Cookbook

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

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:
    1. Name = My Report (this can be any name you want)
    2. Add a form event that fires onLoad
        - Name the Method 'My `Report_setDefaults` ' (this can be anything you want).
    3. Add fields to the form that will match the input you are requesting.
        - Name the fields appropriately ('PartNumber', 'Description', 'Cost')
    4. 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)
    5. 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;
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:
    1. Update `var reportName` with the name of your Report as deployed to the Report Server.
    2. 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
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

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.

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:
    1. Name = `ReportChooserOptions`
    2. 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:
    1. Name = Report Chooser (this can be any name you want)
    2. Add a form event that fires onLoad
        - Name the Method ‘ `ReportChooser_onLoad` ’ (this can be anything you want)
    3. Add a new Radio Button List:
        - Name this field ‘rdio1’

4. Add a Button field with a label **Submit**:
    - Create a Field Event that fires onClick.
    - Name the Method ‘ `ReportChooser_OnClick` ’ (this can be anything you want).
5. 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:

```
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:
    - 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.
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'),
   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.
    - 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");
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.
19. Navigate to the **Administration->Reports** off the Main Tree.
20. Open a new Report Item.
21. Set the Report Item properties as follows:
    - **Name** = Report Chooser (this can be anything you want).
    - **Type** = Item (You do not need the report_query template when prompted).
    - **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.
    - **Target** = None; the windows will be created by `ReportChooser_OpenDialog.`
    - **Method** = ReportChooser_OpenDialog
    - **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.

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.
    - Set the Report properties as follows:
        - **Name** = My Report (this can be any name you want)
        - **Type** = Generic
        - **Location** = Server
        - **Target** = Window
    - Construct an AML query and write the Report as usual.
    - 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:
    - Get the user input for the query, which is a modal dialog for this example.
    - 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.
    - Create an XML document to hold the AML query.
    - Get the Report item.
    - 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
    // 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.

Important

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;
}
.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%">
<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.