	HIDDEN	HIDDEN allows the results of the report to be used in other components of the page, rather than in its native tabular format. In this case, the report is a dual column report returning a list of filter options, and the matching summary value.
`uuid`	73f732e8-7961-4925-9319-740dc4991de8	This contains the Yellowfin UUID of the report in order to call it.
`name`	skiTeamFilterList	This names the report for use elsewhere in the XML. The <summary> node will call the report by this name, rather than having to call it by UUID, but either would work.
`filterMap`		This contains nodes to define any filters that need to be passed to the report from elsewhere on the dashboard tab. See `<filterMap>` node description.

`<filterMap>`

Child	Value	Description
`filterUUID`	73f732e8-7961-4925-9319-740dc4991de8	This is the UUID of the filter in the hidden report that will receive the `<mapToFilter>` value. In this case, it's the date filter applied to the Position Improvement report.
`mapToFilter`	skiTeamFilter	This is the name of the filter applied elsewhere on the dashboard tab. In this example, we're linking to the skiTeamFilter filters.

<filter>

Child	Value	Description
sync	true	Setting this to true allows the values of this filter to be passed to other dashboards.
datatype	TEXT	This specifies that the filter options need to return a text type in order to be applied.
name	skiTeamFilter	This contains the name of the filter, to be used to identify it for the <filterMap> node used on reports receiving the values.
operator	EQUAL	This specifies the operator for the filter, and must be compatible with the operators used in reports you wish to pass the filter values to. EQUAL means only one value is expected. This option should be used for operators including, but not limited to; Equal to, Greater than, Less than, Contains.
styleClass	quickStatFilter	This contains a CSS class name used to apply styling to the component.
options		This contains nodes to define the contents of the display and possible values. Each option to be presented to the user will need to have its own <option> node. In this example we've used a Booking Stats, Athletes, and Ski Camps options. See <options> node description.

<options> (Booking Stats)

Child	Value	Description
title	Booking Stats	This is the label displayed on the filter. A text name describing the value that will be applied.
description	Email, Phone, and Website	This is a descriptive label to be displayed on the filter. This is optional.
summary		This specifies how to retrieve a value from a report in order to display it as a summary value on the filter button. See <summary> node description.
icon	<![CDATA[<div class="filterIconBooking"></div>]]>	This contains the HTML required to display a specific icon on the filter button.
value	Booking	This specifies the value of the filter.
defaultOption	false	This specifies this option as the default selection for the dashboard. Only one <option> node can be the default, the rest should be false or omit this node.

<summary> (Booking Stats)

Child	Value	Description
reportId	skiTeamFilterList	This is the report name, specified where the report is included in the XML that you will use to retrieve and display the summary value. In this case, the report is a dual column report returning a list of filter options, and the matching summary value.
matchColumn	0	This is the number placement of the column in the tabular report you wish to match a value to in order to locate a specific row. In this case, the column that contains the descriptive labels is the first column in the report, and as we count from zero, use 0.
matchValue	Booking	This is the value you wish to locate in the <matchColumn> column in order to locate a specific row. In this case, you need to locate the Booking value.
valueColumn	1	This is the column you wish to retrieve data from, in the row located using <matchColumn> and <matchValue> nodes. This value starts at zero. In this case, the column with numeric values is second in the report, so use 1.

<options> (Athletes)

Child	Value	Description
title	Athletes	This is the label displayed on the filter. A text name describing the value that will be applied.

description	Demographic & Sales	This is a descriptive label to be displayed on the filter. This is optional.
summary		This specifies how to retrieve a value from a report in order to display it as a summary value on the filter button. See <summary> node description.
icon	<![CDATA[<div class="filterIconAthletes"></div>]]>	This contains the HTML required to display a specific icon on the filter button.
value	Athletes	This specifies the value of the filter.
defaultOption	false	This specifies this option as the default selection for the dashboard. Only one <option> node can be the default, the rest should be false or omit this node.

<summary> (Athletes)

Child	Value	Description
reportId	skiTeamFilterList	This is the report name, specified where the report is included in the XML that you will use to retrieve and display the summary value. In this case, the report is a dual column report returning a list of filter options, and the matching summary value.
matchColumn	0	This is the number placement of the column in the tabular report you wish to match a value to in order to locate a specific row. In this case, the column that contains the descriptive labels is the first column in the report, and as we count from zero, use 0.
matchValue	Athletes	This is the value you wish to locate in the <matchColumn> column in order to locate a specific row. In this case, you need to locate the Athletes value.
valueColumn	1	This is the column you wish to retrieve data from, in the row located using <matchColumn> and <matchValue> nodes. This value starts at zero. In this case, the column with numeric values is second in the report, so use 1.

<options> (Ski Camps)

Child	Value	Description
title	Ski Camps	This is the label displayed on the filter. A text name describing the value that will be applied.
description	Stats for each Site	This is a descriptive label to be displayed on the filter. This is optional.
summary		This specifies how to retrieve a value from a report in order to display it as a summary value on the filter button. See <summary> node description.
icon	<![CDATA[<div class="filterIconCamps"></div>]]>	This contains the HTML required to display a specific icon on the filter button.
value	Camps	This specifies the value of the filter.
defaultOption	true	This specifies this option as the default selection for the dashboard. Only one <option> node can be the default, the rest should be false or omit this node.

<summary> (Camps)

Child	Value	Description
reportId	skiTeamFilterList	This is the report name, specified where the report is included in the XML that you will use to retrieve and display the summary value. In this case, the report is a dual column report returning a list of filter options, and the matching summary value.
matchColumn	0	This is the number placement of the column in the tabular report you wish to match a value to in order to locate a specific row. In this case, the column that contains the descriptive labels is the first column in the report, and as we count from zero, use 0.
matchValue	Camps	This is the value you wish to locate in the <matchColumn> column in order to locate a specific row. In this case, you need to locate the Camps value.
valueColumn	1	This is the column you wish to retrieve data from, in the row located using <matchColumn> and <matchValue> nodes. This value starts at zero. In this case, the column with numeric values is second in the report, so use 1.

Example XML

```
<!-- Ski Team Filter List Summary Report -->
<report>
    <display>HIDDEN</display>
    <uuid>73f732e8-7961-4925-9319-740dc4991de8</uuid>
    <name>skiTeamFilterList</name>
    <filterMap>
        <filterUUID>682f856b-09b8-49d8-bf33-6e9bba9d895a</filterUUID>
        <mapToFilter>skiTeamFilter</mapToFilter>
    </filterMap>
</report>

<!-- Ski Team Filters -->
<filter>
    <sync>true</sync>
    <datatype>TEXT</datatype>
    <name>skiTeamFilter</name>
    <operator>EQUAL</operator>
    <styleClass>quickStatFilter</styleClass>
    <options>
        <title>Booking Stats</title>
        <description>Email, Phone, and Website</description>
        <summary>
            <reportId>skiTeamFilterList</reportId>
            <matchColumn>0</matchColumn>
            <matchValue>Booking</matchValue>
            <valueColumn>1</valueColumn>
        </summary>
        <icon><![CDATA[<div class="filterIconBooking"></div>]]></icon>
        <value>Booking</value>
        <defaultOption>false</defaultOption>
    </options>
    <options>
        <title>Athletes</title>
        <description>Demographic & Sales</description>
        <summary>
            <reportId>skiTeamFilterList</reportId>
            <matchColumn>0</matchColumn>
            <matchValue>Athletes</matchValue>
            <valueColumn>1</valueColumn>
        </summary>
        <icon><![CDATA[<div class="filterIconAthletes"></div>]]></icon>
        <value>Athletes</value>
        <defaultOption>false</defaultOption>
    </options>
    <options>
        <title>Ski Camps</title>
        <description>Stats for each Site</description>
        <summary>
            <reportId>skiTeamFilterList</reportId>
            <matchColumn>0</matchColumn>
            <matchValue>Camps</matchValue>
            <valueColumn>1</valueColumn>
        </summary>
        <icon><![CDATA[<div class="filterIconCamps"></div>]]></icon>
        <value>Camps</value
        <defaultOption>true</defaultOption>
    </options>
</filter>
```

Example CSS

```
.quickStatFilter {
    width: 100%;
    display: table;
    border-collapse: separate;
    border-spacing: 15px 0px;
    z-index: 1000;
}
.quickStatFilter .filterOption {
    display: table-cell;
    background: #EDECED;
    padding: 10px;
    text-align: left;
    cursor: pointer;
    table-layout: fixed;
    overflow: hidden;
}
.quickStatFilter .filterOptionMain {
    width: 100%;
    height: 80px;
    overflow: hidden;
    position: relative;
}
.quickStatFilter .filterTitle {
    color: #4C5965;
    font-size: 16px;
    z-index: 1000;
    font-weight: bold;
}
.quickStatFilter .filterDesc {
    color: #909FAC;
    font-size: 12px;
    margin-top: 2px;
}
.quickStatFilter .filterOptionSummary {
    color: #4C5965;
    font-size: 18px;
    position: absolute;
    font-weight: bold;
    margin-top: 15px;
}
.quickStatFilter .filterIcon {
    position: relative;
    float: right;
    bottom: 30px;
    color: #D3D1D3;
}
.quickStatFilter .filterIconBooking {
    background-image: url("../images/booking.png");
}
.quickStatFilter .filterIconAthletes {
    background-image: url("../images/athlete.png");
}
.quickStatFilter .filterIconCamps {
    background-image: url("../images/camp.png");
}
.quickStatFilter  .selected {
    background: #477DB3;
}
.quickStatFilter  .selected  div{
    color: #FFFFFF;
}
.selected .quickStatFilter .filterIconBooking {
    background-image: url("../images/booking.png");
}
.selected .quickStatFilter .filterIconAthletes {
    background-image: url("../images/athlete.png");
}
.selected .quickStatFilter .filterIconCamps {
    background-image: url("../images/camp.png");
}
```

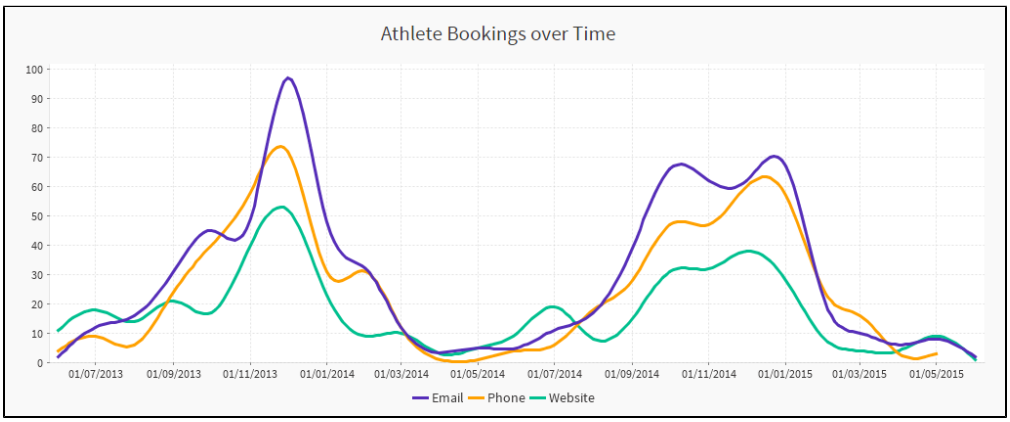
Report Widget

top

Chart

Reports and visualisations can be displayed using the `<report>` XML widget. This will need to call the required report, specify the format and styling, and pass through any filters.

Example Output



Example Elements

`<report>`

Child	Value	Description
`display`	CHART	CHART allows the results of the report to be displayed as a chart, defined in Yellowfin. In this case, the report is a line chart.
`title`	Athlete Bookings over Time	This is the title of the report, to be displayed above the chart on the dashboard tab.
`uuid`	d1f779fb-7dbf-42e2-8e6a-bc7dc84ae04f	This contains the Yellowfin UUID of the report in order to call it.
`styleClass`	fullWidthChart	This contains a CSS class name used to apply styling to the component.
`filterMap`		This contains nodes to define any filters that need to be passed to the report from elsewhere on the dashboard tab. In this example, we've passed the main date filter from the tab to the report (dateFilter). See `<filterMap>` node description.

`<filterMap>`

Child	Value	Description
`filterUUID`	a0bfc7f6-1550-4f4d-a3c5-031b20c1a942	This is the UUID of the filter in the hidden report that will receive the `<mapToFilter>` value.
`mapToFilter`	dateFilter	This is the name of the filter applied elsewhere on the dashboard tab. In this example, we're linking to the dateFilter filters.

Example XML

```
<!-- Line Chart -->
<report>
    <display>CHART</display>
    <title>Athlete Bookings over Time</title>
    <uuid>d1f779fb-7dbf-42e2-8e6a-bc7dc84ae04f</uuid>
    <styleClass>fullWidthChart</styleClass>
    <filterMap>
        <filterUUID>a0bfc7f6-1550-4f4d-a3c5-031b20c1a942</filterUUID>
        <mapToFilter>dateFilter</mapToFilter>
    </filterMap>
</report>
```

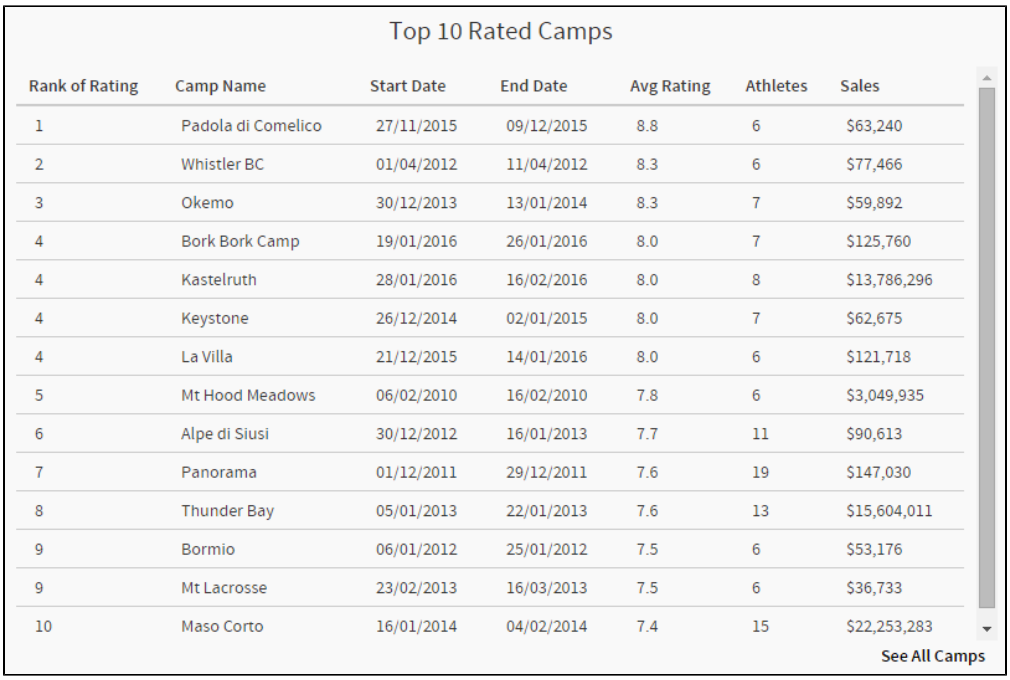
Example CSS

```
.reportDisplay {
    margin: 20px;
    position: relative;
    page-break-inside: avoid;
}
.reportHtml {
    height: 100%;
    padding: 0px 10px 10px 10px;
}
.reportTitle {
    padding: 10px 10px 0px 10px;
    color: #393737;
    font-size: 21px;
    text-align: center;
}
.fullWidthChart  .reportHolder, .twoThirdMap .reportHolder, .thirdTable.reportHolder{
    height: 360px;
}
.fullWidthChart {
    width: 100%;
    float: left;
    overflow: hidden;
}
```

Table

Tables can be displayed using the `<report>` XML widget. This will need to call the required report, specify the format and styling, and pass through any filter values required.

Example Output



Note: the See All Camps link pictured above is called through this widget, but the contents and display of the modal is actually defined through a modal widget. See the Modal Widget section for details.

Example Elements

`<report>`

Child	Value	Description
`display`	TABLE	TABLE displays the results of the report in a tabular format, which does not use standard Yellowfin styling.
`title`	Top 10 Rated Camps	This is the title of the report, to be displayed above the table on the dashboard tab.
`uuid`	d2769b31-5938-4b80-9573-bfe5720440a0	This contains the Yellowfin UUID of the report in order to call it.
`styleClass`	fullTable	This contains a CSS class name used to apply styling to the component.
`filterMap`		This contains nodes to define any filters that need to be passed to the report from elsewhere on the dashboard tab.
`resultsPerPage`	25	This allows you to define how many rows are displayed per page. If the report exceeds this number, pagination will display unless disabled using the `<hidePagination>` node.
`hidePagination`	true	This allows you to hide the pagination that appears by default.
`modalName`	CampRatingListModal	This calls a modal widget defined elsewhere in the XML in order to display content within a modal popup display.
`modalLinkText`	See All Camps	This is the text to be displayed as a link which opens the modal.

`<filterMap>`

Child	Value	Description
`filterUUID`	d10848dc-b892-4890-9933-a463931b0828	This is the UUID of the filter in the hidden report that will receive the `<mapToFilter>` value. In this case, it's the date filter applied to the report.
`mapToFilter`	dateFilter	This is the name of the filter applied elsewhere on the dashboard tab. In this example, we're linking to the dateFilter filters.

Example XML

```
<!-- Top 10 Rated Camps -->
<report>
    <display>TABLE</display>
    <title>Top 10 Rated Camps</title>
    <uuid>d2769b31-5938-4b80-9573-bfe5720440a0</uuid>
    <styleClass>fullTable</styleClass>
    <filterMap>
        <filterUUID>d10848dc-b892-4890-9933-a463931b0828</filterUUID>
        <mapToFilter>dateFilter</mapToFilter>
    </filterMap>
    <hidePagination>true</hidePagination>
    <resultsPerPage>25</resultsPerPage>
    <modalName>CampRatingListModal</modalName>
    <modalLinkText>See All Camps</modalLinkText>
</report>
```

Example CSS

```
.reportDisplay {
    margin: 20px;
    position: relative;
    page-break-inside: avoid;
}
.reportHtml {
    height: 100%;
    padding: 0px 10px 10px 10px;
}
.reportTitle {
    padding: 10px 10px 0px 10px;
    color: #393737;
    font-size: 21px;
    text-align: center;
}

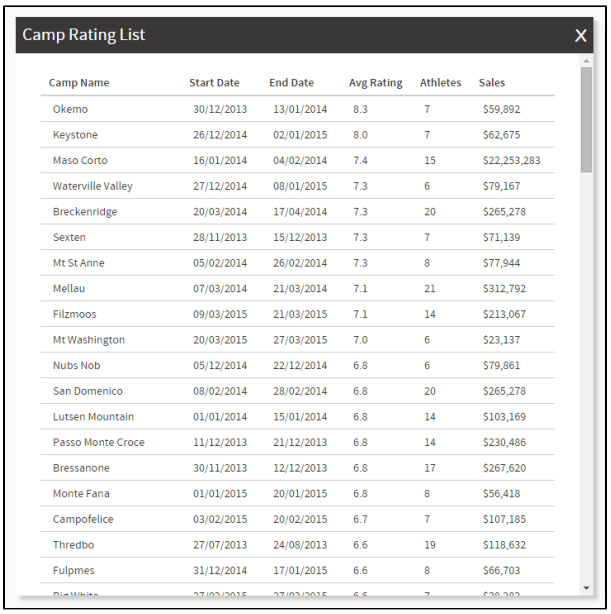
.fullTable {
    width: 100%;
    float: left;
    overflow: hidden;
}
.fullTable .reportTitle {
    padding-bottom: 15px;
}
.fullTable .reportHolder {
    height: 462px;
    overflow-y: auto;
}
.reportModalLink {
    position: relative;
    bottom: -4px;
    float: right;
    right: 10px;
    cursor: pointer;
    font-weight: bold;
}
```

Modal Window Widget

top

The Modal Widget displays a popup window with contents defined within the `<subcontent>` element. The modal itself is called outside of the `<modal>` definition, for example as part of a report. For the purpose of this scenario, we have defined the modal link as part of a report (see Table outline above).

Example Output



Example Elements

`<modal>`

Child	Value	Description
`title`	Camp Rating List	This is the title displayed at the top of the modal window.
`name`	CampRatingListModal	This is the name of the modal, to be called by other widgets in the XML.
`subcontent`		This defines the contents of the modal window. See <subcontent> node description.

`<subcontent>`

Child	Value	Description
`report`		This is the report that will be displayed in the modal window. See `<report>` node description.

`<report>`

Child	Value	Description
`display`	TABLE	TABLE displays the results of the report in a tabular format, which does not use standard Yellowfin styling.
`uuid`	0d16ddf3-749f-4fac-b86e-73cc5dcb1db6	This is the title of the report, to be displayed above the table on the dashboard tab.
`styleClass`	fullTable	This contains a CSS class name used to apply styling to the component.
`filterMap`		This contains nodes to define any filters that need to be passed to the report from elsewhere on the dashboard tab.
`resultsPerPage`	100	This allows you to define how many rows are displayed per page. If the report exceeds this number, pagination will display unless disabled using the `<hidePagination>` node.

`<filterMap>`

Child	Value	Description
`filterUUID`	61ccfce6-6c1b-4221-a5ce-561e1e71feb2	This is the UUID of the filter in the hidden report that will receive the `<mapToFilter>` value. In this case, it's the date filter applied to the report.

`mapToFilter`	dateFilter	This is the name of the filter applied elsewhere on the dashboard tab. In this example, we're linking to the dateFilter filters.

Example XML

```
<!-- "See All Camps" link and popup for the report above -->
<modal>
    <title>Camp Rating List</title>
    <name>CampRatingListModal</name>
    <subcontent>
        <report> <!-- Report to be displayed in modal -->
            <display>TABLE</display>
            <uuid>0d16ddf3-749f-4fac-b86e-73cc5dcb1db6</uuid>
            <styleClass>fullTable</styleClass>
            <filterMap>
                <filterUUID>61ccfce6-6c1b-4221-a5ce-561e1e71feb2</filterUUID>
                <mapToFilter>dateFilter</mapToFilter>
            </filterMap>
            <resultsPerPage>100</resultsPerPage>
        </report>
    </subcontent>
</modal>
```

Example CSS

```
/* Modal Styling */
.reportModalLink {
    position: relative;
    bottom: -4px;
    float: right;
    right: 10px;
    cursor: pointer;
    font-weight: bold;
}
.modal {
    display: block;
    height: 100%;
    width:calc(100% - 230px);
    position:fixed;
    top:0;
    left:230px;
}
.modalBackground {
    position: absolute;
}
.modalContent {
    position: absolute;
    background-color:#FFFFFF;
    box-shadow: 5px 5px 5px #CCCCCC;
    height: 700px;
    width: 700px;
}
.modalContent .modalTopBar {
     background-color: #393737;
     color: #FFFFFF;
     overflow:hidden;
     padding: 8px;
}
.modalContent .modalTitle {
    font-size: 21px;
    float: left;
    font-family: sourceSansPro, sans-serif;
}
.modalContent .modalClose {
    float: right;
    font-size: 20px;
    font-family: cursive;
    cursor: pointer;
}
.modalContent .modalMainContent {
    clear:both;
    height: 100%;
}
.modal .reportHolder {
    height: auto;
}
.modal .fullTable {
    height: 656px;
    overflow-y: auto;
    overflow-x: hidden;
}
```

Custom HTML Widget

top

Text & Report Data

The data from a report can be used within a <customHTML> XML Widget along with other elements to produce a combined output. In this scenario, we wanted to display the first date we have records for within the user selected filter date range. To do this we wanted to combine some static text with a date record from a report. The widget needed to contain HTML to display the static text, such as "Your camp started on: ", and the dynamic date element that comes from the result of a Yellowfin report. In order to do this, you will also need to include the report that returns the date as a hidden report.

Example Output



Example Elements

<report>

Child	Value	Description
display	HIDDEN	HIDDEN allows the results of the report to be used in other components of the page, rather than in its native tabular format. In this case, the report is a single row report returning the Camp Start Date, filtered by the date filter on the tab.
uuid	b0ebacf2-c51d-41c6-8a3f-09cef22adb67	This contains the Yellowfin UUID of the report in order to call it.
filterMap		This contains nodes to define any filters that need to be passed to the report from elsewhere on the dashboard tab. In this example, we've passed the main date filter from the tab to the report (dateFilter). See <filterMap> node description.

<filterMap>

Child	Value	Description
filterUUID	ea41ba19-6362-4e97-b607-cc66d22cc5cb	This is the UUID of the filter in the hidden report that will receive the <mapToFilter> value. In this case, it's the date filter applied to the Camp Start Date report.
mapToFilter	dateFilter	This is the name of the filter applied elsewhere on the dashboard tab. In this example, we're linking to the dateFilter filters, defined in the Filter section of this document.

<customhtml>

Child	Value	Description
styleClass	dashboardDates	Contains a CSS class name used to apply styling to the component.
html	<![CDATA[Your camp started on: <@-campStart @>]]>	Contains the HTML to be used as the component, wrapped in a CDATA section. In this example, we have the CDATA tag containing static text, and the <@- campStart @> parameter which references the campStart replacement, defined in the <replacements> node.
replacements		This allows you to place a parameter in the custom html that will be replaced with values of a filter or report, if required. In order to call the parameter, place the replaceName value in <@- ->. See <replacements> node description.

<replacements>

Child	Value	Description
replaceName	campStart	This is the name given to the parameter in the <customhtml> node. In this example, we've called the replacement campStart.
reportName	b0ebacf2-c51d-41c6-8a3f-09cef22adb67	This is the UUID of the report. If you've used the report elsewhere on the dashboard, you could put the report name in this node. In this example, we're only using the report to provide a value for the custom HTML, nothing else.

`value Column`	0	This is the column of the report that the value will come from. If the report had multiple columns and rows, you would also need the <matchColumn> and <matchValue> nodes, but in this example our report is only one cell. The numbering of the columns starts at 0.

Example XML

```
<!-- Report to return Camp Start Date -->
<report>
    <display>HIDDEN</display>
    <uuid>b0ebacf2-c51d-41c6-8a3f-09cef22adb67</uuid>
    <filterMap>
        <filterUUID>ea41ba19-6362-4e97-b607-cc66d22cc5cb</filterUUID>
        <mapToFilter>dateFilter</mapToFilter>
    </filterMap>
</report>

<!-- Custom HTML to display Camp Start Date -->
<customhtml>
    <styleClass>dashboardDates</styleClass>
    <html><![CDATA[<div>Your camp started on: <@- campStart @></div>]]></html>
    <replacements>
        <replaceName>campStart</replaceName>
        <reportName>b0ebacf2-c51d-41c6-8a3f-09cef22adb67</reportName>
        <valueColumn>0</valueColumn>
    </replacements>
</customhtml>
```

Example CSS

```
.dashboardDates {
    color: #909FAC;
    font-size: 13px;
    margin: 2px 0px 0px 0px;
    padding: 0px 15px;
}
```

top