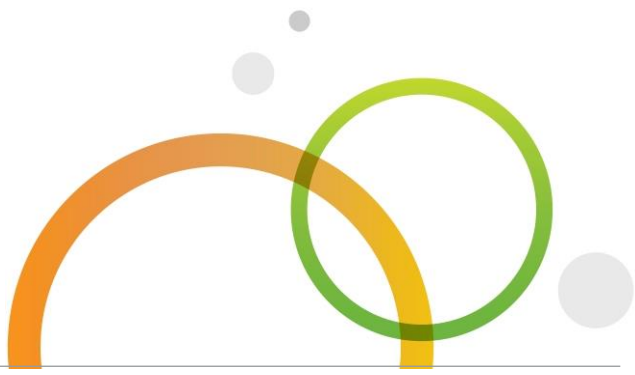
# QlikView Deployment framework

## Qlik Sense and QlikView Deployment Framework

March, 2015



## Table of Contents

**Version 0.3**

# Qlik Sense and QlikView Deployment Framework

All the enhanced functionality that QDF brings to QlikView can also be used by Qlik Sense. QlikView Deployment Framework is still managed from QlikView using the Variable Editor but from QDF 1.4.1 QlikView and Qlik Sense are seamlessly integrates between each other. This means that container resources like sub functions and global variables are identical and available within both systems.

## Qlik Sense folder connection

Qlik Sense has a new folder access method called *folder connection.* Adding a *folder connection* will create a **LIB** URL to the repository folder. The traditional QlikView way to accessing files (*c:\xx\yy* or *\\ServerName\yy*) is in Sense called *legacy mode* and disabled by default while accessing files using LIB's is called *native mode* and enabled by default. QlikView Deployment Framework works in native mode and utilizes the LIB url's.

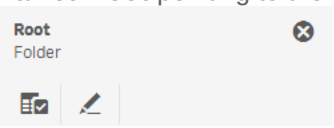
Qlik Sense integration into QDF can be done in two different ways. First is to mount the whole container structure (root) under on single LIB (*Singe LIB mounts*) and the second is to mount each container individually as its own LIB folder connection (*Separate LIB mounts*). Mode used is depending on the LIB setup and will be identified automatically when the framework initializes. The modes can be mixed from app to app in the same setup, for example depending on security level.

Example:

| **QlikView Global Variable** | **Qlik Sense Global Variable** | |
|---|---|---|
| *vG.BasePath=\\Server\QDF\1.Example* | *vG.BasePath=LIB://Example* | *Separate Mounts* |
| | *vG.BasePath=LIB://Root\1.Example* | *Single Mount* |

## Single LIB mount

When using single mount, QDF locates containers in the same way as QlikView by use of the container map. Minimum requirement for single mount is to add a Sense folder connection (LIB) named ***Root*** pointing to the framework starting point.

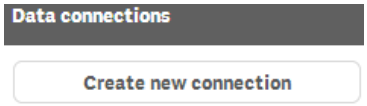



Change the default ***Root*** folder connection name by adding *SET vG.RootContainer='LIB name'* before QDF Initiation, the value should match the folder connection name pointing to ***root***, see more under *Optional settings*.

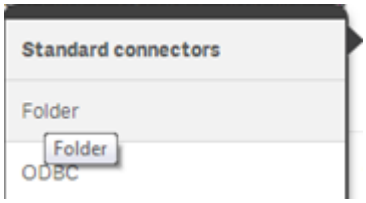
### Home container

Home container is the place where the generated global variables will be pointing to (ex. *vG.QVDPath*) and the location where sub functions and global custom variables are read from. The default home container will be identified automatically (usually *0.Administration)* to change this behavior add *SET vG.HomeContainer='Physical container name'* in the script beginning, see more under *Optional settings*.

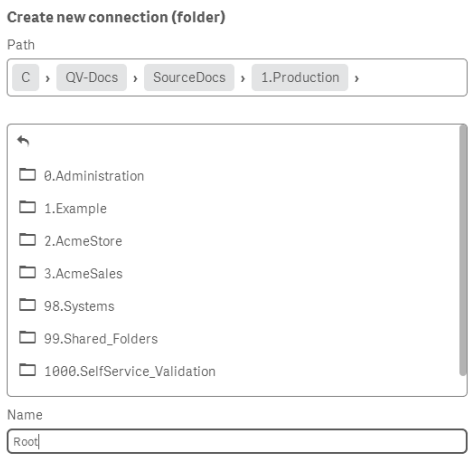
- To create the ***LIB://Root*** described above we first need to create a new *Folder Connection* in Qlik Sense. This is done under *Data connections* selecting *Create new connections.*




- Select *the Folder Connection alternative.*




- Point to the root location of QDF container architecture and give it the name Root. Type in UNC or drive path here.




The *Folder Connection* name is case sensitive.

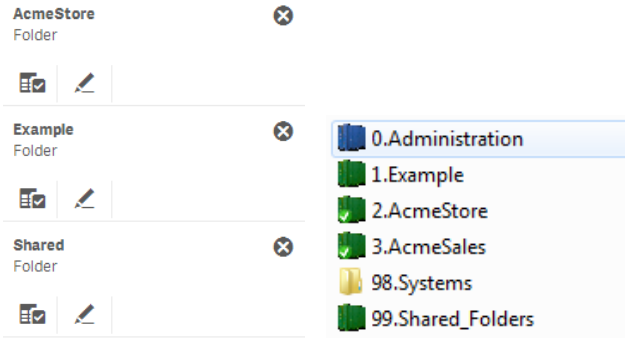
### Notes

When using single mount and a container is not under the root (called Alt Root Path in QDF container Map) a Separate LIB mount need to be created. This mount should have the same name as the *Variable Prefix* in the Container Map.

Home container identification is only done automatically directly under the root LIB, to use specific container always use *vG.HomeContainer* variable before of 1.Init initiation.
*Example SET vG.HomeContainer= '98.Systems\1.Oracle'*

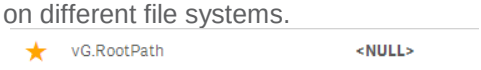
## Separate LIB mounts

In this mode the container map in QDF is not read within Qlik Sense, instead each folder connection (LIB) is pointing to an accessible container. In other words add a folder connection for each container you want to use in your application, use same folder connections name (LIB) the *Variable Prefix* name in Container Map. A good thing of using separate mounts is that in Qlik Sense server version LIB's can be managed by the system administrator thereby restricting file access.




Example, add folder connection with the name *Shared* pointing to your shared container (99.Shared_Folders), this will generate global variables to the shared resources (*vG.SharedQVDPath= LIB://Shared/2.QVDPath*).

Note. When using Separate LIB mounts the global variable *vG.RootPath* (pointing at the framework root path) is removed (NULL) as LIB's does not have any actual root path as different LIB's can point on different file systems.

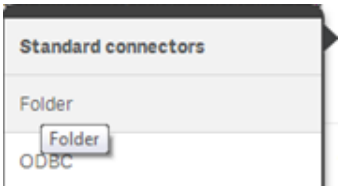


### Home container

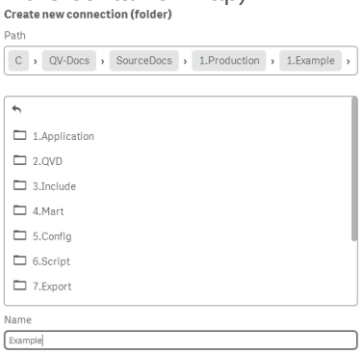
Home container is the place where the generated global path variables will be pointing to (ex *vG.QVDPath*) and the location where sub functions and global custom variables are read from. A Sense folder connection (LIB) need to be created. The lib name ***Home*** is default, but this is easy to change. Just add a *SET vG.HomeContainer='LIB-Name'* statement in the script beginning before framework initiation see Optional settings for more info.

To create *LIB://Example* shown above we first need to create a new *Folder Connection* in Qlik Sense. This is done by under *Data connections* selecting *Create new connections.*


And select *Folder Connection*




Point to the location of 1.Example container path and give it the name Example (same name as used in the container map).




When done you need to add framework initiation statement pointing to the Example, easiest is to use statement below.

```
SET vG.HomeContainer='lib://Example';
$(Include=$(vG.HomeContainer)\InitLink.qvs);
```

## Initiating using 1.Init

The initiation work in almost the same way as in QlikView, the same *1.Init.qvs* Include file (in the script beginning) initiates the framework. The only difference is that the url need a folder connection (LIB).

*Example Single LIB Mount:*
*$(Include=lib://Root/1.Example/3.Include\1.BaseVariable\1.Init.qvs);*

*Example Separate LIB Mounts:*
*$(Include=lib://Example/3.Include\1.BaseVariable\1.Init.qvs);*

When using the optional settings, like changing the default home container these settings should be places above the initiation statement, example:

*SET vG.HomeContainer='Example'*
*$(Include=lib://Example/3.Include\1.BaseVariable\1.Init.qvs);*

## Initiating using InitLink

From QDF 1.5 Qlik Sense InitLink support has been added. This makes it easier to initialize the framework within Qlik Sense. Instead of a long url to *1.Init.qvs* the InitLink will "sling shot" to 1.Init, example:

*Example Single LIB Mount:*

*$(Include=lib://Root/1.Example/InitLink.qvs);*

*Example Separate LIB Mounts:*

*$(Include=lib://Example/InitLink.qvs);*

or

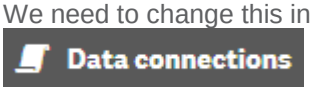
*SET vG.HomeContainer='lib://Example';*
*$(Include=$(vG.HomeContainer)\InitLink.qvs);*

## Qlik Sense Server

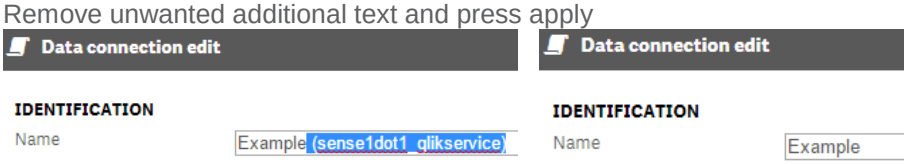
When creating Lib's in Qlik Sense Server (during app development) the Lib name also contains concatenated *ComputerName + UID* this to make Lib's personal by default. When developing using QDF we do not want personal LIB's, instead the QDF folder LIB's should be common for all developers assigned by security rules. In the QMC *Data Connections* tab we can easily modify the name and apply one or more security rules depending on developer access rights.

Our Example Lib have (*ComputerName + UID)* added to the name

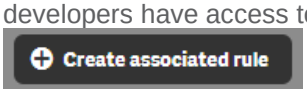

We need to change this in the QMC Data connections Tab




Remove unwanted additional text and press apply




An *Associated Security rule* can also be added to the Data connection, so that only authorized developers have access to the container (In our case Example Container).




## Creating a Qlik Sense *Kick Start* container

An alternative way of integrating Qlik Sense and QlikView is to use a separate container only used as a "kick start" for Qlik Sense. In this Sense container scripts would then be used only by Qlik Sense and could be modified without interfering with QlikView.

- ***Single LIB Mount*** Sense kick start container should have the name *0._Sense* (directly under root) it should then be identified without using *vG.HomeContainer* variable.
  Initiation: *$(Include=lib://Root/0._Sense/3.Include\1.BaseVariable\1.Init.qvs);*
- ***Separate LIB Mounts*** create a folder connection (LIB) named ***Home*** pointing to your kick start container.
  Initiation: *$(Include=lib://Home/3.Include\1.BaseVariable\1.Init.qvs);*

To load data stored under additional containers (the kick start container) use the *LCGV* sub function after the Initiation, example:

*Call LCGV('Oracle', 'QVD;Sub')* would create the global variables *vG.OracleQVDPath* and *vG.OracleSubPath* to Oracle container.

## Optional settings

Single LIB Mount

*SET* vG.HomeContainer=*'1.Example'* Physical container name before 1.Init initiation script will use 1.*Example* container as Home container instead of the default *0.Administration* container. **Remember**, here you need to type the physical container folder name instead of LIB name.

*SET* vG.RootContainer='QDF' before 1.Init initiation script will use *QDF* LIB Folder as Root instead of the default Folder connection name (***Root***)

Separate LIB Mounts

*SET* vG.HomeContainer='Example' before 1.Init initiation script will use *Example* Folder connection as Home container instead of the default Folder connection name (***Home***).

*SET* vG.HomeContainer='lib://Example' before 1.Init initiation script will use *Example* Folder connection as Home container instead of the default Folder connection name (***Home***). This will also work with variables in the Include script as shown below:

*SET vG.HomeContainer='lib://Example';*
*$(Include=$(vG.HomeContainer)\InitLink.qvs);*

## Additional Notes

This is an early release of QDF for Sense integrating only the fundamentals of the framework, there are several tasks that cannot be done inside Sense, like:

- Container administration is still done from QlikView by the *VariableEditor*.
- Management of global variables is also done in QlikView by the *VariableEditor*.
- QDF upgrade wizard needs *VariableEditor* and QlikView 11 to work.
- Functions that create or delete files/folders from disk will not work in Qlik Sense as the *Execution* function been removed. Example is the *CreateFolder* function.