

CREATE A DASHBOARD USING INFOPATH ON O365

To stay on top of things, you'll want to have a tool which shows quick stats to get intel on your business process at a single glance.

In this lab, you will create an InfoPath form that will serve as a dashboard showing essential data, much like what reporting does. The Dashboard will display all existing Expense Reports and allow the user to filter the results displayed.

REQUIREMENTS

This lab requires the following:

- SharePoint Environment (O365 or On-premise) – This lab uses O365
- Expense Report Library (from previous lab)
- InfoPath (2010 or higher) – This lab uses InfoPath 2013
- [qRules](#)
- [FormsViewer](#)

This guide already comes with a starter form pre-injected with qRules.

ADD RULES AND DATA CONNECTIONS THAT RETRIEVE SUBMITTED FORMS

When the Dashboard opens, we would like it to display a list of all Expense Reports in the form library; then we'll add a filter section with logic to filter out queried data.

The Dashboard will pull items from your Expense library using a standard SharePoint library or list data connection.

1. In InfoPath Designer, open the starter template included in this lab – named "Expense Report Dashboard – START.xsn".
2. Go to Data > Data Connections and click "Add" to add a new data connection.
 - a. Select "Create a new data connection to...Receive data" then click Next

This wizard helps you specify a data connection to an external data source.

- ☒ Create a new connection to:
- ☐ Submit data
- ☒ Receive data

- b. Select SharePoint library or list and click Next again

Select the source of your data.

From where do you want to receive your data?

☐ SOAP Web service

☐ REST Web service

☒ SharePoint library or list

☐ Database (Microsoft SQL Server only)

☐ XML document

- c. Enter your site URL, then again click Next

SharePoint site details

Enter the location of the SharePoint site:

Example: http://server/site

- d. Select the Expense Report form library you created earlier, then click Next again

The SharePoint site that you selected has the following lists and libraries. Select a list or library:

Select a list or library:	Description of list or library
<u>Expense Reports Library</u>	Form Library

- e. Put a check mark next to the promoted properties shown below, then click Next twice

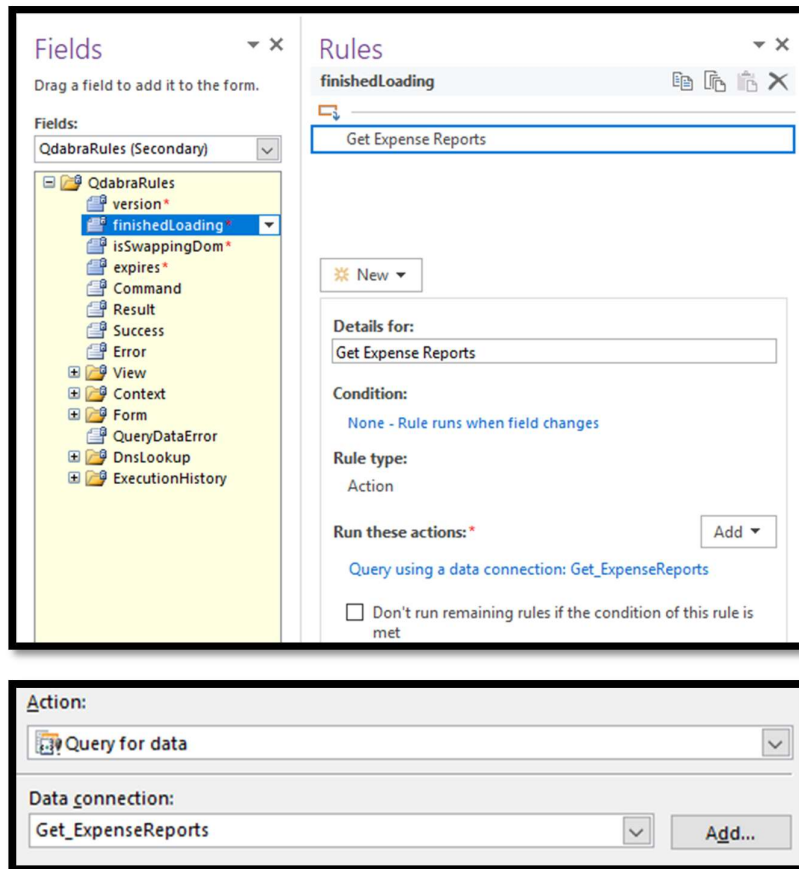
☒	Title
☒	Name
☒	Total
☒	Purpose
☒	Manager_Name (Manager Name)
☒	Status
☒	Expense_Date (Expense Date)

- f. Give your data connection a meaningful name, uncheck the box next to "Automatically retrieve data when form is opened, and click Finish

Enter a name for this data connection:

☐ Automatically retrieve data when form is opened

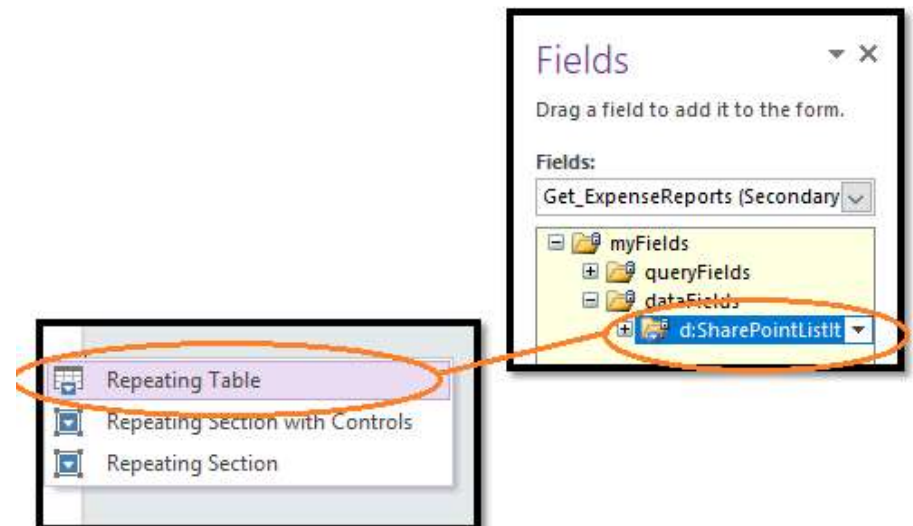
3. Go to **QdabraRules > finishedLoading** node and add a new action rule called "Get Existing Reports". Add a Query for data action to this rule, and select the data connection you just created in the dropdown that appears.



Ideally, on-load queries should be filtered – this prevents forms from querying hundreds of rows which greatly affects form performance.

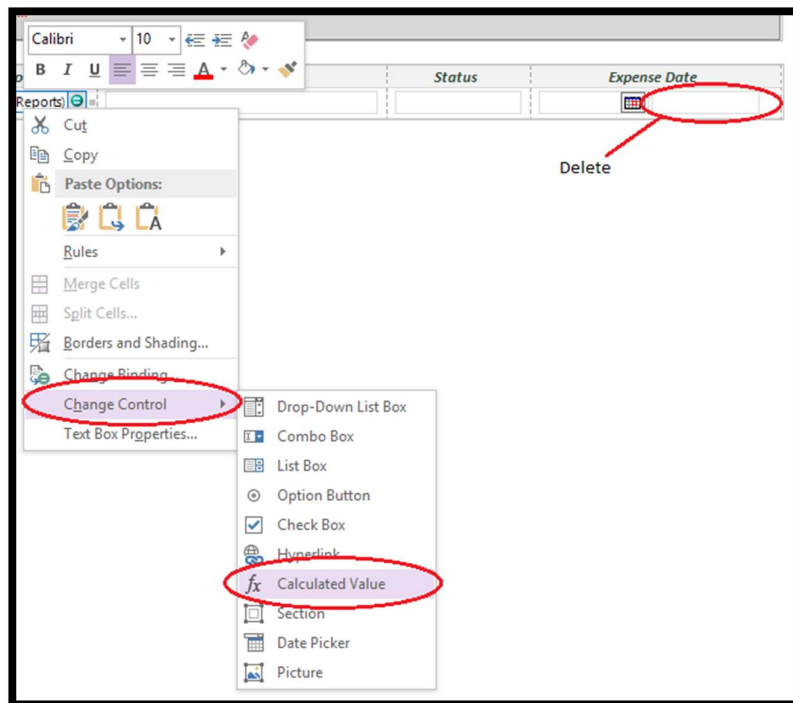
BUILD THE DASHBOARD DISPLAY

1. Now we need to display the Expense Reports on the dashboard.
 - a. Select the data connection you just created from the dropdown at the top of the Fields pane, then expand the dataFields folder and select the **d:SharePointListItem_RW** repeating group. Drag this group onto the design pane and select Repeating Table from the pop-up menu that appears.

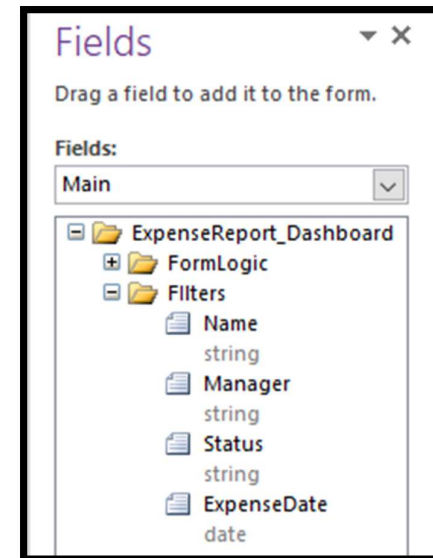


- b. Delete the ID column that appears to the right of the repeating table you just created, and then format the header as you see fit.

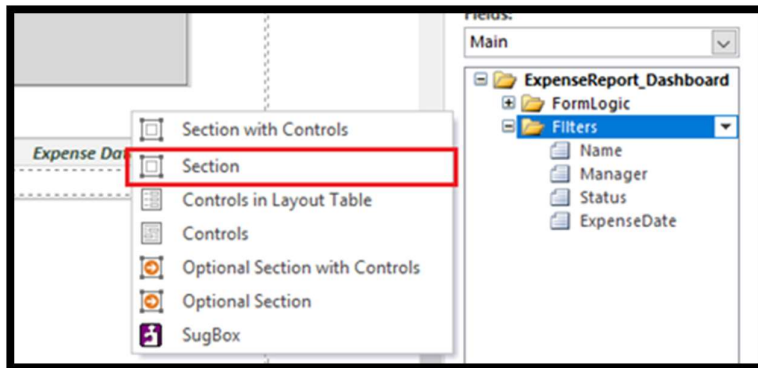
2. Delete the time component of the Date/Time controls that were inserted by default, and then right-click each remaining control and change to calculated values (so that users can't edit them).



3. Now let's add the filters to the dashboard – the starter form comes with a pre-created section and fields for the filter logic.



- a. Drag the Filters group to the canvas, just right above the Results table, and display as a section.



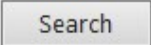
- b. Insert a table inside the Filter section (under Insert tab, insert a Custom Table) and then drag and display the filter fields as desired. Your filter section may look similar to this:

Since we are querying data from SharePoint, using dropdown controls are ideal in order to perform an exact match search – a regular SharePoint retrieve/query data connection does not support partial match search.

- c. Right click on the **Status** dropdown field and click properties. At the list box choices, select **Enter choices manually**, and type in the following options: Approved, Paid, Rejected, and Submitted.

Value	Display Name	Default
Approved	Approved	Yes
Paid	Paid	Yes
Rejected	Rejected	Yes
Submitted	Submitted	Yes

- d. (**OPTIONAL**): In the event that you have SharePoint lists which contains the list of employees and managers, feel free to use these data connections to populate the Employee and Manager dropdown controls.
- e. For purposes of training, you may manually enter Employee and Manager values for the respective dropdown filter controls.

4. Underneath the last column, add a  button.

5. Click the **Search** button.

6. Under the *Home* tab, click **Manage Rules** – this should display a control panel for the rules located at the right side of your screen.

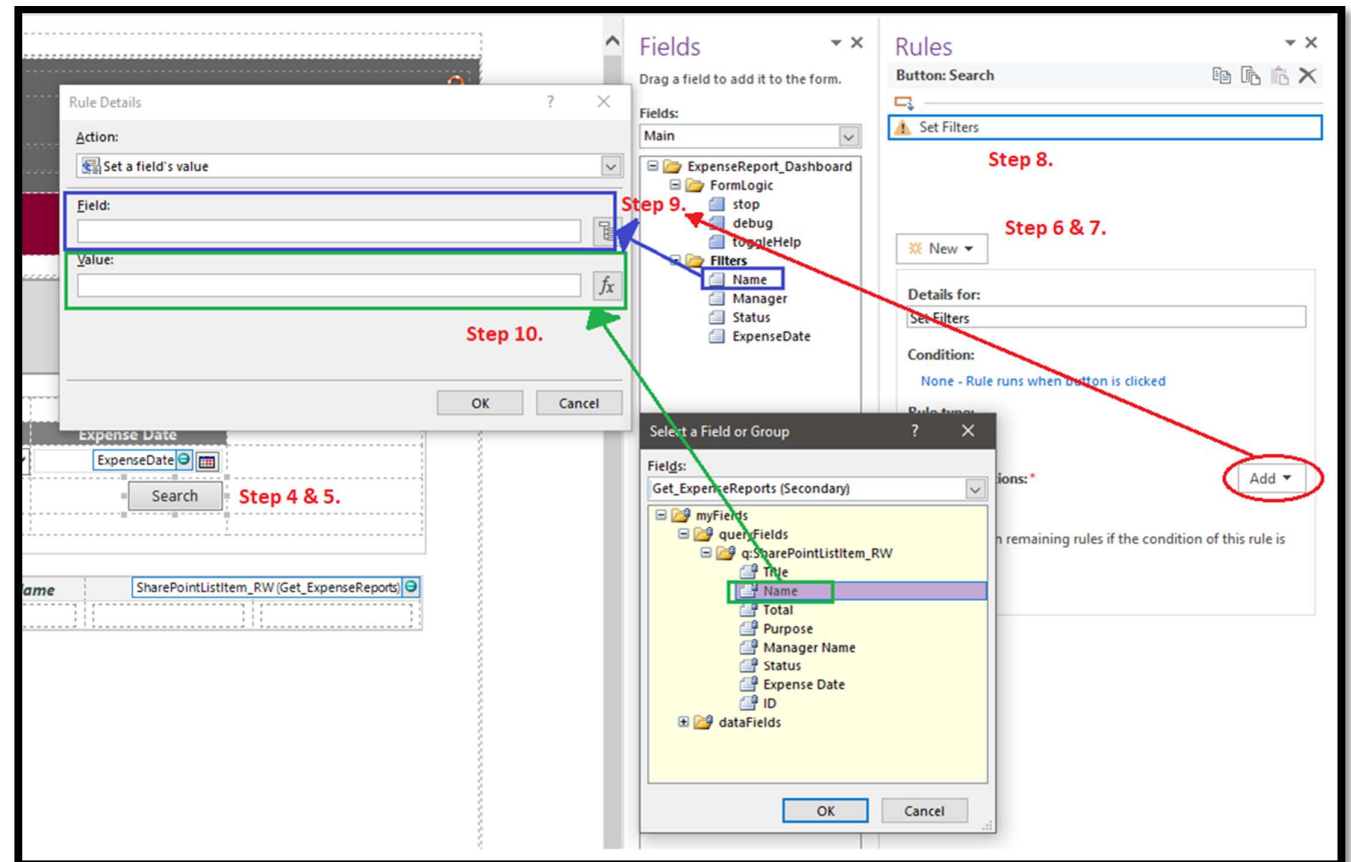
7. Add an **Action** rule.

8. Provide a meaningful name for this rule such as **Set Filters**.

9. Under this rule, add an action that will **Set a field's value**.

10. For the Target field, select the **Name** node from the *Get_ExpenseReports* secondary data source.

11. Set the **Value** to the **Name** filed filter from the Main data source.

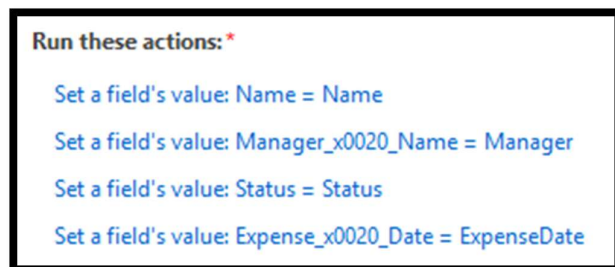


The screenshot displays the QDABRA software interface with several windows and panels illustrating the steps for creating a rule:

- Step 4 & 5:** A **Search** button is added to the form.
- Step 6 & 7:** The **Manage Rules** panel is open, showing a rule named **Set Filters** with the condition **None - Rule runs when button is clicked**.
- Step 8:** The **Rule Details** window is open, showing the action **Set a field's value**.
- Step 9:** The **Field** dropdown in the **Rule Details** window is set to **Name** from the **Filters** group.
- Step 10:** The **Value** dropdown in the **Rule Details** window is set to **Name** from the **Get_ExpenseReports (Secondary)** data source.
- Step 11:** The **Value** dropdown in the **Rule Details** window is set to **Name** from the **Main** data source.

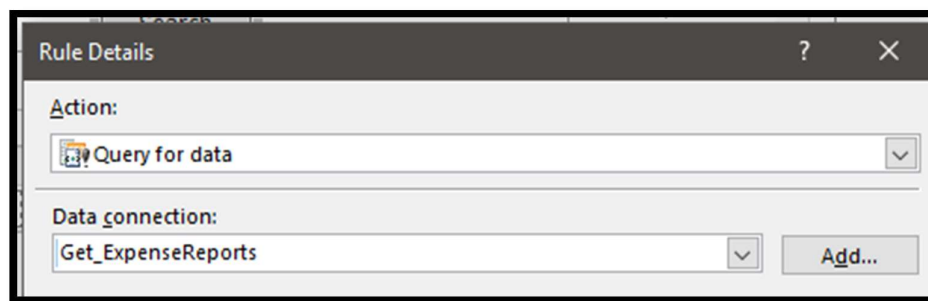
12. Repeat steps for the rest of filter fields from the Main data source's Filter section.

Here's how your actions should look like:



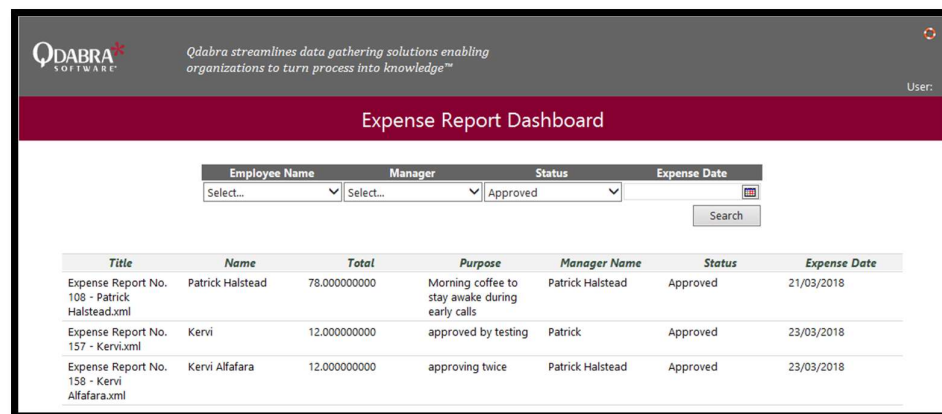
13. Add another *Action* rule that will **Query for data**.

14. Select the **Get_ExpenseReports** data connection.



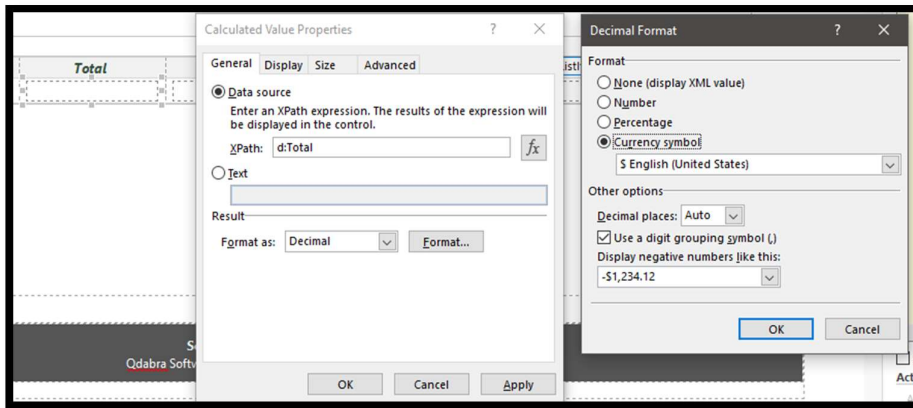
Run the form in Preview by clicking the preview icon located along the set of icons at the upper left portion of the InfoPath form – or just simply click **F5**.

Verify that the form loads the submitted forms from the library and test your filter queries:



Notice that some fields may poorly display returned values.

You may change these fields to use the proper data type and text format:



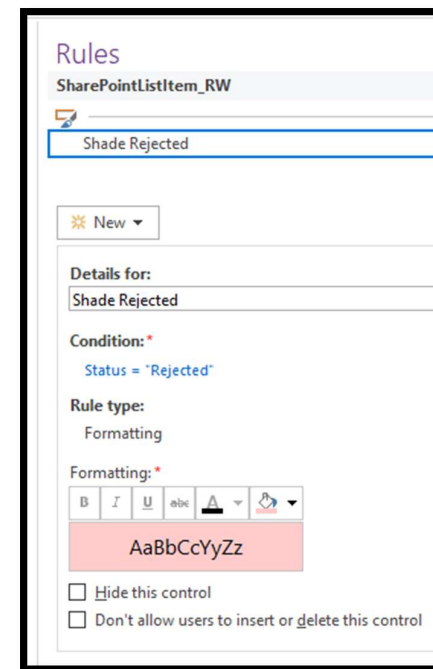
Adjust column width and text formatting as desired to make your table easier to read:

Title	Name	Total	Purpose	Manager Name	Status	Expense Date
Expense Report No. - Jo-Anne Alfafara	Jo-Anne Alfafara	\$1.00	test	Jo-Anne Alfafara	Submitted	26/03/2018
Expense Report No. 159 - Kervi Alfafara	Kervi Alfafara	\$32.00	rejecting	Patrick Halstead	Rejected	23/03/2018
Expense Report No. 160 - Kervi Alfafara	Kervi Alfafara	\$5.00	rejecting again	Patrick Halstead	Rejected	23/03/2018
Expense Report No. 155 - Kervi Alfafara	Kervi Alfafara	\$11.00	testing	Kervi Alfafara	Paid	23/03/2018
Expense Report No. 156 - Kervi	Kervi	\$23.00	approved by	Patrick	Paid	23/03/2018
Expense Report No. 108 - Patrick Halstead	Patrick Halstead	\$78.00	Morning coffee to stay awake during early calls	Patrick Halstead	Approved	21/03/2018
Expense Report No. 157 - Kervi	Kervi	\$12.00	approved by testing	Patrick	Approved	23/03/2018
Expense Report No. 158 - Kervi Alfafara	Kervi Alfafara	\$12.00	approving twice	Patrick Halstead	Approved	23/03/2018

STATUS INDICATORS

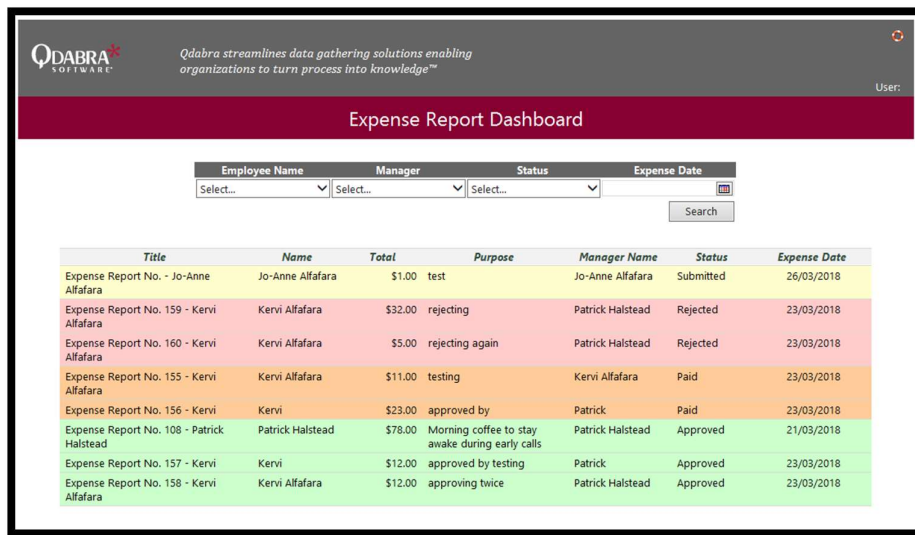
Next, we will add some conditional formatting to highlight the rows based on the status of each Expense Report: **yellow** for Submitted, **orange** for Paid, **green** for Approved and **red** for Rejected.

1. Select the repeating section on the dashboard and add a formatting rule in the Rules pane. Set condition Status (main data source) is equal to "Rejected", and select a reddish background color below.



2. Use the above steps to create a similar formatting rule for other Statuses.

Notice that a simple heat map modification makes data a lot easier to understand at a single glance:



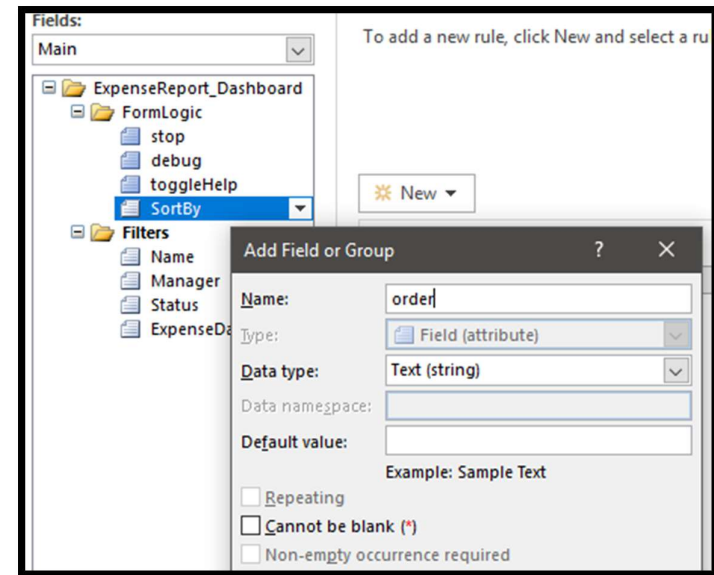
The screenshot shows the 'Expense Report Dashboard' with a search bar and a table of expense reports. The table uses a color-coded heat map for the 'Status' column: yellow for 'Submitted', red for 'Rejected', orange for 'Paid', and green for 'Approved'.

Title	Name	Total	Purpose	Manager Name	Status	Expense Date
Expense Report No. - Jo-Anne Alfara	Jo-Anne Alfara	\$1.00	test	Jo-Anne Alfara	Submitted	26/03/2018
Expense Report No. 159 - Kervi Alfara	Kervi Alfara	\$32.00	rejecting	Patrick Halstead	Rejected	23/03/2018
Expense Report No. 160 - Kervi Alfara	Kervi Alfara	\$5.00	rejecting again	Patrick Halstead	Rejected	23/03/2018
Expense Report No. 155 - Kervi Alfara	Kervi Alfara	\$11.00	testing	Kervi Alfara	Paid	23/03/2018
Expense Report No. 156 - Kervi	Kervi	\$23.00	approved by	Patrick	Paid	23/03/2018
Expense Report No. 108 - Patrick Halstead	Patrick Halstead	\$78.00	Morning coffee to stay awake during early calls	Patrick Halstead	Approved	21/03/2018
Expense Report No. 157 - Kervi	Kervi	\$12.00	approved by testing	Patrick	Approved	23/03/2018
Expense Report No. 158 - Kervi Alfara	Kervi Alfara	\$12.00	approving twice	Patrick Halstead	Approved	23/03/2018

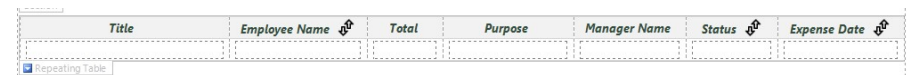
TABLE SORTING

In this section we will use the power of [gRules](#) (**SortTable** command) to sort out columns in ascending or descending order upon querying data.

1. Under the FormLogic section, *add* a node called **SortBy**.
2. Add an attribute field to the SortBy node called **order** – right click on the **SortBy** field and click Add.



3. Add a sort button  to any of the columns that you would like to sort



The table shows the same data as before, but with small sort buttons (up and down arrows) added to the 'Employee Name', 'Status', and 'Expense Date' columns.

Title	Employee Name 	Total	Purpose	Manager Name	Status 	Expense Date 
Expense Report No. 159 - Kervi Alfara	Kervi Alfara	\$32.00	rejecting	Patrick Halstead	Rejected	23/03/2018
Expense Report No. 160 - Kervi Alfara	Kervi Alfara	\$5.00	rejecting again	Patrick Halstead	Rejected	23/03/2018
Expense Report No. 155 - Kervi Alfara	Kervi Alfara	\$11.00	testing	Kervi Alfara	Paid	23/03/2018
Expense Report No. 156 - Kervi	Kervi	\$23.00	approved by	Patrick	Paid	23/03/2018
Expense Report No. 108 - Patrick Halstead	Patrick Halstead	\$78.00	Morning coffee to stay awake during early calls	Patrick Halstead	Approved	21/03/2018
Expense Report No. 157 - Kervi	Kervi	\$12.00	approved by testing	Patrick	Approved	23/03/2018
Expense Report No. 158 - Kervi Alfara	Kervi Alfara	\$12.00	approving twice	Patrick Halstead	Approved	23/03/2018

Feel free to add more Sort By buttons if desired.

Before we proceed, let's study the command first. Here's a sample command syntax for the **SortTable** command:

SortTable /row=my:myFields/my:MyGroup /compare=my:id /order=DESC
/dsname=Query_Library

Where: **SortTable** is the qRules command we are calling

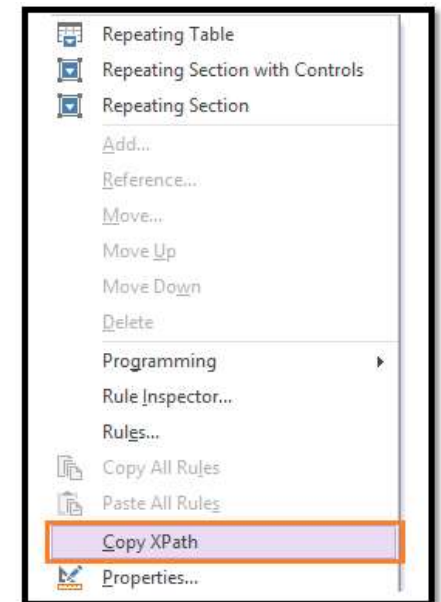
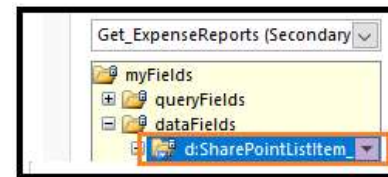
Parameters:

- **/row**: relative path to the repeating row to sort on
- **/compare**: relative path to the field to sort on. Must be a node within the repeating row.
- **/order** (optional): can be ASC or DESC, not case sensitive. Assumes ASC if not specified.
- **/dsname** (optional): name of the data source. Assumes you are using the Main data source if it is not specified.
- **/type** (optional): can be number or text. Note that if the incorrect types are used, the rule will not execute. Assumed text if not specified.

Let's build the command:

4. Grab the xpath to the repeating node of the **Get_ExpenseReports** data connection (right click on the repeating group and select Copy XPath):

/dfs:myFields/dfs:dataFields/d:SharePointListItem_RW



5. Get the relative path for the nodes to be compared (similar to the step above, but just the node along with the name-space prefix – in this case **d:**)

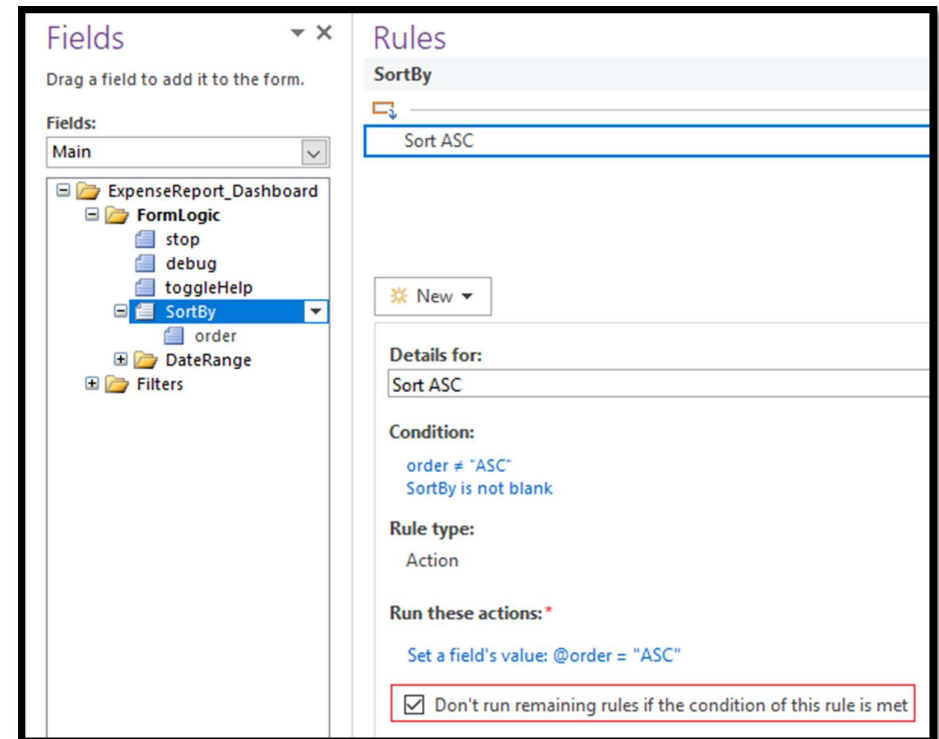
- **d:Status**
- **d:Name**
- **d:Expense_x0020_Date**

Now that we have the relative xpaths, our command should look similar to this:

**SortTable /row=/dfs:myFields/dfs:dataFields/d:SharePointListItem_RW
/compare=d:Status /order=ASC /dsname=Get_ExpenseReports**

Let's add the sort logic

6. At the *SortBy* field, add 2 action rules that toggles the **order** field (ASC and DESC)
 - a. Sort ASC
 - i. Set the value of the order field to the value **ASC**
 - ii. Add a condition to only run this rule if order is not equal to **ASC** and **SortyBy** is not blank
 - iii. **Tick the checkbox to stop running other rules if the condition is met**



- b. Sort DESC
 - i. Set the value of the order field to value **DESC**
 - ii. Add a condition to only run this rule if order is not equal to **DESC** and **SortyBy** is not blank

No need to tick the checkbox to stop other rules - since there are no other rules

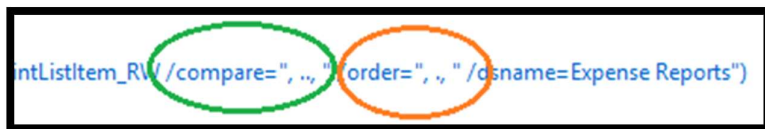
7. At the **@order** field, add a rule called Sort Table
8. Set this rule's target field to the **QdabraRules** data connection's **Command** node.
9. For the value, click the **fx** button and enter the formula:

```
concat("SortTable /row=/dfs:myFields/dfs:dataFields/d:SharePointListItem_RW  
/compare=", SortBy, " /order=", @order, " /dsname=Get_ExpenseReports")
```

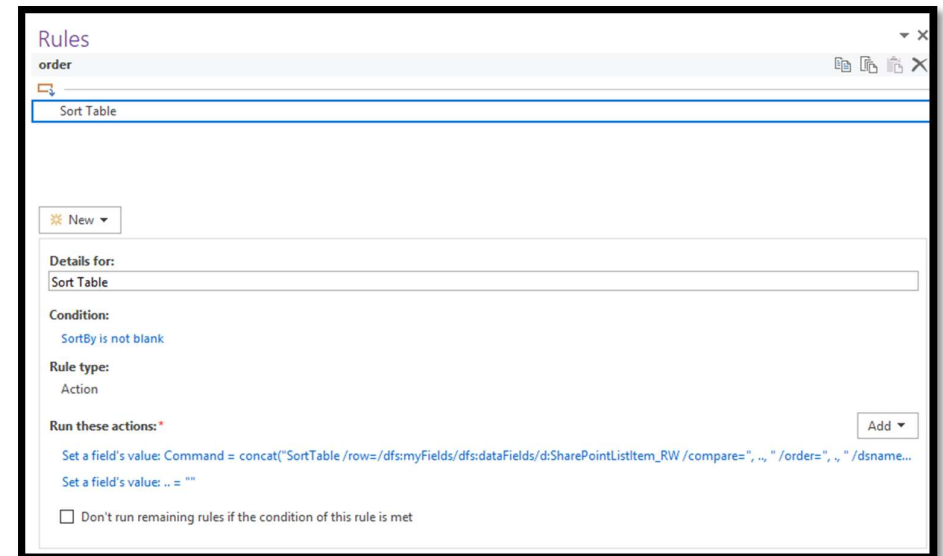
10. Replace **SortBy** and **@order** with the corresponding nodes from the FormLogic group:



The **SortBy** and **@order** field should be replaced with the corresponding xpath elements.



11. Right after the **SortTable** action, add an action that resets the **SortBy** field to a blank value after the **SortTable** command is called
12. Add a condition to only run this rule if **SortBy** is not blank



After adding the centralized sort logic, let us now configure the sort buttons to trigger the **SortTable** command.

13. On the first Sort button (Employee Name column), add a rule called **Sort By**.
14. Set this rule's action to set the **SortBy** field's value **d:Name** – see step 5.

15. Perform the same steps for the other Sort button controls – or copy the rule over to the other buttons but change the **SortBy** target value to the current column the button belongs.
For *status* use **d:Status** – while the sort button for the *expense date* should use the value **d:Expense_x0020_Date**.

Although at first this approach may seem to be a little or very complex, it is an awesome centralized approach which requires fewer steps or set of rules for sorting out repeating tables – no need to add the SortTable command on each of the sort buttons.

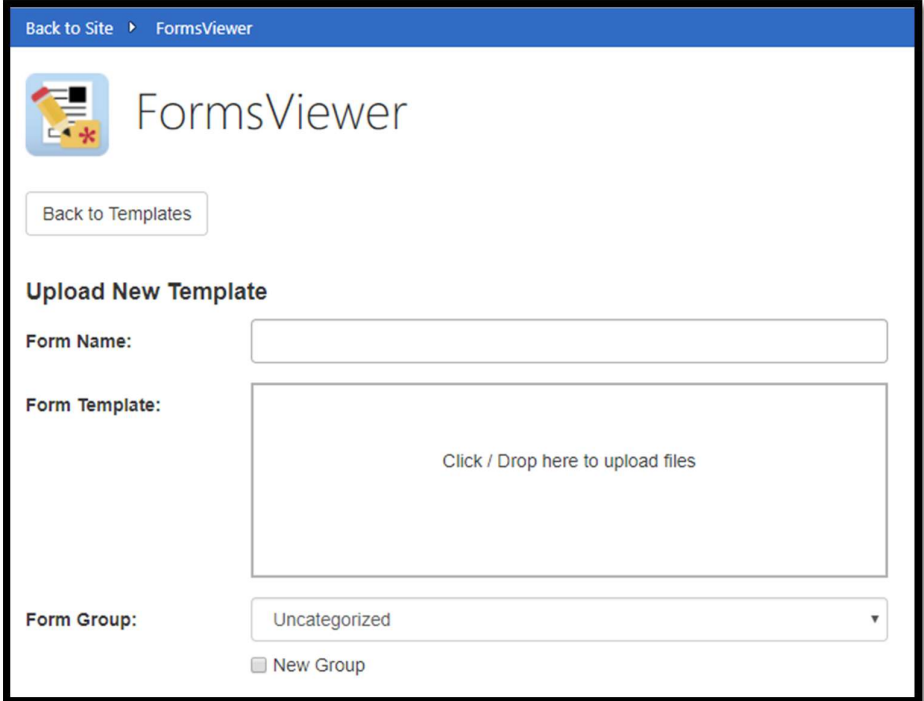
Our form is now ready for FormsViewer testing, but before we proceed, you may want perform one more preview to check if all rules and logic are running properly.

A hidden debug section is placed below the footer section. Click the Solution Version toggle button to view qRules' execution history:

Solution Version: 1.0.0.39 Qdabra Software Copyright © 2006-2018			
Command	Result	Success	Error
SortTable /row=/dfs:myFields/dfs:dataFields/dShare		✓	
SortTable /row=/dfs:myFields/dfs:dataFields/dShare		✓	
SortTable /row=/dfs:myFields/dfs:dataFields/dShare		✓	

UPLOAD TO FORMSVIEWER

1. When done testing, proceed to your **FormsViewer** site > click **Upload New Template**.



2. Provide a meaningful **FormName** such as **Expense Reports** Library
3. Click or Drag and drop the XSN onto the Form template Section
4. **(Optional)** Select a group if desired.
5. Save your template.

Click the Open Form button to view your form in the browser:

Title	Employee Name	Total	Purpose	Manager Name	Status	Expense Date
Expense Report No. 159 - Kervi Alfara	Kervi Alfara	\$32.00	rejecting	Patrick Halstead	Rejected	3/23/2018
Expense Report No. 160 - Kervi Alfara	Kervi Alfara	\$5.00	rejecting again	Patrick Halstead	Rejected	3/23/2018
Expense Report No. 155 - Kervi Alfara	Kervi Alfara	\$11.00	testing	Kervi Alfara	Paid	3/23/2018
Expense Report No. 158 - Kervi Alfara	Kervi Alfara	\$12.00	approving twice	Patrick Halstead	Approved	3/23/2018

DATE RANGE FILTER

One of SharePoint's regular data connection's limitation is its ability to perform a date range query. In order to display a select range of dates, we will use InfoPath's formatting rule to hide the rows as a workaround for this limitation.

1. At the right side of the Expense Date column, after the **SortBy** button, add a new button  (date filter) for toggling the date range section and controls.

2. Inside the **FormLogic** group, add the a new group called **DateRange** and add following and fields inside:

Where:

- **toggleDateRange**: string/text data type (attribute)
- **StartDate**: date data type
- **EndDate**: date data type

3. In between the filter section and the result table, drag and drop the DateRange group and display it as a section.
4. Add a condition to hide this section if the **@my:toggleDateRange** field is not equal to **true**.

5. Inside this hidden section, display the start and end date filter fields as date picker controls.

6. Click the Expense Report result table at the lower left section
 and add a **formatting rule** called *Hide if less than Start Date*.
7. Under this rule, add a condition to hide the row if the **d:Expense_x0020_Date** node is less than the **my:StartDate** field.
8. Click the **And** button to insert another condition.
9. For the second condition first dropdown, select the **my:StartDate** field, and then select “is not blank” for the second dropdown control.

This condition implies that rows that have the Expense Date lower than the specified StartDate (filter field) will be hidden from the view – and we only want this condition to be followed if the StartDate field is not blank.

10. Check the Hide this control check box at the bottom most portion of the formatting rule.
11. Move this rule up towards the top – before the Status Indicator rules

12. Add another formatting rule for the **my:EndDate** field by following the similar steps above, but use the greater than condition this time.

Condition

Run the rule when this condition is true:

Expense Date is greater than EndDate

EndDate is not blank

OK

13. This will not work yet since InfoPath is trying to be funny, but that's not true. It is hiding rows that are equal to the selected **EndDate** value. We will have to make a minor adjustment to the condition and move **EndDate** one day ahead.
14. Click the third dropdown of the first row and set this to **use a formula...**

Condition

Run the rule when this condition is true:

Expense Date is greater than EndDate

EndDate is not blank

Use a formula...

OK Cancel

15. Highlight on the **EndDate** field and click on **Insert Function** button.

16. Select the addDays function – automatically, this will enclose the EndDate field with the addDays function. After the EndDate field, add a comma and the value “1” in order to add one more than to the current EndDate selection.

Insert Formula

Formula:

addDays(EndDate, 1)

Your condition should look similar to this:

Shade Approved

New

Details for:

Hide if greater than End Date

Condition:

Expense Date > addDays(EndDate, 1) and EndDate is not blank

Rule type:

Formatting

Formatting:

(hidden)

☒ Hide this control

17. Add an action rule that will set the **@my:toggleDateRange** to the value:

`not(@toggleDateRange = "true")`

This formula above is a neat approach for toggling fields to true or false, instead of setting the actual value to true and/or false which uses two or more rules. Just one rule with a cool formula is enough for toggling fields.

18. Lastly, add rules which will clear out the Start Date and End Date fields when the Date range button is clicked – this should clear out unwanted date range filters when a new batch of forms are being queried.

Run these actions: *

Set a field's value: `@toggleDateRange = not(@toggleDateRange = "true")`

Set a field's value: `StartDate = ""`

Set a field's value: `EndDate = ""`

☐ Don't run remaining rules if the condition of this rule is met

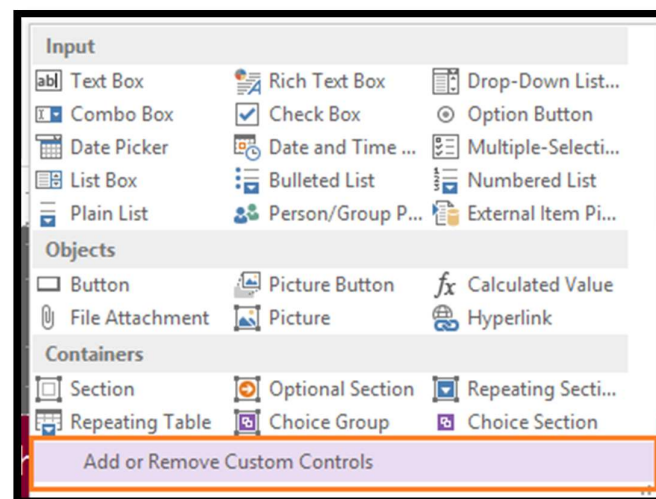
When done, upload to FormsViewer and test your form out!

PAGINATION

Paginating repeating data in InfoPath forms is definitely awesome but can be quite intimidating at first. To make things easier, we have created a re-usable [XTP template part](#) which already contains most of the logic needed for pagination.

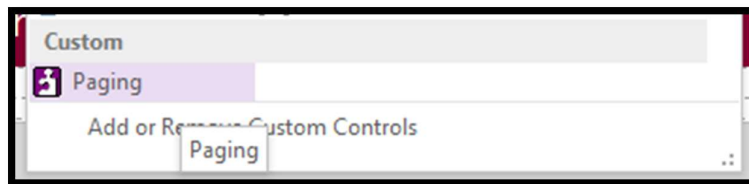
Note that this type of pagination approach is best suited for paginating less than a hundred rows (yes there are other techniques for querying and paginating large amounts of data). Paginating a hundred or more rows significantly affects a forms performance.

To add the XTP to your form, open up the Dashboard in design mode, expand the *controls* section located under the *Home* tab, and add the **Paging.xtp2** as a *Template Part* - custom control.



To add the pagination controls, follow the steps below:

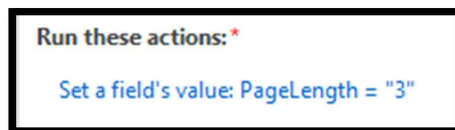
1. Place your cursor underneath the repeating table (results table) or where ever you desire to display the pagination controls buttons.
2. Back at the Custom Controls section, click on the Paging custom control.



This should add the Paging section onto your canvas:

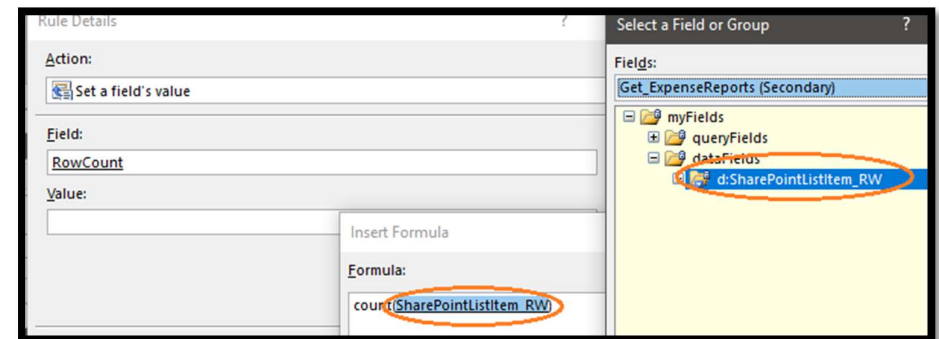


3. Follow the steps inside the red box:
 - a. Modify the [Set Page Length \(Edit\)](#) rule to set the desired number of rows per page.



If you only have a few submitted forms, set the default page length number to a low value for now for testing purposes – you may edit this later on.

- b. Modify the [Set Row Count \(Edit\)](#) rule to determine the count the number of items returned by the query (items in the repeating group)

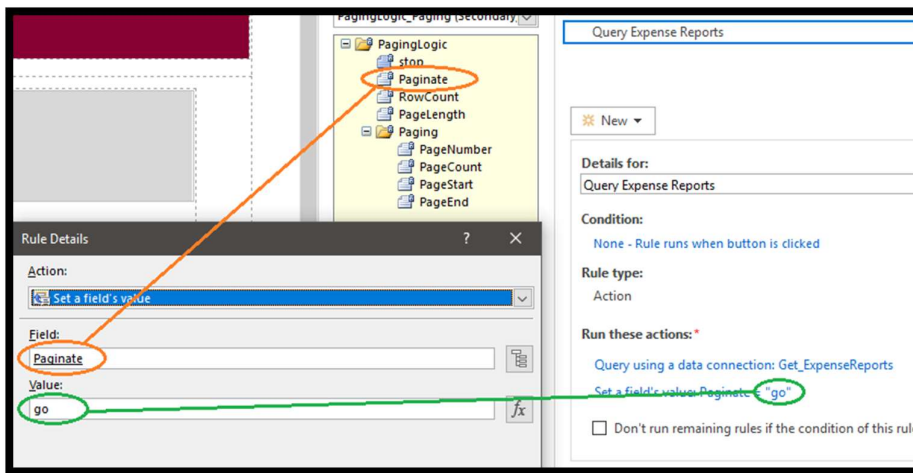


- c. Follow the rest of the steps.

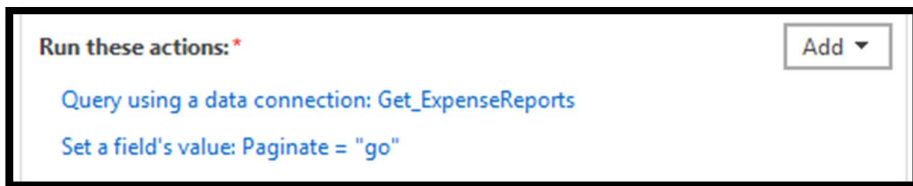
When done, we'll configure the Dashboard's query logic to trigger the Pagination node whenever a query is being performed.

4. Click the Search button of the Dashboard.

- Under the Query rule, add an action that sets **Paging XTP's** the **Paginate** node's value to go.



- Add the same action to qRules' finishedLoading node's query action.



Before we proceed, let's first study these formulas below:

$number(count(preceding-sibling::RepeatingGroup) + 1) < number(xdXDocument:GetDOM("PagingLogic_Paging")/PagingLogic/Paging/PageStart)$

$number(count(preceding-sibling::RepeatingGroup) + 1) > number(xdXDocument:GetDOM("PagingLogic_Paging")/PagingLogic/Paging/PageEnd)$

These formulas use the count preceding-sibling xpath function which determines the row number for each row.

It also uses the **PageStart** and **PageEnd** nodes which the Paging XTP predetermines once the **PageNumber** node of the template part is populated.

If the current row number is less than the **PageStart** or greater than the **PageEnd**, the formula above will hide the row.

The next thing we need to determine is the **RepeatingGroup**, which we have previously identified as **d:SharePointListItem_RW**.

Take note of the name-space **d**: whenever adding this hide/show condition – name spaces should always be included.

7. Replace the repeating group with the appropriate group name

Your formulas may look similar to these two:

a. Hide Previous

```
number(count(preceding-sibling:: d:SharePointListItem_RW) + 1) <
number(xdXDocument:GetDOM("PagingLogic_Paging")/PagingLogic/Pagin
g/PageStart)
```

b. Hide Next

```
number(count(preceding-sibling:: d:SharePointListItem_RW) + 1) >
number(xdXDocument:GetDOM("PagingLogic_Paging")/PagingLogic/Pagin
g/PageEnd)
```

Paste and copy from note pad to remove unnecessary formatting.

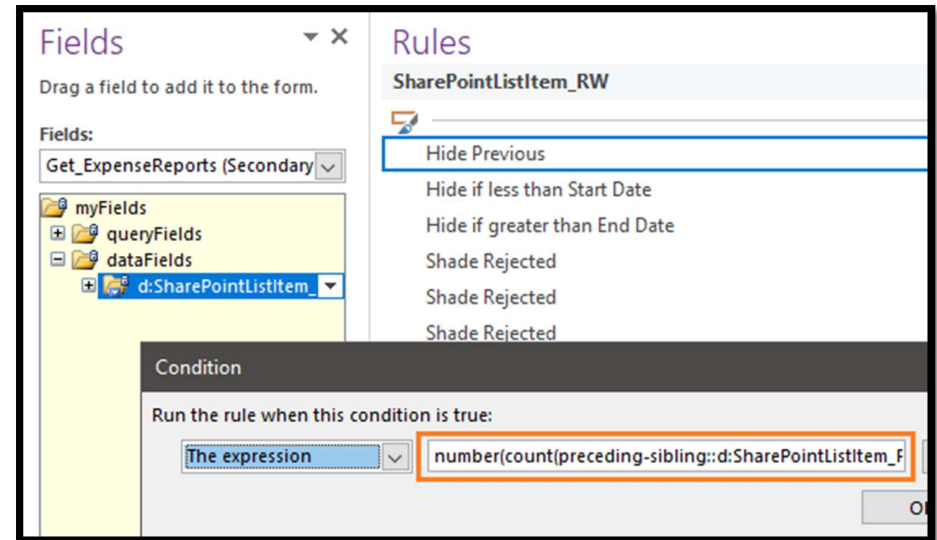
8. Click the Expense Report result table at the lower left section

 and add a *formatting rule* named **Hide Previous** and move this rule to the top most level.

9. Insert a condition.

10. For the first dropdown control, select **The expression**.

11. Paste the formula into the free text field at the right side.



12. Click OK.
13. Click the checkbox to hide the control.
14. Perform the same steps for the **Hide Next** formatting rule.

Test your form in preview! Navigate through the pages and test filtering – the paging controls should be hidden if the number of returned data is lower than the max page length.

FormsViewer_Test

Secure
https://qfs.formsquo.com/FormsViewer/View?SPHostUrl=https%3A%2F%2Fqdabra.sharepoint.com%2FConsulting&SPPlan...


Qdabra streamlines data gathering solutions enabling organizations to turn process into knowledge™

User: don.lambatin

Expense Report Dashboard

Employee Name	Manager	Status	Expense Date
		Select...	

Search

Title	Employee Name	Total	Purpose	Manager Name	Status	Expense Date
Expense Report No. 155 - Kervi Alfafara	Kervi Alfafara	\$11.00	testing	Kervi Alfafara	Paid	3/23/2018
Expense Report No. 156 - Kervi	Kervi	\$23.00	approved by	Patrick	Paid	3/23/2018
Expense Report No. 108 - Patrick Halstead	Patrick Halstead	\$78.00	Morning coffee to stay awake during early calls	Patrick Halstead	Approved	3/21/2018

2
of 3

Solution Version: 1.0.0.27
Qdabra Software Copyright © 2006-2018

BONUS: DEBUGGING

Forms fail most of the time, and having a debug section comes very handy when figuring out why things aren't working properly. At the footer, the Solution Version text is actually a button used for toggling the debug section.

You may display the hidden fields which run all the logic behind the form.

You can also display the Repeating table to debug hide/show, highlighting, and other formatting rules.

The qRules' Command node – you may display this field if you want to run qRules command manually.

The qRules' Execution History shows the list of commands that have been called – here you can determine which rules are successful and which ones are failing. This section is definitely a must have section whenever you are developing your forms.

Solution Version: 1.0.0.29
 Qdabra Software Copyright © 2006-2018

Debug Section:

Form Logic:

Stop:

Debug: true

Toggle Help:

Order: ASC

Sort By: d:Expense_x0020_Date

Date Range:

Toggle Date Range:

Start Date:

End Date:

Filters:

Name:

Manager:

Status:

Expense Date:

Expense Reports:

Title	Name	Total	Purpose	Manager Name	Status	Expense Date	ID
Expense Report No. - Jo-Anne	Jo-Anne Alfafara	1.000000000	test	Jo-Anne Alfafara	Submitted	26/03/2018	249
Expense Report No. 159 - Ker	Kervi Alfafara	32.000000000...	rejecting	Patrick Halstead	Rejected	23/03/2018	247
Expense Report No. 160 - Ker	Kervi Alfafara	5.000000000	rejecting again	Patrick Halstead	Rejected	23/03/2018	248
Expense Report No. 155 - Ker	Kervi Alfafara	11.000000000...	testing	Kervi Alfafara	Paid	23/03/2018	243
Expense Report No. 156 - Ker	Kervi	23.000000000...	approved by	Patrick	Paid	23/03/2018	244
Expense Report No. 157 - Ker	Kervi	12.000000000...	approved by testing	Patrick	Approved	23/03/2018	245
Expense Report No. 158 - Ker	Kervi Alfafara	12.000000000...	approving twice	Patrick Halstead	Approved	23/03/2018	246
Expense Report No. 108 - Pat	Patrick Halstead	78.000000000...	Morning coffee to stay awake during ...	Patrick Halstead	Approved	21/03/2018	196

qRules Command:

Command	Result	Success	Error
SortTable /row=/dfs.myFields/dfs.dataFields/d:SharePo		☒	
SortTable /row=/dfs.myFields/dfs.dataFields/d:SharePo		☐	Parameter compare has no value.
SortTable /row=/dfs.myFields/dfs.dataFields/d:SharePo		☒	

CONTACT INFORMATION

Qdabra Software
218 Main Street,
Suite 731,
Kirkland, WA 98033

Phone: 877.544.2389
Email: Support@Qdabra.com
Website: <http://www.qdabra.com>
Community: <http://www.infopathdev.com>