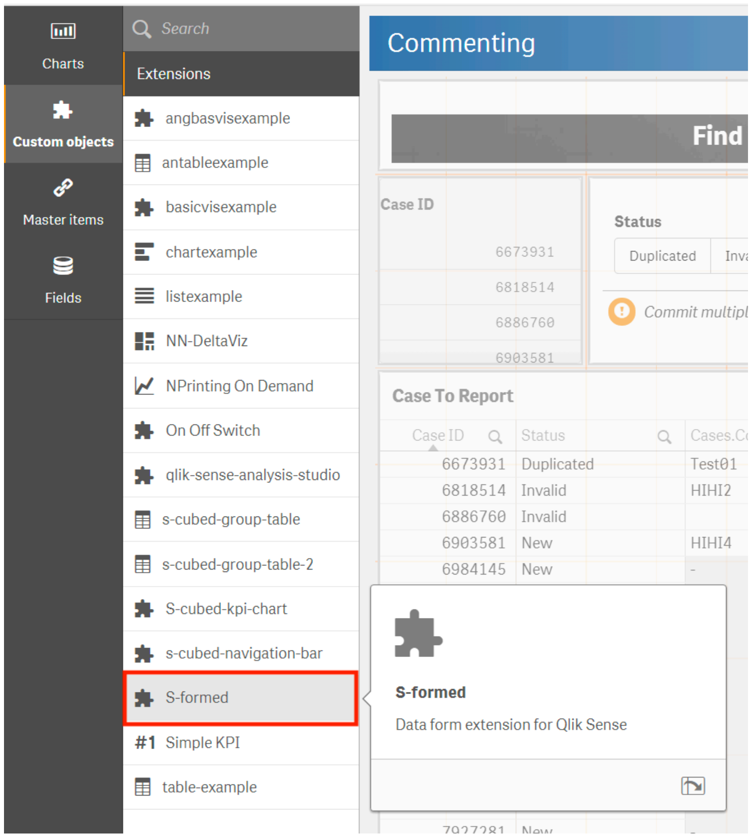
# S-formed Commenting Extension

A seamless way to save data in Qlik Sense app

## Configure Commenting UI

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

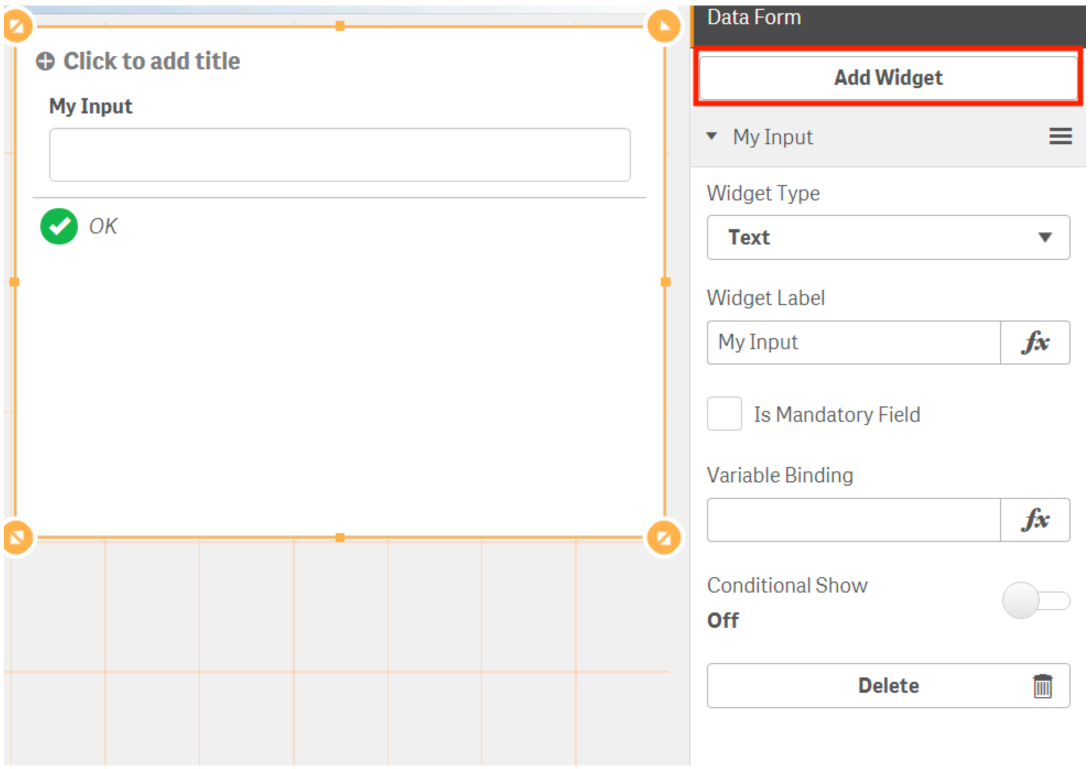


Case To Report

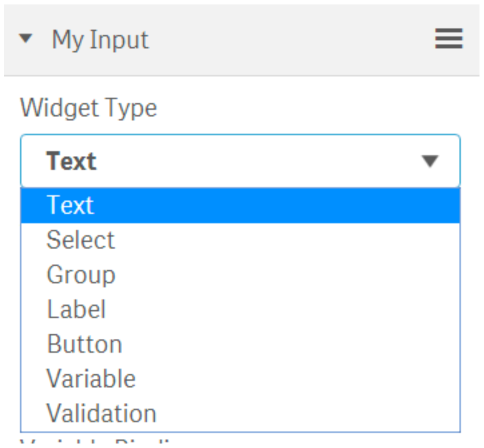
| Case ID | Status | Cases.C |
|---|---|---|
| 6673931 | Duplicated | Test01 |
| 6818514 | Invalid | HIHI2 |
| 6886760 | Invalid | |
| 6903581 | New | HIHI4 |
| 6984145 | New | - |
| 7027281 | New | - |

**S-formed**
Data form extension for Qlik Sense

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

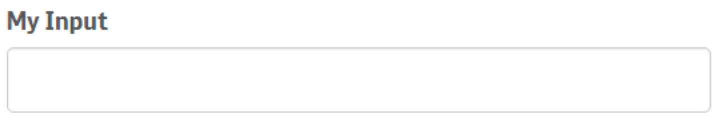


By giving a certain **Widget Type**, you can decide in what kind of iteration way should this widget be rendered and configured.



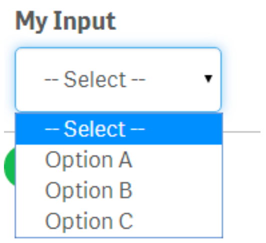
1. Text

   Plain Text input, mainly for free text commenting purposes

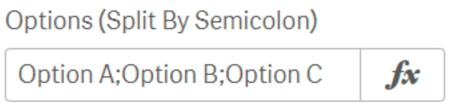


2. Select

   A dropdown menu will be rendered, mainly for scoped options

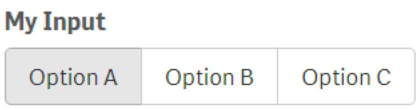


The options can be configured in **Options** property, values need to be split by semicolons. **Qlik Expression** is supported, so by using **concat**() function, you can dynamically present options according to data field or variables



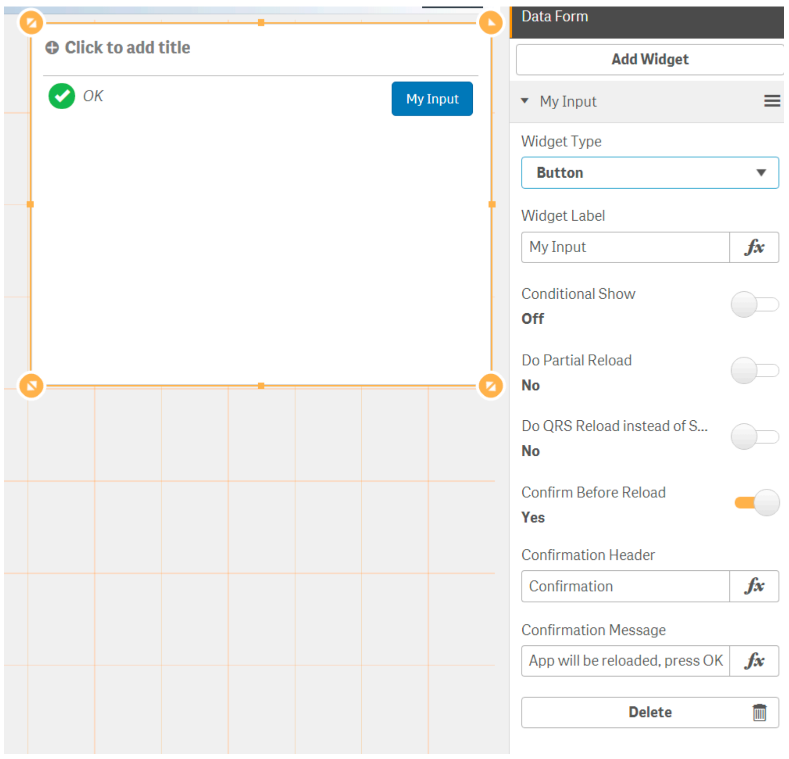
3. Group

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



4. Button

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



Widget Type: Button

Widget Label: My Input

Conditional Show: Off

Do Partial Reload: No

Do QRS Reload instead of S...: No

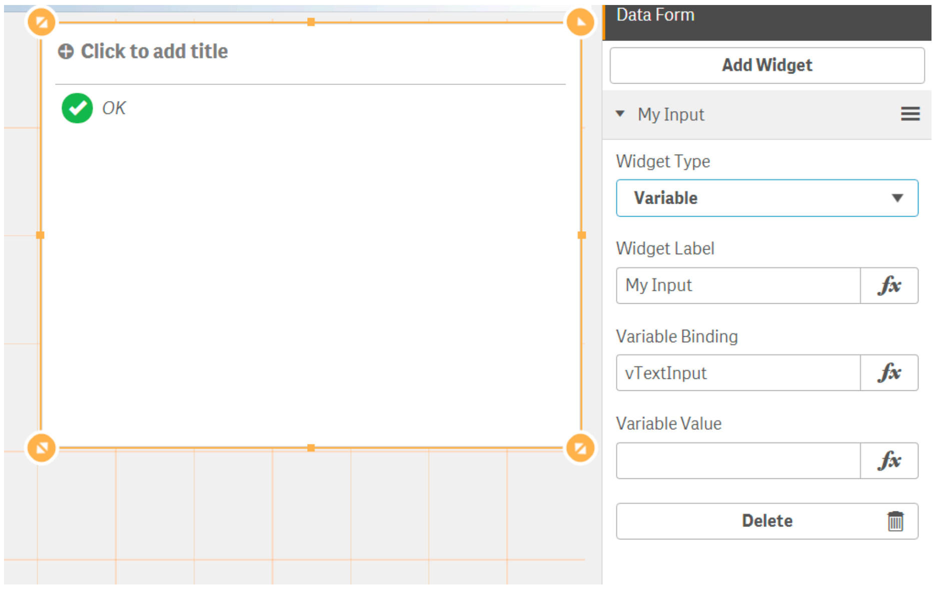
Confirm Before Reload: Yes

Confirmation Header: Confirmation

Confirmation Message: App will be reloaded, press OK

5. Variable

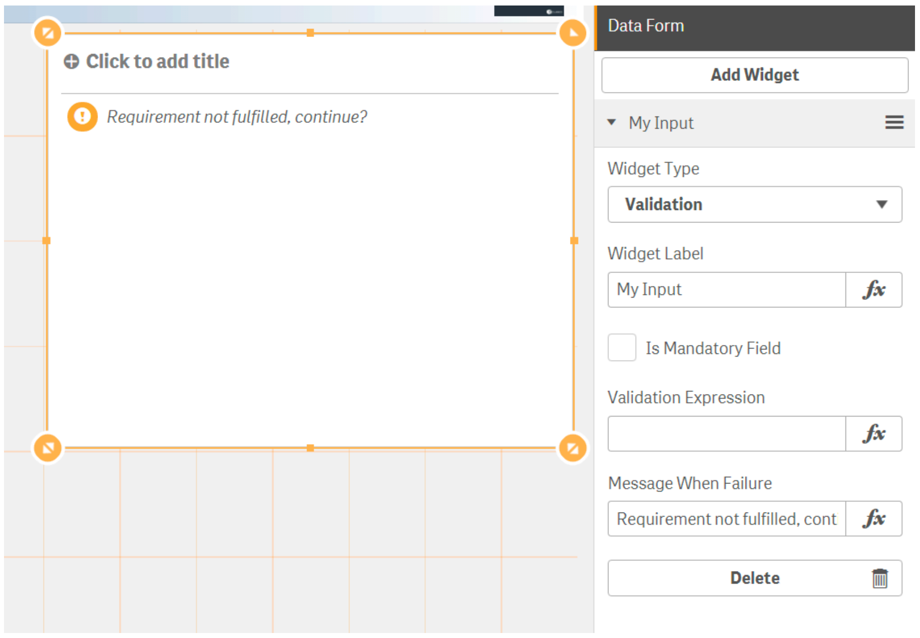
This widget is not visible through UI. It's used to pass a hard coded/calculated value to backend when saving comments, mainly used for meta info to control customized saving procedure during data reload.



Widget Type: Variable
Widget Label: My Input
Variable Binding: vTextInput
Variable Value:

6. Validation

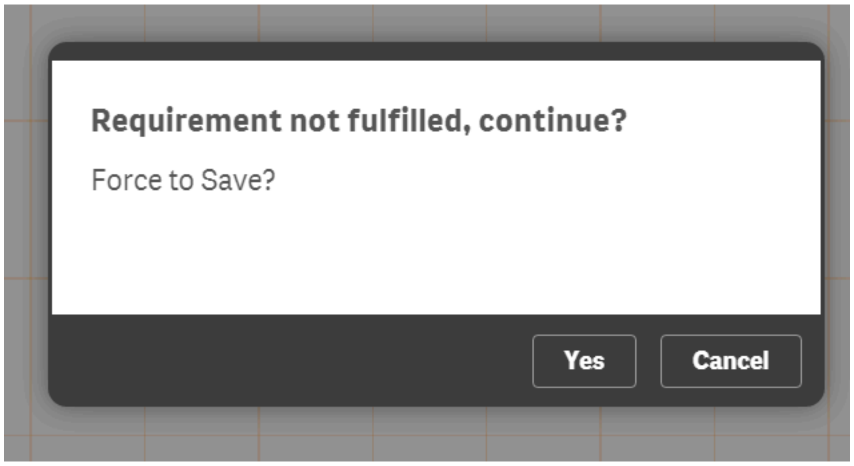
This widget is used to setup data check before submitting your inputs. A validation expression is required here to check if the input data is fulfilling requirements and limits. Check the other widgets' content by evaluating the **binding variables** in each widget.



Requirement not fulfilled, continue?

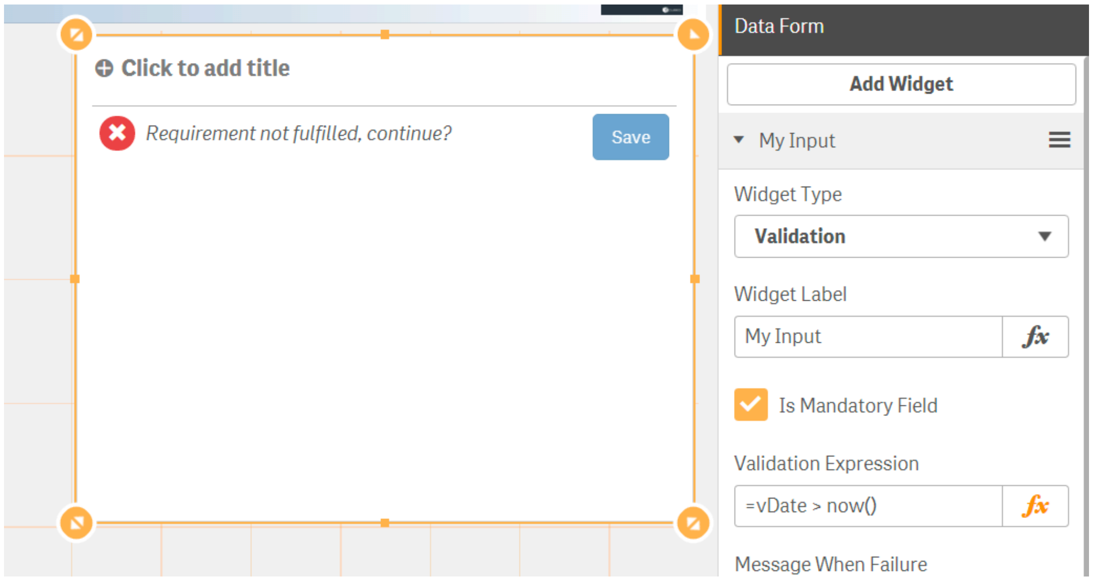
Widget Type: Validation
Widget Label: My Input
Is Mandatory Field
Validation Expression:
Message When Failure: Requirement not fulfilled, cont

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**


Requirement not fulfilled, continue?

Force to Save?

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.



*Requirement not fulfilled, continue?*

Widget Type: Validation

Widget Label: My Input

✓ Is Mandatory Field

Validation Expression: =vDate > now()

Message When Failure

## Setups in load script

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

```
//Load Commenting Lib
$(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:

```
//define the folder to store and load commenting QVDs
SET vQvdPath = 'lib://CommentingQVDs/';

//Just in case a comment table is not generated before
Comments:
Load
    null() as [CASEID],
    null() as [Comments.CreatedAt],
    null() as [Comments.CreatedBy],
    null() as [Comments.Status],
    null() as [Comments.Comment]
AutoGenerate 0;

//Load all comments in when doing a full reload
For each vFile in FileList('$(vQvdPath)\*_Comments.qvd')
    CDRComments:
    LOAD
        *
    FROM [$(vFile)](qvd);
Next vFile


//Comment storage will be handled only in partial reload
If IsPartialReload() then

    //Use an extra vCommand variable to distinguish the purpose of this partial reload
    If vCommand = 'saveComment' Then

        //An input table contains all comment fields together with their values
        [Input]:
        Replace
        Load
            '$(vCM_Status)' as [Comments.Status],
            '$(vCM_Comment)' as [Comments.Comment]
        AutoGenerate 1;

        //Call commenting lib function GenerateComment to generate a new comment and stores as a qvd
        //GenerateComment(SrcTable, CommentTable, LinkageFieldName, InputTable, Matches)
        Call GenerateQvdComment('Cases', 'Comments', 'CASEID', 'Input', vKeysToBeReloaded, vQvdPath)

        //Reset variable if necessary
        SET vCM_Status = '';
        SET vCM_Comment = '';
    EndIf
EndIf

//This function will generate an aggregated value for a certain commenting field in order to get latest value from all history log
Call AggregateComments('Cases', 'Comments', 'CASEID', 'Status', 'New')
Call AggregateComments('Cases', 'Comments', 'CASEID', 'Comment')

```

# 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.