Sequence Tips

Contents

[Troubleshooting: 2](#_Toc499419126)

[Tips and How-To's 2](#_Toc499419127)

[How to add Excel button to Grid: 2](#_Toc499419128)

[Get BaseUrl: 2](#_Toc499419129)

[Start Workflow: 2](#_Toc499419130)

[Load latest Form: 2](#_Toc499419131)

[Example 1: 2](#_Toc499419132)

[Example 2: 3](#_Toc499419133)

[Custom Validator 4](#_Toc499419134)

[Display dynamic text on form 4](#_Toc499419135)

[FormView Control 4](#_Toc499419136)

[Disable button on Submit: 5](#_Toc499419137)

[How to configure Rest Consumer authentication if security token is in header: 5](#_Toc499419138)

[How to configure Email Listener: 5](#_Toc499419139)

[Trigger Client Side Validator 6](#_Toc499419140)

[Client-side objects: 6](#_Toc499419141)

[Client-Side APIs: 7](#_Toc499419142)

[Sample Code: 7](#_Toc499419143)

[Case I: Validate the Group: 8](#_Toc499419144)

[Case II: 9](#_Toc499419145)

[Case III: Enable/Disable Validator: 10](#_Toc499419146)

[How to add an Edit Form to a grid 11](#_Toc499419147)

[Question and answers regarding Task (Yellow/Red) indicator and Due Day: 13](#_Toc499419148)

[Workaround to change a task due date 13](#_Toc499419149)

[USING SQL AZURE DMVS TO PROFILE QUERIES 14](#_Toc499419150)

[OData Sample 14](#_Toc499419151)

[Advanced Data Source Where Clause: 16](#_Toc499419152)

[Grid - How to differentiate Insert vs Edit mode on client side 17](#_Toc499419153)

[Remove Process Wall from top bar: 17](#_Toc499419154)

[Flowtime controls 17](#_Toc499419155)

# Troubleshooting:

<http://omembers.pnmsoft.com/Online%20Help/SitePages/Troubleshooting.aspx>

# Tips and How-To's

http://omembers.pnmsoft.com/Online%20Help/SitePages/TipsAndHow-To's.aspx

# How to add Excel button to Grid:

1. Inside Grid tag: add <ExportSettings> e.g.

<ExportSettings OpenInNewWindow="true" FileName="ExportNotes" ExportOnlyData="true"></ExportSettings>

1. Inside <MasterTableView> add

<CommandItemSettings ShowExportToExcelButton="true"></CommandItemSettings>

# Get BaseUrl:

var baseUrl = '<%= {rt.HttpRequest.Url.Scheme + "://" + rt.HttpRequest.Url.Host + IIF(rt.HttpRequest.Url.IsDefaultPort = true, "", ToString(rt.HttpRequest.Url.Port) ) }%>';

# Start Workflow:

$(document).ready(function() {

var baseUrl = '<%= {rt.HttpRequest.Url.Scheme + "://" + rt.HttpRequest.Url.Host + IIF(rt.HttpRequest.Url.IsDefaultPort = true, "", ToString(rt.HttpRequest.Url.Port) ) }%>';

// alert(baseUrl);

var startMasterWorkflowUrl = baseUrl + "/\_Layouts/RunTime.aspx?workflowid=a1731536-686f-457c-8962-ef7a9b04f437&\_weShowSideBar=0&\_weShowTopBar=0&Culture=US";

$(location).href(startMasterWorkflowUrl);

});

</script>

# Load latest Form:

## Example 1:

<script type="text/javascript">

var URL\_redirect='<%={{Form Services}.Query("KPMG\_Independence\_GeneralInformation").Where(Key == "Runtimeurl").First().Value}%>';

var actid= '<%={act.activityInstanceId}%>';

var LastActivityInstanceId= '<%={try(wf.activities.where (activityName="IntakeForm" or activityName="Intake SuperAdmin Task" or activityName="F\_SuperAdmin Intake" or activityName="F\_SuperAdmin Mobile").last().ActivityInstanceId)}%>';

function RedirectMsg()

{

if (LastActivityInstanceId != actid)

{

URL\_redirect += "?culture=en-US&uiculture=en-US&ActivityInstanceId=" + LastActivityInstanceId;

document.location.href=URL\_redirect; return false;

}

}

Sys.Application.add\_load(RedirectMsg);

</script>

## Example 2:

<script type="text/javascript">

var baseUrl = '<%= {rt.HttpRequest.Url.Scheme + "://" + rt.HttpRequest.Url.Host + IIF(rt.HttpRequest.Url.IsDefaultPort = true, "", ToString(rt.HttpRequest.Url.Port) ) }%>';

var serverVariableUrl = '<%= { rt.HttpRequest.ServerVariables["URL"] } %>'

var runtimePage = (serverVariableUrl.indexOf("WFDiagramDebug.asp") > 0) ? "/Modules/WFDebug/WFDiagramDebug.aspx" : "/\_Layouts/RunTime.aspx" ;

var startMasterWorkflowUrl = baseUrl + runtimePage + "?workflowid=a1731536-686f-457c-8962-ef7a9b04f437&\_weShowSideBar=0&\_weShowTopBar=0&Culture=US";

function RedirectMsg()

{

document.location.href = startMasterWorkflowUrl;

return false;

}

Sys.Application.add\_load(RedirectMsg);

</script>

# Custom Validator

  <asp:CustomValidator ID="companyNameValidator" runat="server" ErrorMessage="Company Name is required"

                                                                               ControlToValidate="txtCompanyName" ValidateEmptyText="True"  ClientValidationFunction="validateCompanyName" />

  //Company name is only required if Customer Type is Commercial

  function validateCompanyName(sender, args){

var companyName = args.Value;

       args.IsValid = companyName =!= '';

  }

# Display dynamic text on form

<sq8:Label runat="server" Text='<%$ sq:{ Left(rt.HttpRequest.url.AbsoluteUri, IndexOf(Substring(rt.HttpRequest.url.AbsoluteUri, 8), "/")+8) } %>' ID="Label3"></sq8:Label>

# FormView Control

<%@ Page Title="" Language="C#" MasterPageFile="~/LAYOUTS/Masters/Portal.Master" CodeBehind="Default.aspx.cs" Inherits="PNMsoft.Sequence.Runtime.Web.UI.Pages.Default" AutoEventWireup="true"  %>

                <%@ Register TagPrefix="sq" TagName="FormViewerControl"

                Src="~/\_controltemplates/Flowtime/FormViewerControl.ascx" %>

<asp:Content ID="Content" ContentPlaceHolderID="content" runat="server">

    <div class="sqpt-container">

                                <sq:FormViewerControl ID="FVTest" FormVirtualPath="~/SequenceForms/wf/FVT/FVT/Form1/DefaultView.ascx" runat="server" />

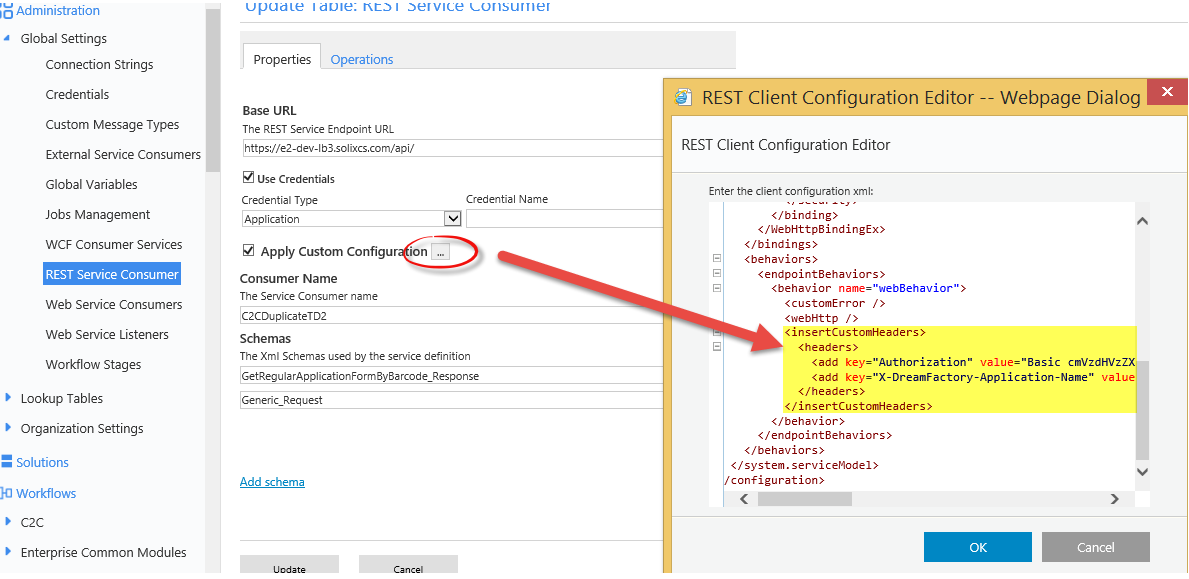
    </div>

</asp:Content>

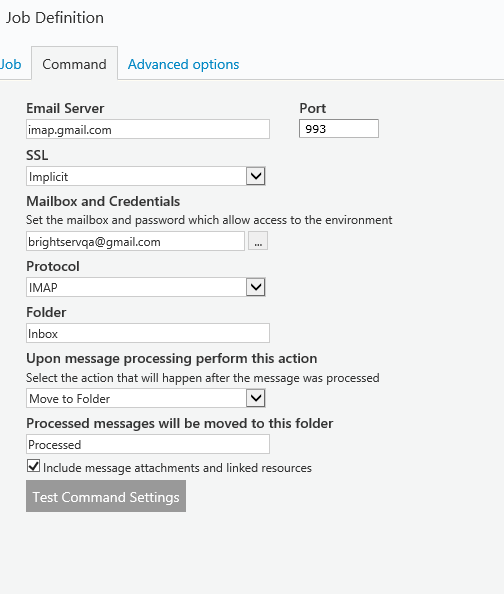
# Disable button on Submit:

function DisableButton() {  
    var btn=getElementById1('btnSubmit');   //replace the btnSubmit with the ID of the submit button.  
    btn.set\_enabled(false);  
  }  
  window.onbeforeunload = DisableButton;

# How to configure Rest Consumer authentication if security token is in header:



# How to configure Email Listener:



# Trigger Client Side Validator

<http://techbrij.com/client-side-validation-using-asp-net-validator-controls-from-javascript>

### Client-side objects:

**Page\_IsValid**  
Boolean variable  
Indicates whether the page is currently valid. The validation scripts keep this up to date at all times.

**Page\_Validators**  
Array of elements  
This is an array containing all of the validators on the page.

**Page\_ValidationActive**  
Boolean variable  
Indicates whether validation should take place. Set this variable to False to turn off validation programmatically.

**isvalid**  
Boolean property  
This is a property on each client validator indicating whether it is currently valid.

**Page\_ValidationSummaries**  
Array of elements  
This is an array containing all of the validation summaries on the page.

### Client-Side APIs:

**ValidatorValidate(val)**  
Takes a client-validator as input. Makes the validator check its input and update its display.

**ValidatorEnable(val, enable)**  
Takes a client-validator and a Boolean value. Enables or disables a client validator. Being disabled will stop it from evaluating and it will always appear valid.

**ValidatorHookupControl(control, val)**  
Takes an input HTML element and a client-validator. Modifies or creates the element’s change event so that it updates the validator when changed. This can be useful for custom validators that depend on multiple input values.

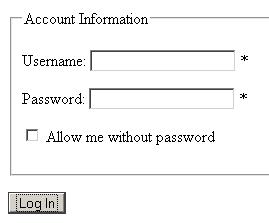
**Page\_ClientValidate(val)**  
Takes validation group as input and validate all validators of the group and returns bool.

### Sample Code:

Let us take an example to understand, consider following asp.net code:

[?](http://techbrij.com/client-side-validation-using-asp-net-validator-controls-from-javascript)

|  |
| --- |
| <div>         <fieldset class="login">             <legend>Account Information</legend>             <p>                 <asp:Label ID="UserNameLabel" runat="server" AssociatedControlID="UserName">Username:</asp:Label>                 <asp:TextBox ID="UserName" runat="server" CssClass="textEntry"></asp:TextBox>                 <asp:RequiredFieldValidator ID="UserNameRequired" runat="server" ControlToValidate="UserName"                     CssClass="failureNotification" ErrorMessage="User Name is required." ToolTip="User Name is required."                     ValidationGroup="LoginUserValidationGroup">\*</asp:RequiredFieldValidator>             </p>             <p>                 <asp:Label ID="PasswordLabel" runat="server" AssociatedControlID="Password">Password:</asp:Label>                 <asp:TextBox ID="Password" runat="server" CssClass="passwordEntry" TextMode="Password"></asp:TextBox>                 <asp:RequiredFieldValidator ID="PasswordRequired" runat="server" ControlToValidate="Password"                     CssClass="failureNotification" ErrorMessage="Password is required." ToolTip="Password is required."                     ValidationGroup="LoginUserValidationGroup">\*</asp:RequiredFieldValidator>             </p>             <p>                 <asp:CheckBox ID="AllowMe" runat="server" />                 <asp:Label ID="RememberMeLabel" runat="server" AssociatedControlID="AllowMe" CssClass="inline">Allow me without password</asp:Label>             </p>         </fieldset>         <p class="submitButton">             <asp:Button ID="LoginButton" runat="server" CommandName="Login" Text="Log In" ValidationGroup="LoginUserValidationGroup"                 OnClientClick="performCheck();" />         </p>     </div> |



### Case I: Validate the Group:

In our example, See following code to validate **LoginUserValidationGroup**:

[?](http://techbrij.com/client-side-validation-using-asp-net-validator-controls-from-javascript)

|  |
| --- |
| function performCheck() {              Page\_ClientValidate("LoginUserValidationGroup");              if (Page\_IsValid) {                  alert('it is valid');                  return true;              }              else {                  alert('No valid');                  return false;              }          } |

In other way, you can directly use **Page\_ClientValidate** method:

[?](http://techbrij.com/client-side-validation-using-asp-net-validator-controls-from-javascript)

|  |
| --- |
| function performCheck() {             if (Page\_ClientValidate("LoginUserValidationGroup")) {                  alert('it is valid');                  return true;              }              else {                  alert('No valid');                  return false;              }          } |

### Case II:

To understand other client side objects and APIs, let us validate LoginUserValidationGroup **without**using Page\_ClientValidate.

[?](http://techbrij.com/client-side-validation-using-asp-net-validator-controls-from-javascript)

|  |
| --- |
| function performCheck() {              if (checkValidationGroup("LoginUserValidationGroup")) {                  alert('it is valid');                  return true;              }              else {                  alert('No valid');                  return false;              }          }            function checkValidationGroup(valGrp) {              var rtnVal = true;              for (i = 0; i < Page\_Validators.length; i++) {                  if (Page\_Validators[i].validationGroup == valGrp) {                      ValidatorValidate(Page\_Validators[i]);                      if (!Page\_Validators[i].isvalid) { //at least one is not valid.                          rtnVal = false;                          break; //exit for-loop, we are done.                      }                  }              }              return rtnVal;          } |

In **checkValidationGroup**method, first validation group of all validators on page is checked. If it belongs to given group, it is validated. If it is not valid, the method will return false.

### Case III: Enable/Disable Validator:

Suppose you have to disable password requiredfield validator If **‘Allow me’** checkbox is true. See following code:

[?](http://techbrij.com/client-side-validation-using-asp-net-validator-controls-from-javascript)

|  |
| --- |
| function performCheck() {              if (document.getElementById('<%=AllowMe.ClientID%>').checked) {                ValidatorEnable(document.getElementById('<%=PasswordRequired.ClientID%>'), false);            }              if (Page\_ClientValidate("LoginUserValidationGroup")) {                alert('it is valid');                return true;            }            else {                alert('No valid');                return false;            }        } |

# How to add an Edit Form to a grid

<http://kb.pnmsoft.com/help/add-an-edit-form-to-a-grid>

|  |
| --- |
| 1. Add a grid to your form. |
| 2. Select ***MasterTableViewEditForm***.  <http://dyzz9obi78pm5.cloudfront.net/app/image/id/596b05368e121c5971af9e37/n/add-an-edit-form-to-a-grid-1.png> |
| 3. Add controls to the form, as you normally would, and bind the controls to a data source.  <http://dyzz9obi78pm5.cloudfront.net/app/image/id/596b074cec161cc3731e98ce/n/add-an-edit-form-to-a-grid-3.png> |
| 4. To save data, the command name on the save button must be ***PerformUpsert***.  <http://dyzz9obi78pm5.cloudfront.net/app/image/id/596b0796ec161c3a6f1e994c/n/add-an-edit-form-to-a-grid-2.png> |
| 5. The form markup should look like the example below.   * Set *EditFormType* to *Template*. |

<**sq:Grid** runat="server" ID="Grid2">

<**MasterTableView**>

<**RowIndicatorColumn** Visible="True">

<**HeaderStyle** Width="20px"></**HeaderStyle**>

</**RowIndicatorColumn**>

<**ExpandCollapseColumn** Visible="True">

<**HeaderStyle** Width="20px"></**HeaderStyle**>

</**ExpandCollapseColumn**>

<**EditFormSettings** EditFormType="Template"><**FormTemplate**>

<**sq:Label** runat="server" Text="Label" ID="Label1"></**sq:Label**>

<**sq:TextBox** runat="server" ID="TextBox1"></**sq:TextBox**>

<**sq:Button** runat="server" Text="Button" ID="Button1" CommandName="PerformUpsert"></**sq:Button**>

</**FormTemplate**>

</**EditFormSettings**>

</**MasterTableView**>

</**sq:Grid**>

# Question and answers regarding Task (Yellow/Red) indicator and Due Day:

The due date stoplight icon, that comes off-the-shelf, has a single setting for the entire environment and the resolution is days.   
Meaning, you can set how many days before the actual due date, green changes to yellow. And this setting affects all tasks in all workflows.   
  
The requirement you had described calls for a custom stop light which requires development.   
Also, keep in mind that the grids are not "self refreshing", so if users refresh the page once an hour, they will miss many of the of color changes.   
  
Regarding the dynamic "due date", this depends on how you want to implement the due date requirement.   
The product has a due date functionality, and it can be set dynamically (with expression) and the minimal unit is "hours" (meaning the minimal due date is 1 hour). Once the due date has been set to a task, it cannot be changed.   
From the examples you provided, it seems that a different "due date" functionality is needed, so consider to implement a due date reminder with workflow timers and then you have the flexibility to set it however you want.

# Workaround to change a task due date

However, there's a workaround for this - use an SQL stored procedure to set a new value to the due date field in tblActionItems.   
  
Example:   
CREATE PROCEDURE spWFA\_UpdateIActDueDate (@iActId BIGINT, @newDueDate DATETIME)   
AS   
BEGIN   
DECLARE @completiondate DATETIME, @currDueDate DATETIME   
SELECT @completiondate = fldCompletionDate,   
@currDueDate = fldDueDate   
FROM tblActionItems WITH (NOLOCK)   
WHERE fldIActId=@iActId   
  
IF @completiondate IS NOT NULL   
OR @currDueDate < GETDATE()   
RETURN   
  
UPDATE tblActionItems   
SET fldDueDate=@newDueDate   
WHERE fldIActId=@iActId   
END

# USING SQL AZURE DMVS TO PROFILE QUERIES

|  |
| --- |
| SELECT query\_stats.query\_hash AS "Query Hash", |
|  | SUM(query\_stats.total\_worker\_time) / SUM(query\_stats.execution\_count) AS "Avg CPU Time", |
|  | MIN(query\_stats.statement\_text) AS "Statement Text" |
|  | FROM |
|  | (SELECT QS.\*, |
|  | SUBSTRING(ST.text, (QS.statement\_start\_offset/2) + 1, |
|  | ((CASE statement\_end\_offset |
|  | WHEN -1 THEN DATALENGTH(ST.text) |
|  | ELSE QS.statement\_end\_offset END |
|  | - QS.statement\_start\_offset)/2) + 1) AS statement\_text |
|  | FROM sys.dm\_exec\_query\_stats AS QS |
|  | CROSS APPLY sys.dm\_exec\_sql\_text(QS.sql\_handle) as ST) as query\_stats |
|  | GROUP BY query\_stats.query\_hash |
|  | ORDER BY 2 DESC;  SELECT QS.creation\_time,  SUBSTRING(ST.text, (QS.statement\_start\_offset/2) + 1,  ((CASE statement\_end\_offset  WHEN -1 THEN DATALENGTH(ST.text)  ELSE QS.statement\_end\_offset END  - QS.statement\_start\_offset)/2) + 1) AS statement\_text  FROM sys.dm\_exec\_query\_stats AS QS  CROSS APPLY sys.dm\_exec\_sql\_text(QS.sql\_handle) as ST  order by QS.creation\_time desc |

# OData Sample

1. Enable OData Service in Advanced Options
2. Add ODataDataSource to form. E.g. <sq:ODataEntityDataSource runat="server" ID="ODataScripting" ></sq:ODataEntityDataSource>
3. Include the stored procedure or table in Data Model
4. Construct the service URL and parameters and do Ajax call as follow:
5. Make sure to use X-SqXsrfToken in header if it’s enabled
6. Make sure to enclose string parameter in single quotes
7. Make sure to append “l” after number for bigint. E.g. “123l”

/\*\*\*\*\* ========= GetServiceBaseUrl function starts ============ \*\*\*\*/

function GetServiceBaseUrl(dataModalQuery){

var webServiceUrl = $findByControlId("ODataDS").get\_serviceUrl() + dataModalQuery;

webServiceUrl = window.location.protocol+ '//' + window.location.host + webServiceUrl.substr(webServiceUrl.indexOf("/sequenceservices"),webServiceUrl.length)

return webServiceUrl;

}

/\*\*\*\*\* ========= GetServiceBaseUrl function ends ============ \*\*\*\*/

/\*\*\*\*\* ========= PopulateUserPhoneAndEmail function starts ============ \*\*\*\*/

function PopulateUserPhoneAndEmail(userType, id, phoneTextBodyId, emailTextBodyId){

var url;

if (userType == "adjustor")

url = adjustorsServiceBaseUrl + "(" + id + ")"; //Getting a record from a table

else

url = userInfoServiceBaseUrl + "?userId=" + id; //Getting a record from SP

//console.log("Ajax URL:'" + url + "'");

$sq.ajax({

type: "get",

contentType: "application/json; charset=utf-8",

dataType: "json",

url: url,

headers: {"X-SqXsrfToken": $sq("#\_\_SqXsrfTokenValue").val() },

success: function (data, textStatus, XmlHttpRequest) {

console.log(data);

if (data && data.d){

if (userType == "adjustor") { //Returns an object

$findByControlId(phoneTextBodyId).set\_value(data.d.Phone);

$findByControlId(emailTextBodyId).set\_value(data.d.Email);

}

else { //Returns an array

$findByControlId(phoneTextBodyId).set\_value(data.d[0].fldBusinessPhone);

$findByControlId(emailTextBodyId).set\_value(data.d[0].fldEmail);

}

}

},

error: function (data, textStatus, XmlHttpRequest1) {

alert("Error retrieving user info");

console.log(data);

console.log(textStatus);

console.log(XmlHttpRequest1);

}

});

}

# Advanced Data Source Where Clause:

<sq8:DataSource runat="server" ID="refns0NotesDataSource"

QueryName="ref:ns0/Notes" OrderBy="CreationDate desc" Where="AudienceCode == @Public || AudienceCode == @Vendor || @IsVendor == False">

<WhereParameters>

<sq:ExpressionParameter Expression="&quot;PUBLIC&quot;" Name="Public"></sq:ExpressionParameter>

<sq:ExpressionParameter Expression="&quot;VENDOR&quot;" Name="Vendor"></sq:ExpressionParameter>

<sq:ExpressionParameter Expression="Join(MemberOf(rt, rt.currentuser.userId), &quot;,&quot;).contains(&quot;Vendors&quot;)" Name="IsVendor"></sq:ExpressionParameter>

</WhereParameters>

</sq8:DataSource>

# Grid - How to differentiate Insert vs Edit mode on client side

There is an onCommand attribute in <ClientSettings><ClientEvents /> </ClientSettings><element

function OnCommand(sender, args)

{

if (args.get\_commandName() == "Edit")

{

alert("Edit");

args.set\_cancel(true); //cancel the command

}

if (args.get\_commandName() == "InitInsert")

{

alert("Insert");

}

}

# Remove Process Wall from top bar:

$(document).ready(function() {

$("table.weTopBarLinksContainer td:contains('Process Wall')").text("");

});

# Flowtime controls

----------------------------------------

-Inbox Summary (InboxSummaryControl.ascx)

-Message Grid (UserMessagesGridControl.ascx)

-Group Message Grid (GroupMessageGridControl.ascx)

-My Delegation (DelegationGridControl.ascx)

-Process Grid: Display all the process instanceses of a specific process (filtered by the user permissions) (ProcessGridControl.ascx)

--------------------------

-My Processes: Display all process instances that were created by a specific user (UserInstancesGridControl.ascx???)

-Process I Own: Display the processes that the user own (What does own means???) (ProcessIOwnControl.ascx)

-Process I Started: A summary bar which shows the number of processes the user started (ProcessesIStartedControl.ascx)

-Process List: Displays all available processes that the user has permissions to view/manage (ProcessIntancesGridControl.ascx???)

-Start New Process: Enables a user to select and start a new processes (NewProcessControl.ascx)

-Visual Web Part: Displays a visual created using Sequence Analytics (VisualControl.ascx)

Form Viwers: (FormViewControl.ascx)

Forms are for viewing only (not for user inputs) and don't support CRUD operations. The embeded form doesn't have a workflow/activity instance

Support Rest Basic Authentication:

Sequence REST Service support NTLM by default, but can be configured as Basic authentication or token based authentication. In order to configure it as Basic authentication, change WebHttpsBindings and WebHttpBinding security mode to “Transport” and change Transfort clientCredentialType to from Windows to Transport

<bindings>

<webHttpBinding>

<binding name="WebHttpBinding">

<security mode="TransportCredentialOnly">

<transport clientCredentialType=”Basic "></transport>

</security>

</binding>

<binding name="WebHttpsBinding">

<security mode="Transport">

<transport clientCredentialType="Basic"></transport>

</security>

</binding>

</webHttpBinding>

</bindings>

# XPath Expression

Expression Description

nodename Selects all nodes with the name "nodename"

/ Selects from the root node

// Selects nodes in the document from the current node that match the selection no matter where they are

. Selects the current node

.. Selects the parent of the current node

@ Selects attributes

## XPath Samples

<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book>

<title lang="en">Harry Potter</title>

<price>29.99</price>

</book>

<book>

<title lang="en">Learning XML</title>

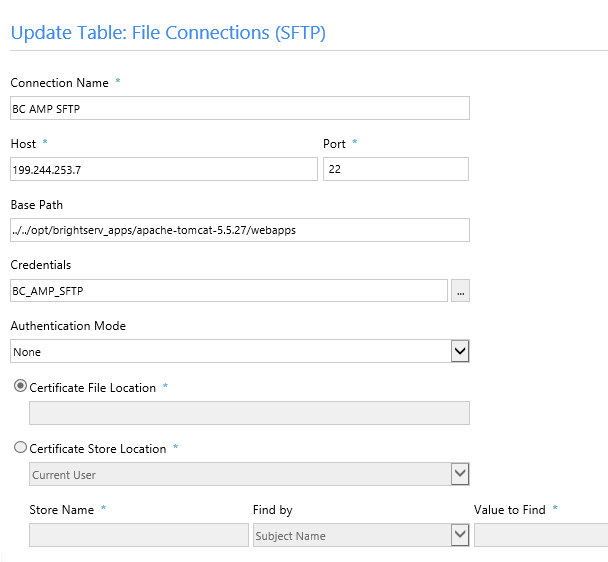
<price>39.95</price>

</book>

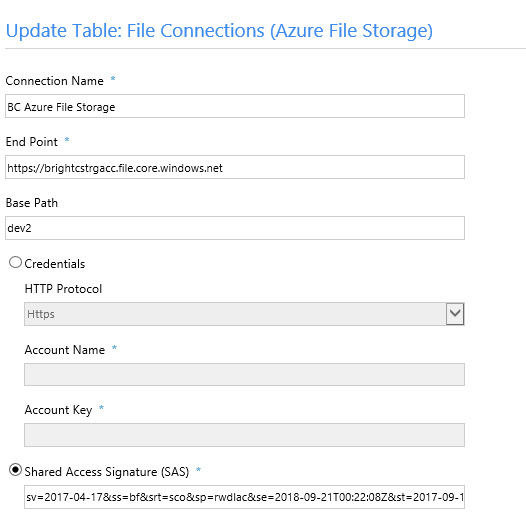
</bookstore>

# Get File Configuration:

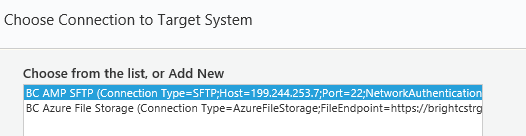
Step 1: Create a file connection in Global Settings



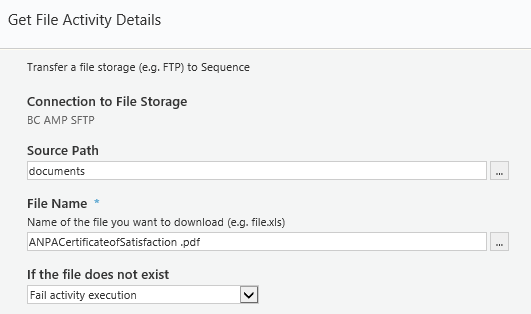
Or



## Step 2: Configure Get File activity:



Step 2: Configure the Get File Activity



# Configure Dashboard

## Configure HotOperation TaskList

1. Copy a Service.Config and make changes and rename to “Service.BC\_ManualTasks.Config”
2. Modify TasksList.aspx. For Manual Tasks solution, change the ConfigName for TeamLeaderTasksGridControl to “BC\_ManualTasks”
3. Modify Service.BC\_ManualTasks.Config to use different  ServiceMetaData file for the parts that need customization such as “ServiceMetadata="~/shared resources/Components/HotOperations/Flowtime/Config/ServiceMetadata/TeamMemberTasksServiceMetadata\_BC\_ManualTasks.config"
4. Modify  ServiceMetadata config such as

“TeamMemberTasksServiceMetadata\_BC\_ManualTasks.config”                                                                                                                                                                                                            

# Add Javasript file link:

<sq:scriptcontrol runat="server" src="~/Shared Resources/js/ALG Price Match/PriceMatchQuote.js" type="text/javascript"/>

# Change the task message template:

Program Files -> PNMSoft -> Backgroup Runtime Service -> Message.xltls

# Hot Operation Tables

USL[Solution GUID]\_Cases -- Stores cases created in HotOperations, and indicates if there was manual allocation.

USL[Solution GUID]\_AllocatedCases --Stores manual allocation of cases.

USL[Solution GUID]\_Tasks --Stores Dynamic Tasks properties such as subject due date and others.

USL[Solution GUID]\_AllocatedTasks --Stores Dynamic Tasks allocation (replicated from the case) and manual assignment type.

USL[Solution GUID]\_TaskEmployeeAssignments --Stores manual assignment by employee name.

USL[Solution GUID]\_TaskRoleAssignments -- Stores manual assignment by role.

select \* from USL7eb607f387ed4c90b4ae6244401f5ee3\_Cases where fldMasterIWfId = 463586

select \* from USL7eb607f387ed4c90b4ae6244401f5ee3\_AllocatedCases where fldMasterIWfId = 463586

select \* from USL7eb607f387ed4c90b4ae6244401f5ee3\_Tasks where fldIActId = 7446341 --- !!!

select \* from USL7eb607f387ed4c90b4ae6244401f5ee3\_AllocatedTasks

select \* from USL7eb607f387ed4c90b4ae6244401f5ee3\_TaskEmployeeAssignments

select \* from USL7eb607f387ed4c90b4ae6244401f5ee3\_TaskRoleAssignments

select \* from USL7eb607f387ed4c90b4ae6244401f5ee3\_Tasks where fldIActId = 7446341

select \* from tblInstanceActivities where fldId = 7446341

select \* from UACT184b3b569e544672bb4d3e2a3329eab8 assignmemnt order by fldId desc --

Notes:

--- Not feteched: tblInstanceActivties: fldStatus = 2, fldEmployeeCode = 0; fldCompletionDate = null

-- USL7eb607f387ed4c90b4ae6244401f5ee3\_Tasks:

fldFetched = null; fldStatus = 0

Feteched:

tblInstanceActivties: fldStatus = 2, fldEmployeeCode > 0; fldCompletionDate = null

USL7eb607f387ed4c90b4ae6244401f5ee3\_Tasks: fldFetched = null (why?); fldStatus = 1

Completed:

tblInstanceActivties: fldCompletionDate update to the actual completion date

USL7eb607f387ed4c90b4ae6244401f5ee3\_Tasks: no change

- If a HO task is fetched by someone else, the non-team lead cannot see teh task in her/his queue

- If a HO is assigned (assigned is different than fetched) to another user or another role , only the team-lead and assigned user can see

# Rollback Tips

exec sp\_GetIWfInfo 267696

-- abort the subworkflow 262020

begin  tran update tblInstanceWorkflows

set           fldStatus=7

where  fldid=267728

rollback

commit

-- update variables

begin  tran update UWF22F135C5A0AA44CC8E3A8AC1023FA925

set           UserAction='DISPATCH\_SUBMIT'

where  fldIWfId=267696

-- delete from switch UACT

begin  tran delete

from   UACTdb6b7ae2f38a4859b636a702ebc17bb6

where  fldIWfId=267696

-- rollback the workflow instance

declare @iwf bigint = 267696

declare @iact\_RollbackTo bigint = 4061595

begin  tran delete

from   tblInstanceActivities

where  fldInstanceWfId = @iwf

              and fldId > @iact\_RollbackTo

update tblInstanceActivities

set           fldStatus=3

              , fldCompletionDate=null

              , fldRedirectFlag=1

where  fldId = @iact\_RollbackTo

              and fldInstanceWfId=@iwf

update tblInstanceWorkflows

set           fldLastUpdated=getdate()

where  fldId=@iwf

rollback

commit

# How to increate Maximum number of activity executions

The change is in the BRS config file if the execution is from the brs, or the JES config.   
  
<workflowExecution redirectMode="Automatic" instanceExecutionLimit="2000” />   
  
Need to add instanceExecutionLimit="2000” or any other number