

S - cubed Commenting Extension

A seamless way to save data in Qlik Sense app

Configure Object

After **Import** the extension from **Qlik Management Console** (QMC), s-formed commenting extension will be available from **Extension List**

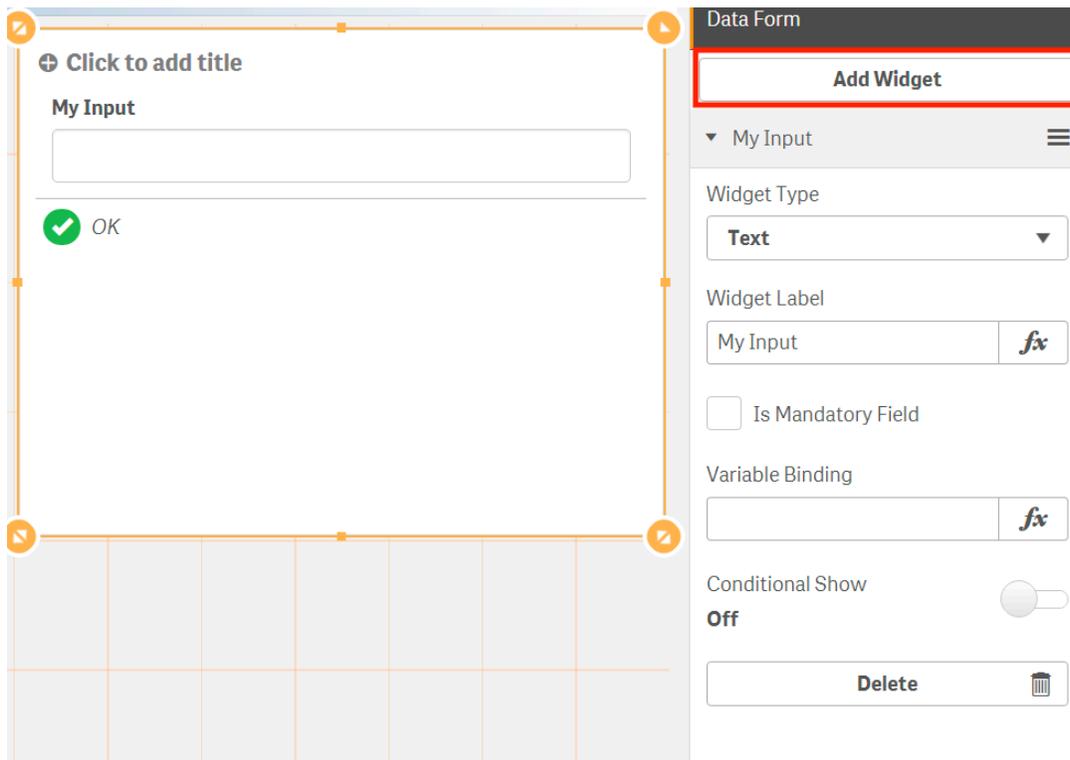
The screenshot shows the Qlik Sense interface. On the left is a navigation pane with categories: Charts, Custom objects, Master items, and Fields. The 'Extensions' list is visible, with 'S-formed' highlighted in red. The main area shows a preview of the 'S-formed' extension, which is a data form for Qlik Sense. The preview includes a 'Commenting' header, a 'Find' search bar, a table of 'Case ID' and 'Status', and a 'Case To Report' table.

Case ID	Status
6673931	Duplicated
6818514	Invalid
6886760	Invalid
6903581	New
6984145	New

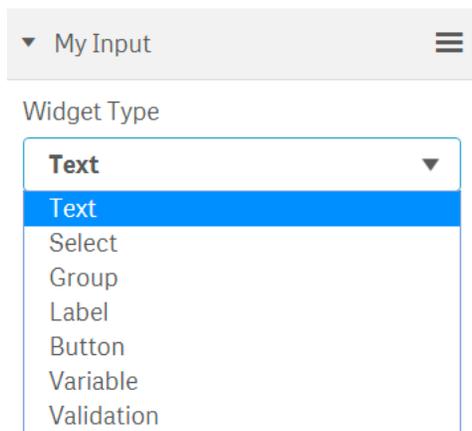
Case ID	Status	Cases.C
6673931	Duplicated	Test01
6818514	Invalid	HIHI2
6886760	Invalid	
6903581	New	HIHI4
6984145	New	-

S-formed
Data form extension for Qlik Sense

After adding the extension to your sheet, you will be able to configure the object by simply adding **Widgets** in **Property Panel**



By giving a certain **Widget Type**, you can decide in what kind of iteration way should this widget be rendered and configured.



1. Text

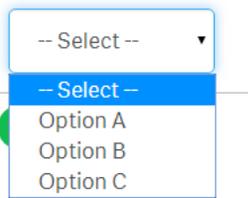
Plain Text input, mainly for free text commenting purposes

My Input

2. Select

A dropdown menu will be rendered, mainly for scoped options

My Input



-- Select --
-- Select --
Option A
Option B
Option C

The options can be configured in **Options** property, values need to be split by semicolons. **Qlik Expression** is supported, so by using **concat()** function, you can dynamically present options according to data field or variables

Options (Split By Semicolon)

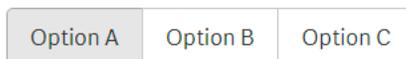


Option A;Option B;Option C *fx*

3. Group

Similar to Select, but rendering all values with group buttons, mainly used for data flags

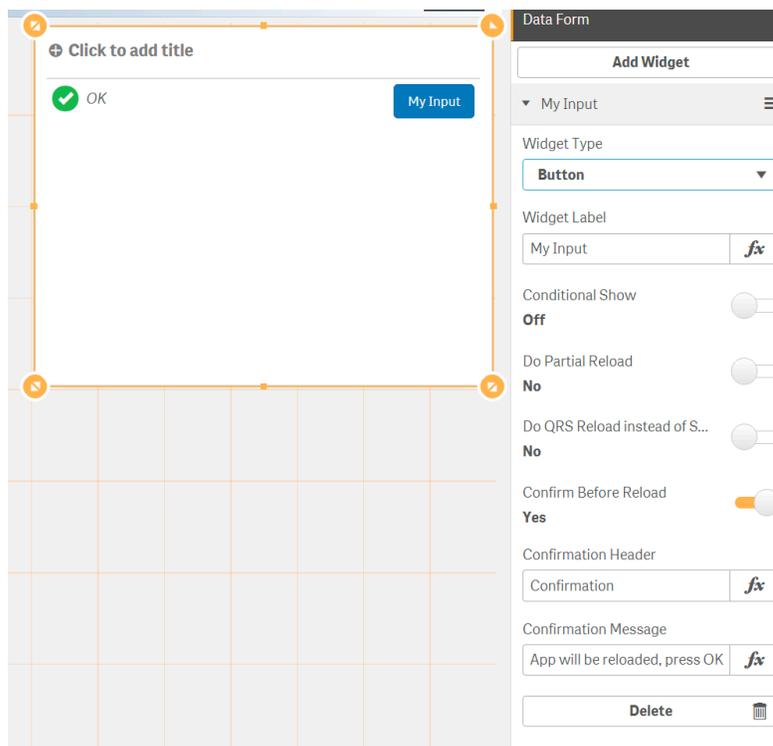
My Input



Option A Option B Option C

4. Button

Key component. Used to commit your inputs and trigger saving action.

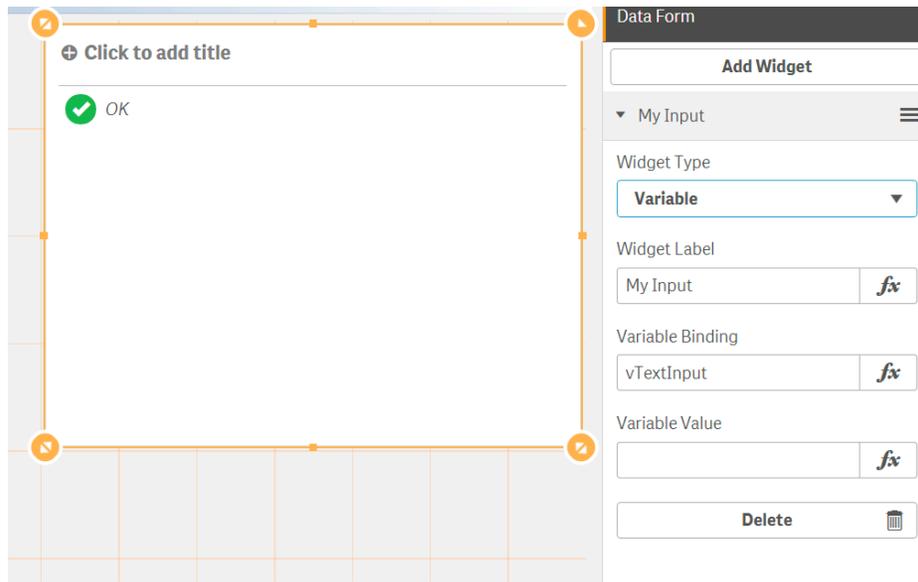


The screenshot shows a data form editor interface. On the left, a form preview displays a title "Click to add title", a green checkmark with "OK", and a blue button labeled "My Input". On the right, the "Data Form" configuration panel is open, showing the "Add Widget" section. The "My Input" widget is selected, and its configuration is as follows:

- Widget Type: **Button**
- Widget Label: My Input *fx*
- Conditional Show: **Off** (toggle off)
- Do Partial Reload: **No** (toggle off)
- Do QRS Reload instead of S...: **No** (toggle off)
- Confirm Before Reload: **Yes** (toggle on)
- Confirmation Header: Confirmation *fx*
- Confirmation Message: App will be reloaded, press OK *fx*
- Delete button with trash icon

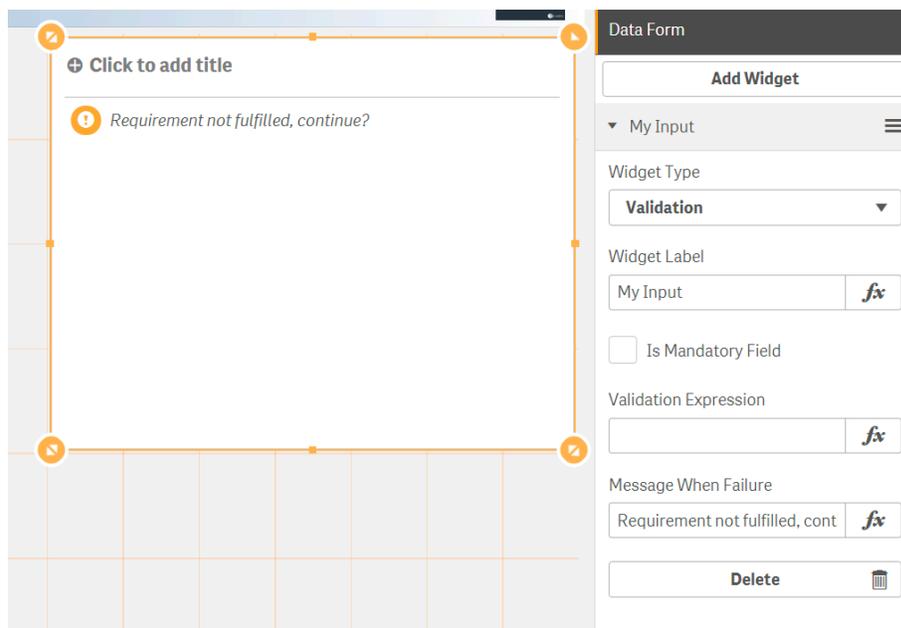
5. Variable

This widget is not visible through UI. It's used to pass a hard coded/calculated value to backend when saving comments, mainly used for meta info to control customized saving procedure during data reload.

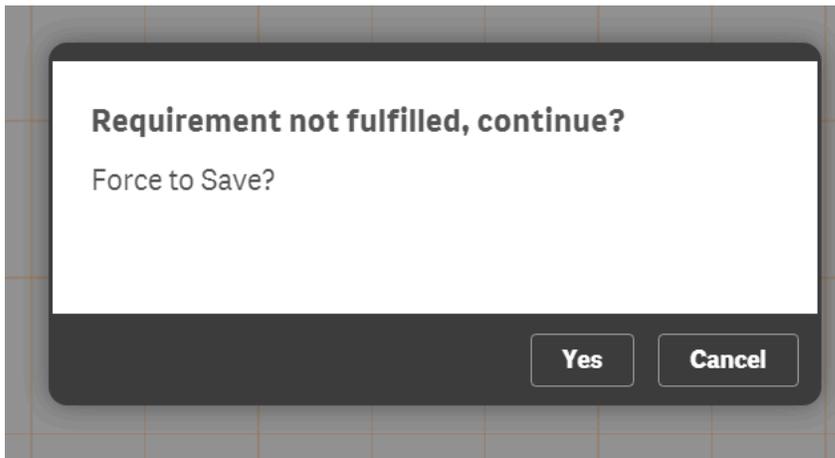


6. Validation

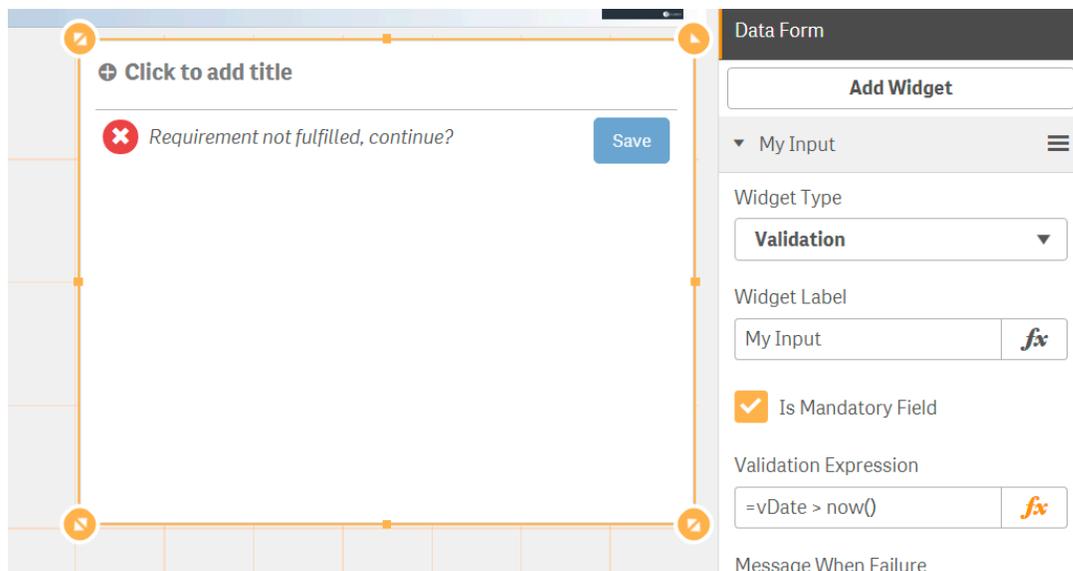
This widget is used to setup data check before submitting your inputs. A validation expression is required here to check if the input data is fulfilling requirements and limits. Check the other widgets' content by evaluating the **binding variables** in each widget.



When **Is Mandatory Field** is unchecked, even the validation is not passed, a confirmation will be prompted before submitting. And it's possible to force the save by pressing **Yes**



When **Is Mandatory Field** is checked, you will not be able to do the save before fulfilling requirements. The save button will be greyed out.



Setups in load script

A **QlikView Script file** (.qvs) is required to be included in Load script for saving data during partial reload.

```
1 //Load Commenting Lib
2 $(Must_Include=[lib://Qlik_Home/GENERAL\QVS\CommentingLib.qvs]);
```

The **S-formed Script Lib** contains most commonly used **SUB** during saving procedure.

A typical commenting Script will look like this:

```

1 //define the folder to store and load commenting QVDs
2 SET vQvdPath = 'lib://CommentingQVDs/';
3
4 //Just in case a comment table is not generated before
5 Comments:
6 Load
7     null() as [CASEID],
8     null() as [Comments.CreatedAt],
9     null() as [Comments.CreatedBy],
10    null() as [Comments.Status],
11    null() as [Comments.Comment]
12 AutoGenerate 0;
13
14 //Load all comments in when doing a full reload
15 For each vFile in FileList('${vQvdPath}\*_Comments.qvd')
16     CDRComments:
17     LOAD
18     *
19     FROM [$(vFile)](qvd);
20 Next vFile
21
22
23 //Comment storage will be handled only in partial reload
24 If IsPartialReload() then
25
26     //Use an extra vCommand variable to distinguish the purpose of this partial reload
27     If vCommand = 'saveComment' Then
28
29         //An input table contains all comment fields together with their values
30         [Input]:
31         Replace
32         Load
33             '${vCM_Status}' as [Comments.Status],
34             '${vCM_Comment}' as [Comments.Comment]
35         AutoGenerate 1;
36
37         //Call commenting lib function GenerateComment to generate a new comment and stores as a qvd
38         //GenerateComment(SrcTable, CommentTable, LinkageFieldName, InputTable, Matches)
39         Call GenerateQvdComment('Cases', 'Comments', 'CASEID', 'Input', vKeysToBeReloaded, vQvdPath)
40
41         //Reset variable if necessary
42         SET vCM_Status = '';
43         SET vCM_Comment = '';
44     EndIf
45 EndIf
46
47 //This function will generate an aggregated value for a certain commenting field in order to get latest value from all history log
48 Call AggregateComments('Cases', 'Comments', 'CASEID', 'Status', 'New')
49 Call AggregateComments('Cases', 'Comments', 'CASEID', 'Comment')
50

```

Security Rules for Published Apps

In most cases, end users need to share a published app for doing comments on. This will require a minimum additional **Security Rules**.

CommentingRole - App	App_*	Update
CommentingRole - AppObject	App.Object_*	Read
CommentingRole - DataConnection	DataConnection_*	Read

When assigned with **Commenting Role**, users will be able to do comments directly on published apps.