Aras Innovator 12

XSLT Report Tool Users Guide

Document #: 12.0.02015056501

Last Modified: *1/3/2020*

Copyright Information

Copyright © 2020 Aras Corporation. All Rights Reserved.

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810

Phone: 978-691-8900
Fax: 978-794-9826

E-mail: Support@aras.com

Website: https://www.aras.com/support/

Notice of Rights

Copyright © 2020 by Aras Corporation. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at http://www.opencontent.org/openpub/).

Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.

Distribution of the work or derivative of the work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

Aras Innovator, Aras, and the Aras Corp "A" logo are registered trademarks of Aras Corporation in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

Notice of Liability

The information contained in this document is distributed on an "As Is" basis, without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement. Aras shall have no liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this document or by the software or hardware products described herein.

Table of Contents

©2020 Aras Corporation All Copyrights Reserved.

Send Us Your Comments

Aras Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for future revisions.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where and what level of detail?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, indicate the document title, and the chapter, section, and page number (if available).

You can send comments to us in the following ways:

Email:
Support@aras.com
Subject: Aras Innovator Documentation

Or,

Postal service:
Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Attention: Aras Innovator Documentation

Or,

FAX:
978-794-9826
Attn: Aras Innovator Documentation

If you would like a reply, provide your name, email address, address, and telephone number.

If you have usage issues with the software, visit https://www.aras.com/support/

©2020 Aras Corporation All Copyrights Reserved.

Document Conventions

The following table highlights the document conventions used in the document:

Table 1: Document Conventions

Convention	Description
Bold	Indicates the names of menu items, dialog boxes, dialog box elements, and commands. Example: Click **OK**.
`Code`	Code examples appear in `courier` font. It may represent text you type or data you read.
`Yellow highlight`	Code highlighted in yellow draws attention to the code that is being indicated in the content.
`Yellow highlight with red text`	Red text highlighted in yellow indicates the code parameter that needs to be changed or replaced.
Italics	Reference to other documents.
Note:	Notes contain additional useful information.
Warning	Warnings contain important information. Pay special attention to information highlighted this way.
Successive menu choices	Successive menu choices may appear with a greater than sign (-->) between the items that you will select consecutively. Example: Navigate to **File** --> **Save** --> **OK**.

©2020 Aras Corporation All Copyrights Reserved.

1 Introduction

The Report Tool is the user interface for creating and maintaining Reports about items and their configurations. The intent of the Report Tool is to offer a very simple user interface for designing Reports.

In technical terms, the Reports are the result of an XSLT transformation of an AML item configuration, which is the result of a query. The Report Tool allows direct editing of the XSLT style sheet.

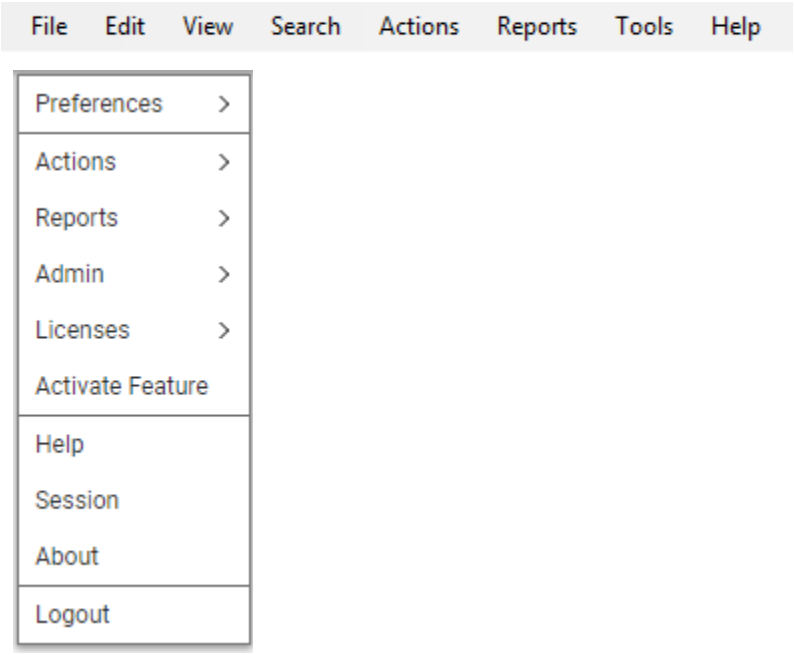
Reports are relationships on ItemTypes. The Report item has properties that define how to query data for the Report and format it for display. Reports tend to fall into the following types:

- A report about an item and its relationships. For example, a BOM report. These are Item type Reports.
- A static query report about items by ItemType. For example, an Open ECO report. These are ItemType type Reports.
- A dynamic query report about items by ItemType. The user interactively enters query criteria for the report. These are Generic type Reports.

©2020 Aras Corporation All Copyrights Reserved.

2 Running Reports

The main menu bar has a **Reports** menu choice, which is context sensitive as you navigate the main tree. The drop down menu is populated with the Report items related to the ItemType you selected from the tree.



How Reports appear on the **Reports** menu:

- The **Generic** type reports are always shown. (Generic Reports tend to be interactive and the user interface will be implemented in future release.)
- The **ItemType** type reports are shown when you select an ItemType from the main tree.
- The **Item** type reports are only shown when you select an item. Additionally, Item Reports appear on the right-mouse context popup menu.

©2020 Aras Corporation All Copyrights Reserved.

3 Report ItemType

Use the Report Tool to edit Report items, which are instances of the Report ItemType. This section provides a brief overview of the Report ItemType as well as a technical understanding of how Reports are implemented in Aras Innovator.

Reports are the result of an AML query or a Method. The Report ItemType implements the properties for you to define the query and XSLT style sheet or Method. The Report also describes where and how the Report is viewed on the client.

The following table lists the Report ItemType properties:

Table 2: ItemType Report Properties

Property	Data Type	Length/ Source	Content/Comments
name	string	32	The name for the Report.
description	string	128	The description for the Report.
type	list	Report Types	This specifies the type of Report: Item, ItemType, or Generic. The type of item defines how the Report is handled by the server and when it appears on the Reports main menu bar.
target	list	Report Targets	This specifies where to show the Report results: in the main window, its own window, or all reports in one window when many items are multi-selected.
label	Multilingual String	32	This allows us to specify the text, in different languages, of Menu entry for this Report under the Reports Menu of the Client interface.
report_query	text		The AML query to generate data for the Report or an XSLT style sheet to generate the actual AML query if the Report type is Item.
xsl_stylesheet	text		The XSLT style sheet used to transform the query results into the Report.
method	item	Method	The Method is evaluated and the result is the Report. If a Report is based on the results of a Method the report_query and xsl_stylesheet property are not used.

©2020 Aras Corporation All Copyrights Reserved.

Property	Data Type	Length/ Source	Content/Comments
location	list	Report Locations	This defines where the logic for generating the Report is performed: on the client or server. Typically this is server side but it is possible to run a Method client side to generate a Report.

ItemTypes have relationships to Reports called the 'Item Report' RelationshipType. You can create new Reports and edit existing Reports from the Reports tab on the ItemType window. Reports are related items because Generic Reports are not specific to any particular ItemType thus are standalone Reports. Reports tend to be based on an item or items by ItemType and their configurations.

©2020 Aras Corporation All Copyrights Reserved.

4 Report Tab

When you open a Report item from the Reports tab on the ItemType window the Report Tool opens. This is the standard Tear off Item Window including the standard menu bar and toolbar. Below that is the Report tab bar allowing you to switch between the Report details and the XSLT style sheet text editor. This section describes the Report tab, which is the Report Form. The Report tab is the page where you set up the Report. Here you provide its name, its type, the target window to show the Report results plus set up the query criteria to generate the data for the Report.

The following screenshot shows a sample Report that is of type ItemType, meaning the Report Query is a static query that searches for Vendor items. The AML query also limits the results by defining only the properties desired using the select attributes. The target is set to One Window so the Report appears in a separate window when run from the client.

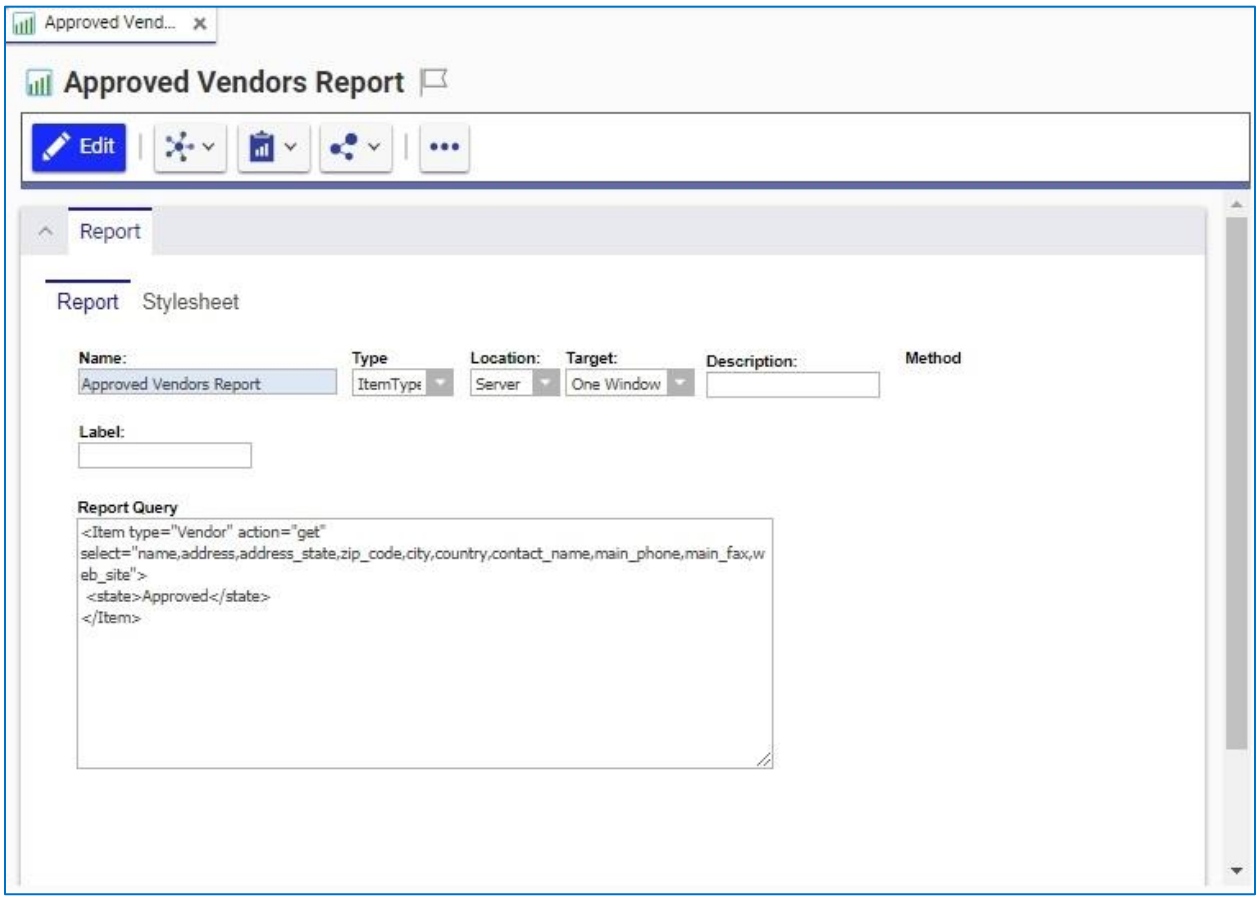


Figure 1.

©2020 Aras Corporation All Copyrights Reserved.

The following table provides a description for each field on the Report Form.

Table 3: Report Form Fields

<table>
<tr><th>Field</th><th>Content/Comments</th></tr>
<tr><td>Name</td><td>The Report name</td></tr>
<tr><td>Description</td><td>The Report description</td></tr>
<tr><td>Type</td><td>
<table>
<tr><th>Type</th><th>On Menu Bar</th><th>Report Query Contains</th></tr>
<tr><td>Generic</td><td>Always</td><td>The Report Query is an AML query for the Report, but if not set then use the <Item> passed in the body of the client request as the AML query. These are dynamic Reports where the query is known at run time.</td></tr>
<tr><td>ItemType</td><td>When selected off the main tree.</td><td>The report_query is the AML query.</td></tr>
<tr><td>Item</td><td>When selected off the main search grid.</td><td>The report_query is an XSLT style sheet applied against the <Item> passed in the body of the client request as the AML query, which is the selected item from the main search page.</td></tr>
</table>
</td></tr>
<tr><td>Target</td><td>Where to show the Report results: in the main window, its own window, and all reports in one window.</td></tr>
<tr><td>Location</td><td>Where the method for generating the Report is evaluated on the client or server.</td></tr>
<tr><td>Report Query</td><td>The AML query or XSLT style sheet to make the AML query from the item depending on whether the type is set to Item or ItemType.</td></tr>
<tr><td>Label</td><td>This allows us to specify the text, in different languages, of Menu entry for this Report under the Reports Menu of the Client interface.</td></tr>
</table>

4.1 Report Query

When the Report type is set to ItemType or Generic, the Report Query is the static AML query used to generate the data for the Report. The following attributes are very useful to help filter and control the results for your Report.

- **select** – the select attribute is used to define the set of properties you want returned from the query. This is a comma delimited list of property names (not the property label). It is identical to the select clause in SQL if you are familiar with relational database technology. You can gain significant performance improvements to your Reports by limiting the data returned from the query.

  The following example illustrates the select attribute by selecting the item_number, description, and cost properties from the Part ItemType:

```
<Item action="get" type="Part" select="item_number,description,cost"/>
```

©2020 Aras Corporation All Copyrights Reserved.

- **order_by** – the order_by attribute is used to order the results. The Report Tool offers the ability to sort the data in a variety of ways but sometimes you may want the data in a pre-sorted order. This is a comma delimited list of property names (not the property label). This is identical to the order by clause in SQL if you are familiar with relational database technology.

  The following example illustrates the select attribute by selecting the item_number, description, and cost properties from the Part ItemType.

```
<Item
 action="get"
 type="Part"
 select="item_number,description,cost"
 order_by="item_number"/>
```

- **page & pagesize** – if you are performing a query for items and you know that there can be hundreds or thousands of items, you can limit the number of items returned by using the page and pagesize attributes. Together they control how many items to return (the pagesize) and which page to return.

  The following example illustrates the select attribute by selecting the item_number, description, and cost properties from the Part ItemType.

```
<Item
 action="get"
 type="Part"
 select="item_number,description,cost"
 order_by="item_number"
 page="1"
 pagesize="50"/>
```

- **Relationships** – often you want the relationship items in addition to the source item to generate a Report showing the items configuration. The query may contain the <Relationships> tag to hold the <Item> tags for the relationship items you want included in the query results. The <Item> tags can themselves include the same control attributes.

  The following example illustrates the Relationships tag to include the BOM relationships. Notice the select attribute for the BOM type Item defines the properties to be returned for both the BOM relationship item plus the properties for the Part item referenced by the related_id property, which is defined within the parenthesis following the related_id property.

```
<Item
  action="get"
  type="Part"
  select="item_number,description,cost"
  order_by="item_number">
  <Relationships>
    <Item
      action="get"
      type="Part BOM"
      select="qty,ref_des,related_id(item_number,description,cost)"/>
  </Relationships>
</Item>
```

©2020 Aras Corporation All Copyrights Reserved.

4.2 Stylesheet Query

When the Report type is set to Item, the Report Query is an XSLT style sheet. Use the stylesheet to transform AML for the selected item into a new AML query that you can use to query the data for the Report. The reason for creating the new query is that the item selected tends not to already include the item relationships, so you need to generate the actual query dynamically using the data known from the selected item.

For example, the following is an XSLT style sheet that transforms a Part item into a new AML query for a BOM Report. Notice that for the most part this is actually an AML query but because it's an XSLT style sheet the Attribute value templates {…} can be used to substitute values inline from the selected item. The text within the curly brackets is evaluated as an expression, which in this case is simply returning the values for the type and id attributes from the selected item.

```
<Item
  action="get"
  type="{@type}"
  id="{@id}"
  select="item_number,description,cost">
  <Relationships>
    <Item
      action="get"
      type="Part BOM"
      select="qty,ref_des,related_id(item_number,description,cost)"/>
  </Relationships>
</Item>
```

©2020 Aras Corporation All Copyrights Reserved.

5 Stylesheet Tab

The Stylesheet Tab presents the XSLT style sheet for the Report in its native format so you can edit the stylesheet to make changes. This is a simple text area where you can edit the XSLT style sheet directly.

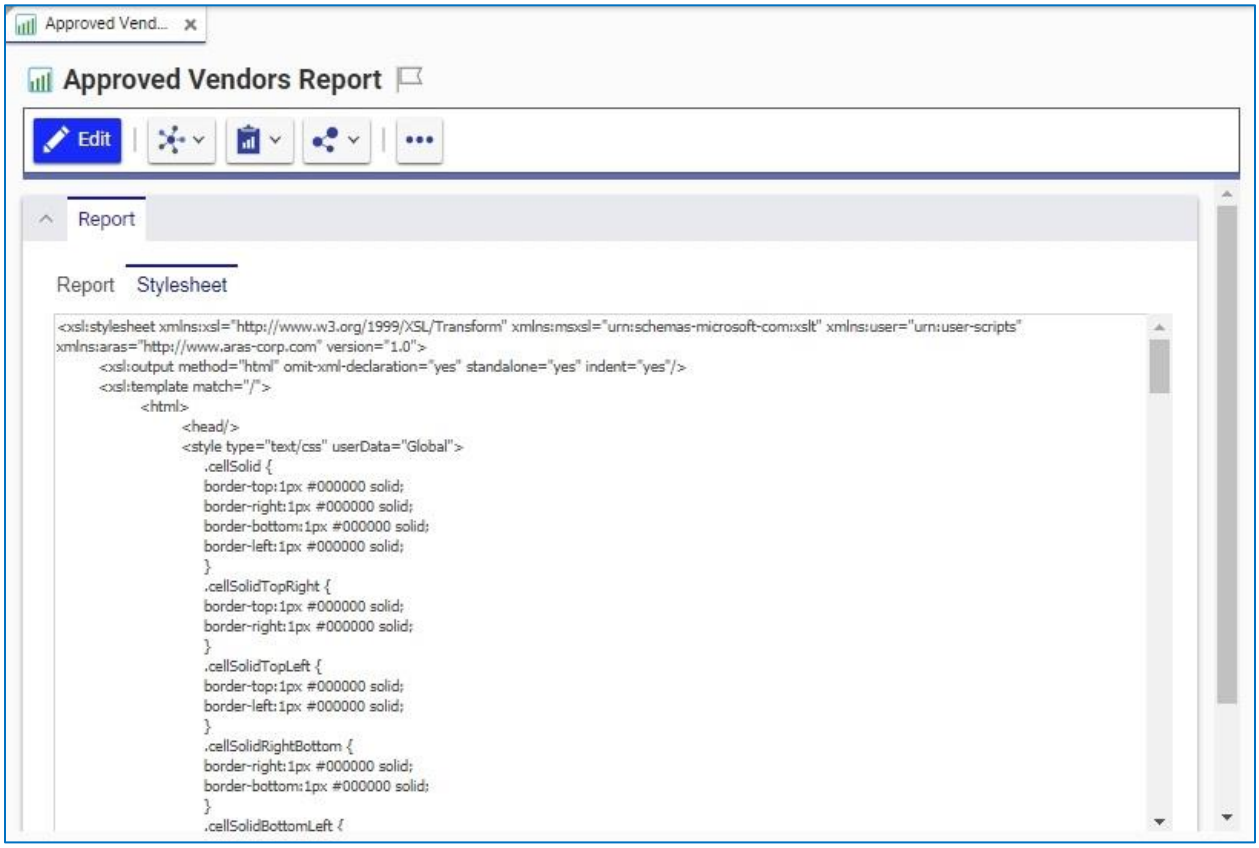


Figure 2.

- Clicking **Apply** applies the changes.
- Clicking **Reset** reverts back to the text before the changes.

©2020 Aras Corporation All Copyrights Reserved.

6 Report Tool Tips and Tricks

This document is a guide to help you become more productive with the Report Tool by showing you how to address some of the more common tasks that need to be done when designing a Report.

6.1 Query Tips

- Using the select attribute

  The select attribute enables you to filter down the set of properties for the Items returned from a query.

```
<Item type="Part" action="get"
select="part_number,part_description,part_cost"/>
```

- Using the order_by attribute

  The order_by attribute enables you to order the results of the query. The style sheet also supports sorting on the properties.

```
<Item type="Part"
      action="get"
      select="part_number,part_description,part_cost"
      order_by="part_number"
/>
```

- Using the condition attribute

  The condition attribute enables you to specify a condition for the properties used as search criteria. All the conditions that are allowed in SQL are allowed as the values to the condition attribute.

Note: The greater then and less then symbols are replaced with the two letter pneumonic word. The < symbol is lt and the > symbol is gt, >= is ge and so on.

```
<Item type="Part"
      action="get"
      select="part_number,part_description,part_cost"
      order_by="part_number">
  <part_number condition="like">%123%</part_number>
</Item>
```

- OR'ing property values

  The <or> tag provides a way to logically OR property values for the search criteria.

```
<Item type="Part"
      action="get"
      select="part_number,part_description,part_cost"
      order_by="part_number">
```

©2020 Aras Corporation All Copyrights Reserved.

```
  <or>
    <state>Released</state>
    <state>In Review</state>
  </or>
</Item>
```

- Return Relationships in the results

  You can include the <Relationships> tag and the relationship items you want to return.

```
<Item type="{@type}" id="{@id}" action="get"
select="part_no,part_desc,part_cost">
  <Relationships>
    <Item type="Part EBOM" action="get"
select="position,qty,related_id">
      <related_id>
        <Item type="Part" action="get"
select="id,part_no,part_desc,part_cost">
          <Relationships>
            <Item type="Part AVL" action="get"
select="avl_status,related_id(description,manufacturer,component_no)"/>
          </Relationships>
        </Item>
      </related_id>
    </Item>
  </Relationships>
</Item>
```

- Using Relationships as search criteria

  You can include the <Relationships> tag to describe the search criteria for the relationship items.

```
<Item type="{@type}" id="{@id}" action="get"
select="part_no,part_desc,part_cost">
  <Relationships>
    <Item type="Part EBOM" action="get"
select="position,qty,related_id">
      <qry condition="gt">1000</qty>
    </Item>
  </Relationships>
</Item>
```

©2020 Aras Corporation All Copyrights Reserved.

6.2 Stylesheet Tips

- Displaying a non-breaking white space

  To display a non-breaking white space, use the <xsl:text> tag.

  ```
  <xsl:text disable-output-escaping="yes">&amp;nbsp;</xsl:text>
  Or this:  
  ```

- Displaying the formatted text property

  To display formatted text (the marked up text) property value in XSLT, use the disable-output-escaping attribute with the <xsl:value-of> tag. The marked up text for the formatted text property is a complete HTML document and only the content for the BODY tag is required.

  ```
  <xsl:value-of disable-output-escaping="yes" select=".//BODY/*"/>
  ```

- Displaying nested tables for relationships.

  To display the relationship Items for nested tables, reduce the XPath to the relative path based on the context node for the <xsl:for-each> loop that the nested relationships table is in.

  ```
  <xsl:for-each select="Relationships/Item">
    <xsl:for-each select="related_id/Item/Relationships/Item">
    </xsl:for-each>
  </xsl:for-each>
  ```

- Resize the Report window.

  To resize the report window, add an onload callback function for the HTML page. Add the following code before the <body> tag:

  ```
  <script>
  onload = function()
  {
    top.window.resizeTo(800,600);
  }
  </script>
  ```

- Show a different value based on the value for a property.

  To show a different value in the Report based on the actual value of the property. For example, the actual value of the property is the hex number for a color but you want to show the name of the color in text.

  ```
  <xsl:choose>
    <xsl:when test="mgr_opinion='#FF0000'">Red</xsl:when>
    <xsl:when test="mgr_opinion='#FFFF00'">Yellow</xsl:when>
    <xsl:when test="mgr_opinion='#00FF00'">Green</xsl:when>
  </xsl:choose>
  ```

- Using script functions in the style sheet.

  To call a JavaScript or VBscript function in the style sheet, there is a predefined namespace named "user" that is used to implement your <msxsl:script> block. The first step is to add the JavaScript or VBscript to your stylesheet and the second step is to call the function defined.

©2020 Aras Corporation All Copyrights Reserved.

The following example shows how to call the VBscript DatDiff() function:

```
<msxsl:script language="VBScript" implements-prefix="user">
  Function myDateDiff(day)
    myDateDiff = DateDiff("d", day.item(0).text, Now)
  End Function
</msxsl:script>
Add the code above in the stylesheet after the <xsl:stylsheet> tag.
Then call the function using the <xsl:value-of> tag like this:

<xsl:value-of select="user:myDateDiff(created_on)"/>
```

- Supporting European languages.

  To show the results in a European language:

  a. Set the encoding attribute for the root xml tag:

  ```
  <?xml version="1.0" encoding="iso-8859-1" ?>
  ```

  b. Set the encoding attribute for the output tag:

  ```
  <xsl:output encoding="iso-8859-1" method="html"
  omit-xml-declaration="yes" standalone="yes" indent="yes" />
  ```

©2020 Aras Corporation All Copyrights Reserved.

7 AML Schema

The following is the AML Schema:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="Item">
    <xsd:all>
<!-- Any user elements defined by the ItemType for the item -->
      <xsd:complexType name="Relationships" minOccurs="0" maxOccurs="1">
        <xsd:all>
          <xsd:element name="Item" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:all>
<!-- Any user elements defined by the ItemType for the item -->
                <xsd:attribute name="id" type="xsd:ID" />
                <xsd:attribute name="type" type="xsd:string" />
                <xsd:attribute name="typeID" type="xsd:IDREF" />
                <xsd:attribute name="action" type="xsd:string" />
                <xsd:attribute name="sort_order" type="xsd:integer" />
              </xsd:all>
            </xsd:complexType>
          </xsd:element>
        </xsd:all>
      </xsd:complexType>
    </xsd:all>
    <xsd:attribute name="id" type="xsd:ID" />
    <xsd:attribute name="type" type="xsd:string" />
    <xsd:attribute name="typeID" type="xsd:IDREF" />
    <xsd:attribute name="action" type="xsd:string" />
    <xsd:attribute name="select" type="xsd:string" />
    <xsd:attribute name="order_by" type="xsd:integer" />
    <xsd:union>
      <xsd:attributeGroup>
        <xsd:attribute name="pagesize" type="xsd:integer" />
        <xsd:attribute name="page" type="xsd:integer" />
      </xsd:attributeGroup>
      <xsd:attributeGroup>
        <xsd:attribute name="levels" type="xsd:integer" />
        <xsd:attribute name="config_path" type="xsd:string" />
      </xsd:attributeGroup>
    </xsd:union>
  </xsd:complexType>
</xsd:schema>
```

©2020 Aras Corporation All Copyrights Reserved.

8 Cookbook

This section provides a set of recipes for performing common reporting tasks.

8.1 User Input for Reporting Services-based Reports

You need a Form to collect user input for Named Parameters for the SQL query for Reporting Services based Reports.

Technique

A Client Method-based Report is used to present a Form to collect the user input. The user input is mapped to a query_string for the Report Server request via the RSGatwway.aspx page.

Procedure for building a user input Form for Reporting Services-based Reports with Named Parameters:

1. Login to Aras Innovator as **admin**.
2. Navigate to the **Administration->Forms** off the Main Tree.
3. Open a new Form Item.
4. Set the Form Item properties as follows:
   a. Name = My Report (this can be any name you want)
   b. Add a form event that fires onLoad
      - Name the Method 'My `Report_setDefaults`' (this can be anything you want).
   c. Add fields to the form that will match the input you are requesting.
      - Name the fields appropriately ('PartNumber', 'Description', 'Cost')
   d. Add a Button field with a label 'Submit'
      1. Create a Field Event that fires onClick.
      2. Name the Method 'My `Report_OnClick`' (this can be anything you want)
   e. Save, unlock, and close.
5. Navigate to the **Administration->Methods** off the Main Tree.
6. Edit the My `Report_OnClick` Method Item.
7. Set the method code as shown.
   - Set document.forms[0].<FieldNameOnForm>.value to the names of the fields on the My Report Form.

```
var retVal=[];

// retVal will be used in MyReport_OpenDialog
// document.forms[0].<FieldNameOnForm>.value
retVal["PartNum"] = document.forms[0].PartNumber.value;
```

©2020 Aras Corporation All Copyrights Reserved.

```
    retVal["Desc"] = document.forms[0].Description.value;
    retVal["Cost"] = document.forms[0].Cost.value;

    parent.args.dialog.result = retVal;
    parent.args.dialog.close();
```

8. Save, unlock, and close.
9. Create another Method Item (‘`MyReport_OpenDialog`’, this can be called whatever you want).
10. Set the method code as shown:
    a. Update `var reportName` with the name of your Report as deployed to the Report Server.
    b. The URL query strings (“&np:”) should reference the parameter as defined in the report.

```
var inn = this.getInnovator();
var ReportInputForm = inn.newItem("Form", "get");
ReportInputForm.setAttribute("select", "id, width, height");
ReportInputForm.setProperty("keyed_name", "My Report");
ReportInputForm = ReportInputForm.apply();

var param =
{
    title: "My Report",
    formId: ReportInputForm.getID(),
    isEditMode: "1",
    aras: aras,
    opener: window,
    innovator: inn,
    // use param.item only if the Report is of Type=Item
    //and you need to pass the context item to the dialog
    //item : this
   dialogHeight: ReportInputForm.getProperty('height','800'),
   dialogWidth: ReportInputForm.getProperty('width','600'),
   content: 'ShowFormAsADialog.html'
};

function callback (dialogWrapper)
{
    var result = dialogWrapper.result;
    if (!result){ return }
    var reportName = "My Report"; // Your Report Name Here
```

©2020 Aras Corporation All Copyrights Reserved.

```
    var url = top.aras.getServerBaseURL() + "RSGateway.aspx?irs:Report=" +
       reportName +
                "&np:Item_Number="+result.PartNum +
                "&np:Description="+result.Desc +
                "&np:Cost="+result.Cost;
    var w = window.open();
    w.location = url;
    return "<html></html>";
}


var wnd = top.aras.getMainWindow();
wnd = wnd === top ? wnd.main : top;
wnd.ArasModules.Dialog.show("iframe", param).promise.then(callback);
```

11. Save, Unlock, and Close
12. Edit the My `Report_SetDefaults` Method Item.
13. Set the method code as shown:

```
var fields = ["PartNumber", "Description", "Cost"];
var values = ["%",          "%",     "%"];
for (var i = 0; i < fields.length; i++){
   // currentField = <FieldID> + "span"
   var currentFieldID =  getFieldByName(fields[i]).id;
   // currentField = <Actual field ID>
   currentFieldID =
     currentFieldID.substring(0,currentFieldID.indexOf('s'));
   var currentField = document.getElementById(currentFieldID);
   currentField.value = values[i];
}
```

14. Save, unlock, and close.
15. Navigate to the **Administration->Reports** off the Main Tree.
16. Open a new Report Item.
17. Set the Report Item properties as follows:
    - **Name** = My Report (this can be anything you want).
    - **Type** = Generic, ItemType, depending on when you want the Report to appear on the Report menu; Generic always, ItemType only when that ItemType is selected off the Main Tree.
    - **Location** = Client; this is required because the `My Report_OpenDialog` Method returns an HTML page with JavaScript that needs to be parsed and Client side Report only writes the transformed page to the browser window and thus the JavaScript is not parsed.
    - **Target** = None; the windows will be created by `My Report_OpenDialog.`
    - **Method** = `My Report_OpenDialog`
    - **Report Query** = Leave blank; it is currently not used for this purpose

©2020 Aras Corporation All Copyrights Reserved.

The Report Item should now appear on the Report menu and when selected the User Input Dialog should open. When the user clicks **Submit** the user input is mapped as a query_string and the actual Report is requested from the Report Server via the RSGateway.

If you want to add validation, add this either in My Report_onClick or in MyReport_OpenDialog, after the modal dialog is returned.

8.2 Report Chooser

You need a dialog to present a set of Reporting Services-based Reports to choose from for the selected Item.

Technique

A Client Method based Report is used to present a Form to get the Report choice. The user selection is mapped to a query_string for the Report Server request via the RSGatwway.aspx page. This is similar to recipe 8.1 User Input Form but in this case we do want the ID for the selected Item to be passed as a Named Parameter on the query_string.

Procedure for building a Report Chooser dialog for Reporting Services based Reports:

1. Login to Aras Innovator as **admin**.
2. Navigate to the **Administration->Lists** off the Main Tree.
3. Open a new List Item
4. Set the List Item properties as follows:
   a. Name = `ReportChooserOptions`
   b. Add values with Labels and Values that match the names of your Reports as deployed to the Report Server.
5. Save, unlock, and close
6. Navigate to the **Administration->Forms** off the Main Tree.
7. Open a new Form Item.
8. Set the Form Item properties as follows:
   a. Name = Report Chooser (this can be any name you want)
   b. Add a form event that fires onLoad
      - Name the Method '`ReportChooser_onLoad`' (this can be anything you want)
   c. Add a new Radio Button List:
      - Name this field 'rdio1'
   d. Add a Button field with a label **Submit**:
      1. Create a Field Event that fires onClick.
      2. Name the Method '`ReportChooser_OnClick`' (this can be anything you want).
   e. Save, unlock, and close.
9. Navigate to the **Administration->Methods** off the Main Tree.
10. Edit the `ReportChooser_OnClick` Method Item.
11. Set the method code as shown:

©2020 Aras Corporation All Copyrights Reserved.

```
var inputOptions = document.getElementsByTagName("input");
var retVal = [];

for (var i = 0; i < inputOptions.length; i++){
   if (inputOptions[i].className != "arasCheckboxOrRadio"){
      continue;
   }
   if (inputOptions[i].checked === true){
      retVal['reportSelected'] = inputOptions[i].value;
      parent.args.dialog.result = retVal;
   }
}
parent.args.dialog.close();
```

12. Save, unlock, and close.
13. Create another Method Item (`'ReportChooser_OpenDialog'`), this can be called whatever you want).
14. Set the method code as shown:
    a. The URL query strings ("&np:") should reference the parameter as defined in the report.

```
var inn = this.getInnovator();

var itemIDs = typeof(thisItem)!=="undefined" ? thisItem.getID() :
aras.getMainWindow().main.work.grid.getSelectedItemIds(',').split(',')[
0];

var ReportInputForm = inn.newItem("Form", "get");

ReportInputForm.setAttribute("select", "id, width, height");

ReportInputForm.setProperty("keyed_name", "ReportChooser");

ReportInputForm = ReportInputForm.apply();

var param =

{

    title: "ReportChooser",

    formId: ReportInputForm.getID(),

    isEditMode : "1",

    aras : aras,

    opener : window,

    innovator : inn,

    item : this,

   dialogHeight: ReportInputForm.getProperty('height','800'),
```

©2020 Aras Corporation All Copyrights Reserved.

```
        dialogWidth: ReportInputForm.getProperty('width','600'),
        content: 'ShowFormAsADialog.html'
    }
    function callback(dialogWrapper)
    {
        var result = dialogWrapper.result;
        if (!result || !result['reportSelected']){ return; }
       var test=0;
        var url = aras.getServerBaseURL() +
            "RSGateway.aspx?irs:Report=" + result['reportSelected'] +
                "&np:id=" + itemIDs; // and the Item ID.
        var w = top.window.open();
        w.location = url;
        return "<html></html>";
    }

    var wnd = aras.getMainWindow();
    wnd = wnd === top ? wnd.main : top;
    wnd.ArasModules.Dialog.show("iframe", param).promise.then(callback);
```

15. Save, Unlock, and Close
16. Edit the ReportChooser_onLoad Method Item.
17. Set the method code as shown.
    a. var radioButtonSpan = getFieldByName('rdio1') uses the name of the radio button field as defined on the ReportChooser Form Item.

```
var radioButtonSpan = getFieldByName('rdio1');
var sysValueSpan = radioButtonSpan.getElementsByClassName("sys_f_value");
if (sysValueSpan.length != 1){
   return;
}
sysValueSpan = sysValueSpan[0];

var inn = this.params.thisItem.getInnovator();
var reports = inn.newItem("List", "get");
reports.setAttribute("levels", "2");
```

©2020 Aras Corporation All Copyrights Reserved.

```
reports.setProperty("keyed_name", "ReportChooserOptions");
reports = reports.apply();
var reportRel;

if (reports.IsError() || reports.getItemCount > 1){
   return false;
}
else{
   reportRel = reports.getRelationships("Value");
}
for (var i = 0; i < reportRel.getItemCount(); i++){
   var value = reportRel.getItemByIndex(i);
   var valueLabel = value.getProperty("label", "");

   var reportButtonSpan = document.createElement("span");
   reportButtonSpan.setAttribute("id", "MyRadio");
   reportButtonSpan.setAttribute("name", valueLabel);
   reportButtonSpan.setAttribute("style",
"position:relative;display:block;");

   var radioInput = document.createElement("input");
   radioInput.setAttribute("type", "radio");
   radioInput.setAttribute("value", valueLabel);
   radioInput.setAttribute("name", "ReportOptions");
   radioInput.setAttribute("class", "arasCheckboxOrRadio");

   var reportLabel = document.createElement("label");
   reportLabel.setAttribute("for", "MyRadio_id");

   reportButtonSpan.appendChild(radioInput);
   reportButtonSpan.appendChild(radioInput);
   reportButtonSpan.appendChild(reportLabel);
   reportButtonSpan.appendChild(document.createTextNode(valueLabel));

   sysValueSpan.appendChild(reportButtonSpan);
}
```

18. Save, unlock, and close.

©2020 Aras Corporation All Copyrights Reserved.

19. Navigate to the **Administration->Reports** off the Main Tree.
20. Open a new Report Item.
21. Set the Report Item properties as follows:
    a. **Name** = Report Chooser (this can be anything you want).
    b. **Type** = Item (You do not need the report_query template when prompted).
    c. **Location** = Client; this is required because the `ReportChooser_OpenDialog Method` returns an HTML page with JavaScript that needs to be parsed. The Client side Report only writes the transformed page to the browser window and thus the JavaScript is not parsed.
    d. **Target** = None; the windows will be created by `ReportChooser_OpenDialog`.
    e. **Method** = ReportChooser_OpenDialog
    f. **Report Query** = Leave blank; it is currently not used for this purpose
22. Save, unlock, and close the Report.
23. Navigate to the **Administration->ItemTypes** off the Main Tree.
24. Search for the ItemType for this Report.
25. Edit the ItemType and select the **Reports** Tab.
26. Add an Existing related Item and select this Report.
27. Save, unlock, and close the ItemType.

The Report should now appear on the Report menu when the Item is selected. When the user clicks **Submit the selected Report** the actual Report is requested from the Report Server via the RSGateway.

The ID for the selected Item is automatically passed as a Named Parameter np:id so you must name your Named Parameter in the SQL query with lowercase id also.

8.3 User Input for XSLT based Reports

You need user input for XSLT based Reports.

Technique

Use a Generic Report called from a client Method so that user input can be collected for the query.

This section describes the steps for building the User Input Report. This is an example and there are alternative ways to implement the user interface but this does illustrate a way to dynamically run a Report.

In this example, we use a very simple Report to get one input from the user, which is the Part Number for the item_number property for the Part ItemType to dynamically build the AML query for our Report.

The following are the steps for building an Interactive Report:

1. Create a Generic Report as a template.
    a. Set the Report properties as follows:
        - **Name** = My Report (this can be any name you want)
        - **Type** = Generic
        - **Location** = Server
        - **Target** = Window

©2020 Aras Corporation All Copyrights Reserved.

b. Construct an AML query and write the Report as usual.

c. Once the Report is written the AML query is no longer required because the Report acts as a template and the AML query is provided from the user input interactively so remove the Report Query AML from the Report item. But you may want to use the AML query string as the starting point for building the query in the Method coming up next.

2. Create a client side Method (named "My Report" for this example) that will:

a. Get the user input for the query, which is a modal dialog for this example.

b. Construct the actual AML query based on the user input. Use the AML query from the Report you created as the starting point. We replace hard-coded search criteria in the AML with variables in our Method.

c. Create an XML document to hold the AML query.

d. Get the Report item.

e. Run the Report passing the query XML document as the query criteria.

3. Create an Action to call the Method. Set the Action properties as follows:

- **Type** = Generic
- **Location** = Client
- **Target** = None
- **Method** = My Report
- **Name** = My Report

This provides the logic to get user input and to invoke the Report from an Action menu choice. The following is client side Method code to run the Interactive Report:

```
// The Report is run as a Generic Report so we can pass our own AML query
for the Report.
// Get the query criteria from the user via a modal dialog.
var params =
{
   aras:aras, innovator: this.getInnovator(),
   dialogHeight:227, dialogWidth:350,
   content: 'ReportTool/UserDialogs/MyReport.html'
};

function callback(dialogWrapper)
{
    var result = dialogWrapper.result;
    if (!result){ return; }
        // Create a query Item to hold the AML query for the Report.
        // The ItemType Part is an example and you can set this as
required.
        var qry = aras.IomInnovator.newItem('Part','get');

    // The user input from the modal dialog is serialized the into a
string.
    // The format for the serialized string is | delimited for the list of
properties
```

©2020 Aras Corporation All Copyrights Reserved.

```
    // for constructing the query and each property is a name/value pair :
delimited.
      var params = result.split('|');
      for (var i=0; i<params.length; ++i)
      {
        var param = params[i].split(':');
        qry.setProperty(param[0], param[1]);
      }

    // Get the Report item by name.
      var reportQry = aras.IomInnovator.newItem('Report','get');
      reportQry.setProperty('name', 'My Report')
      reportQry.setAttribute('select',
        'name,description,report_query,target,type,xsl_stylesheet,location
,' +
        'method(name,method_type,method_code,runas_user,runas_pwd)');
      var report = reportQry.apply();

    // Run the report passing the query Item as the query criteria for the
Report.
    // The runReport function is part of the client Toolkit API,
    // which expects node object not IOM Item objects.
      aras.runReport(report.node, '', qry.node);
}

var wnd = aras.getMainWindow();
wnd = wnd === top ? wnd.main : top;
wnd.ArasModules.Dialog.show("iframe", param).promise.then(callback);
```

4. Create the HTML page that captures the user's input for the Report. In the sample Method code above, you notice the call to open the modal dialog and load it with the ReportDialogs/MyReport.html page. This is the relative URL to the file from the Client/scripts folder for which the Methods on the server side are invoked.

   Note: If the folder Client/scripts/ReportTool/UserDialogs does not exist create it. The folder name can be anything you want; basically we are simply creating a location to put the HTML files for the Report dialogs.

   The following is the sample HTML for the MyReport.html page:

```
<html>
  <style type="text/css">
    body {
      background-color:#d4d0c8;
    }
    .header {
      background-color:#CCCCFF;
      font-family:helvetica;
      font-weight:bold;
      font-size:16pt;
      font-style: italic;
      color:#000000;
      border-width:1px;
      border-style:inset;
      padding:8px;
    }
```

©2020 Aras Corporation All Copyrights Reserved.

```
    .buttons {
      background-color:#d4d0c8;
      border-width:2px;
      border-style:groove;
      padding:10px;
    }
    td {
      font-family:helvetica;
      font-weight:bold;
      font-size:8pt;
    }
  </style>

  <script>
   var args = window.frameElement ? window.frameElement.dialogArguments
: window.dialogArguments,
   aras = (args && args.aras) ? args.aras : parent.aras;
   window.dialogArguments = args;
  function cancel()
  {
    args.dialog.result = '';
    args.dialog.close();
  }

  function apply(form)
  {
    args.dialog.result = '';

// Serialize the field values into a string.
// Delimit the fields with the | character
// and delimit the property name/value with the : character.
// i.e. prop-name:value|prop-name:value

    for (var i=0; i<form.elements.length; ++i)
    {
      var field = form.elements[i];
      if (field.type == 'text')
      {
        if (field.value)
        {
          if (args.dialog.result) args.dialog.result+= '|';
          args.dialog.result+= field.name + ':' + field.value
        }
      }
    }
    args.dialog.close();
  }

  onload = function() {
    document.body.scroll = 'no';
  }
  </script>

  <body>
    <form>
      <table border="0" cellspacing="0" cellpadding="0" width="100%">
```

©2020 Aras Corporation All Copyrights Reserved.

```
        <tr height="60">
          <td class="header" colspan="2">
            My Report
          </td>
        </tr>
        <tr>
          <td align="right" style="padding:20px 10px 0px 0px;" nowrap>
            Description:
          </td>
          <td style="padding:20px 00px 0px 0px;"><input type="text"
name="part_desc"></td>
        </tr>
        <tr>
          <td align="right" style="padding:4px 10px 20px 0px;" nowrap>
            Cost:
          </td>
          <td style="padding:4px 0px 0px 0px;"><input type="text"
name="part_cost"></td>
        </tr>
        <tr>
          <td align="center" class="buttons" colspan="2">
            <input type="button" value="OK" onclick="apply(this.form)">
            <input type="button" value="Cancel" onclick="cancel()">
          </td>
        </tr>
    </form>
  </body>
</html>
```

5. You should now be able to click on **My Report** in the Action menu and interactively run the Report.

©2020 Aras Corporation All Copyrights Reserved.