



Qlik Deployment Framework




QlikView Getting Started Guide

April, 2017





Table of Contents

Why a Framework?	3
Standards	3
Qlik Deployment Framework	3
Qlik Deployment Framework resource containers	4
Benefits using Resource Containers	4
Environmental Global Variables	4
QDF Deploy Tool	5
Container Type Selector	5
Container architecture	6
My first QDF QlikView app	6
Additional scripts and functions	7
Locale files	7
Sub Functions	7
Linking (mount) Containers	7
Default containers	9
Administrative Container (0.Administration)  0.Administration	9
Shared Folders Container (99.Shared_Folder)  99.Shared_folders	10
Resource Containers  1.Project_HR	10
Example Container (1.Example)	11
Manage and add containers using Deploy Tool	12
Variable Editor	13
Security to run Variable Editor	13
Container Map Editor	13
Help	14
Variable Editor Tab	15

Getting Started Guide V 1.7.0

Why a Framework?

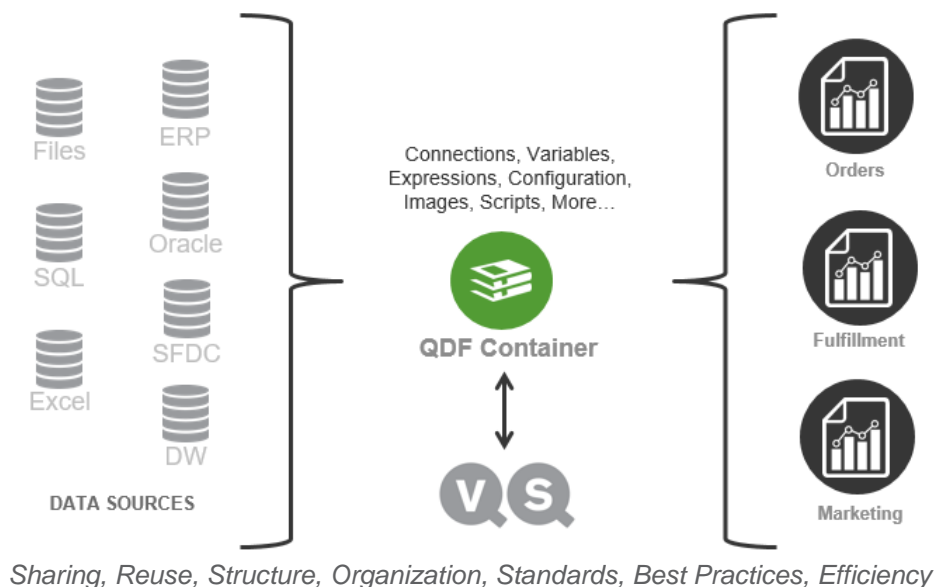
- **Do more in Less Time**
- **Improve Overall Quality**
- Develop in a way that others can understand
- Standards and Best Practice *“Don’t reinvent the wheel”*
- Rapid Development and Manageability in a Growing Deployment *“Scale up and out”*
- Multiple development teams

Standards

It’s important to have and use standards when developing and maintaining a development platform. There are many ways of getting the same result, but not all of them are efficient and understandable. By use of QDF in combination with guidelines it’s possible to create a cohesive multi development environment.

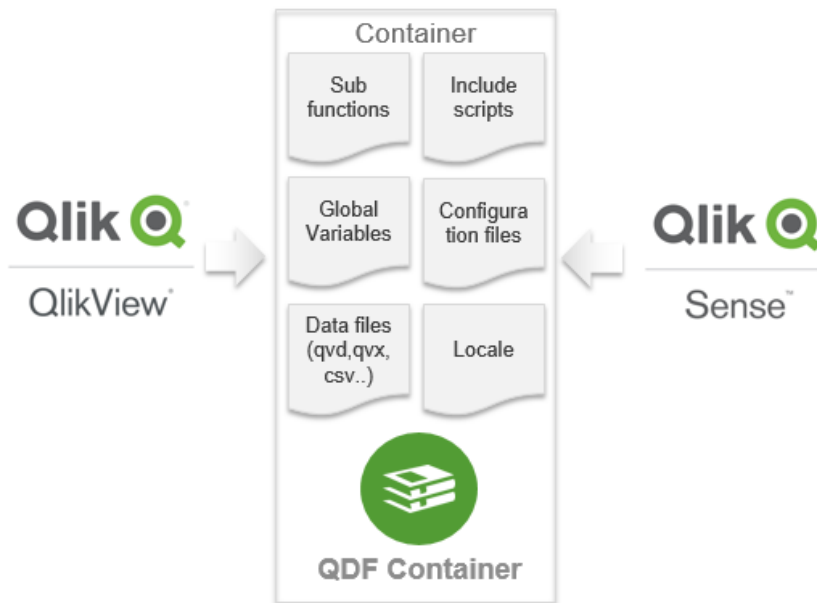
Qlik Deployment Framework

With the introduction of Qlik Deployment Framework building and managing Qlik deployments becomes easier and more efficient, for not only IT but professional services as well. QDF is a set of guidelines and utilities that enable: Resource Sharing, Reuse, Organization, Structure and Standards providing an effective and efficient deployment including both QlikView and Qlik Sense. The Qlik Deployment Framework (QDF) design and documentation is based on knowledge gathered from real life scenarios.



Qlik Deployment Framework resource containers

QDF is based on the concept of **resource containers**. This is building blocks and security separators that is aligned to fit the current needs. when demand changes its easy to reorganize and add additional containers. QlikView and/or Qlik Sense applications are hooked into the containers in which all needed resources are resided. Each container has identical file structures and contains predefined script functions.



Benefits using Resource Containers

- The built-in function library will simplify development and management.
- Containers represents security boundaries separating access based on roles.
- The container content is seamlessly reused between QlikView and Qlik Sense
- Containers can be created for different purpose, like common data and expressions (shared resources), departmental data, source data and many more...
- A container can be specialized for a single purpose, like SAP data source that IT want to control.
- Consolidation is easy when using container architecture. Just move containers from different locations to one single place.
- QDF is designed for application lifecycle management, to simply promote content and applications from development to production
- It is easy to share data between containers and still have the data separation.
- Scale from a small one container setup to a big 10 container setup.
- Metadata, the framework provides metadata regarding containers, data and variables.

Environmental Global Variables

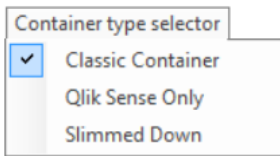
Qlik applications stored inside the containers need to execute an initiation script during reload. The initiation will dynamically create variable called **Global Variables** they are associated to the container you are in. When moving applications or container the initiation script will identify the new location and regenerate the Global Variables based on this new location so that applications works independent on location. This logic works with both UNC path and mapped drives.

QDF Deploy Tool

Containers are created using the QDF Deploy tool available for download on [Qlik community](#). The container script code is open source available on GitHub approved updated will be incorporated into the deploy tool that also updates existing frameworks to latest standard. **Read more regarding Deploy Tool in the attached Read-me notes.**

Container Type Selector

From QDF version 1.7.0 you have the possibility to select between three container layouts. This mean that QDF Deploy tool extracts different container layout (folders) depending on the selection done in the *Container Type Selector*. The different settings and correlating layout are presented below.



- **Classic Container (default)** has same container layout as traditional QDF (from version 1.0)
- **Qlik Sense Only** Here we have removed everything related to QlikView, for example Application and mart folders. Extract, Transform, Load folders have also been added under the QVD structure.
- **Slimmed Down** is intended for smaller deployments and several folders have been removed. Extract, Transform, Load folders have also been added under the QVD structure.

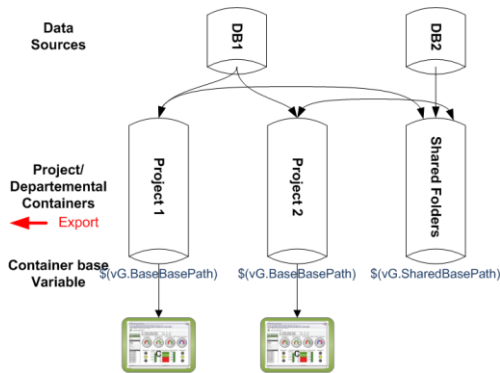
Classic Container	Qlik Sense Only	Slimmed Down
<ul style="list-style-type: none">1.Example<ul style="list-style-type: none">> 1.Application2.QVD> 3.Include> 4.Mart> 5.Config> 6.Script7.Export8.Import> 9.Misc	<ul style="list-style-type: none">1.Example<ul style="list-style-type: none">1.QVD<ul style="list-style-type: none">1.Extract2.Transform3.Load> 2.Config> 3.Include4.Export5.Import> 6.Misc	<ul style="list-style-type: none">1.Example<ul style="list-style-type: none">> 1.Application2.QVD<ul style="list-style-type: none">1.Extract2.Transform3.Load> 3.Include> 4.Config5.Import

Default Framework containers.

- **Administration** container it's from here additional containers are created and managed.
- **Shared folders.** A fresh QDF installation always contains a shared container, this is a repository to store scripts, configuration and data files that are shared and reusable by all applications.
- **Example** This is a container containing some examples and QVD files, this is used during the exercise documentation.

Container architecture

The Deploy tool creates and structures containers align to the development and deploy processes. In a basic container setup, each container loads and stores data from sources directly, common data is stored in the shared folders container and loaded into the project containers as shown in this picture.



My first QDF QlikView app

As mentioned earlier Qlik Deployment Framework applications need to have an initiation include statement in the beginning of the QlikView script.

1. Create or save QlikView application (preferably in a subfolder) under 1.Application folder in the container.
2. Create a first Tab called **Qlik Deployment framework** and Paste in the initiation script code below.

```
Let vL.InitLinkPath = Left(DocumentPath(), Index(DocumentPath(), '\1.Application')) & 'InitLink.qvs';
$(Must_Include=$(vL.InitLinkPath));
```

3. Reload and check in QlikView Variable Overview for Global Variables (vG.xxx) pointing into the container folders, as shown below:

Variable Overview		
Variables		
Variable Name	Value	Comment
vG.BasePath	C:\Users\mbg\Documents\QlikViewDeployment\	
vG.TemplatePath	C:\Users\mbg\Documents\QlikViewDeployment\	
vG.BaseVariablePath	C:\Users\mbg\Documents\QlikViewDeployment\	
vG.SubPath	C:\Users\mbg\Documents\QlikViewDeployment\	
vG.RootPath	C:\Users\mbg\Documents\QlikViewDeployment\	
vG.ApplicationPath	C:\Users\mbg\Documents\QlikViewDeployment\	
vG.DocumentationPath	C:\Users\mbg\Documents\QlikViewDeployment\	
vG.QVDPath	C:\Users\mbg\Documents\QlikViewDeployment\	

4. These variables are used when developing instead of hardcoding or using relative paths. The benefit is that applications can be moved in all directions without breaking the script.

Additional scripts and functions

Below are **optional** executions that can be done after *InitLink.qvs*.

Locale files

Contains include files with parameters that defines language, country and any special variant preferences that the user wants to see in their user interface. The locale global variable is *vG.localePath*.

Use this folder for custom Locale settings as well.

```
// Locale for US English
$(Include=$(vG.LocalePath)\1.US.qvs);
```

Sub Functions

Qlik have the possibility of reusing scripts and functions by using the **Sub** and **Call** commands. As presented below with the *LCGV* function. The Framework contains library of nice to have functions. All sub functions are stored under the *3.Include\4.Sub* folder and are initiated during QDF initiation.

Use *Call function_name('Input parameters or variables')* command to execute the preloaded function.

Another function *example, vL.FileExist* will return true or false depending on if the file exists

```
Call vL.FileExist ('$(vG.QVDPath)\SOE.qvd')
```

- More examples can be found in **Qlik Deployment Framework-QlikView Development Guide**.

Linking (mount) Containers

After QDF initiation you can start using the sub function library function is important as it creates Global Variable Links other resource containers (and available folders). The benefit of using Global Variable links in the script is to create generic, reusable and movable code as QDF validates and regenerates the links every time the script runs.

Load Container Global Variables (LCGV)

By using *LCGV* function it's possible to create Global Variable links (mounts) between containers. In this way a script can fetch data from other container without hard coded url's (security access needed).

Example: `call LCGV ('AcmeTravel', 'QVD')` Will create all Global Variables linking to *AcmeTravel* QVD folder. Variables created will have similar name as home container but with the additional *AcmeTravel* prefix in the variable name, like *vG.AcmeTravelQVDPath* for QVD path to *AcmeTravel* container

`call LCGV ('Oracle', 'QVD;Include')`; Will create two Global Variable links to different resources in Oracle container, by using an additional switch and ';' separator creates Global Variables *vG.OracleQVDPath* and *vG.OracleIncludePath* (instead of linking all folders as in the first example).

Additional Sub Functions

Qlik have the possibility of reusing scripts and functions by using the [Sub](#) and [Call](#) commands. As presented above with the [LCGV](#) function. The Framework contains library of nice to have functions.

Qlik Deployment Framework- Function Reference Guide

The function reference guide is available as a separate document ***Qlik Deployment Framework-Function Reference Guide.pdf***. As the sub functions are identical to both Qlik Sense and QlikView the same guide applies to both platforms.

Connection strings for QlikView

Best practice is to keep the connection strings in a separate Include file, this behavior is supported by QDF (Qlik Sense is not using connection strings). Use the Global Variable `vG.ConnStringPath` to connect inside your container, example:

```
// Connection string to Northwind Access data source
$(Must_Include=$(vG.ConnStringPath)\0.example_access_northwind.qvs);
```

If the connection string is in another container like the Shared folders container use the Global Variable `vG.SharedConnStringPath` to connect, example:

```
// Connection string to Northwind Access data source
$(Must_Include=$(vG.SharedConnStringPath)\0.example_access_northwind.qvs);
```


Default containers

After Framework basic installation (using the Deploy Tool) pre-installed containers are:

- **Administration** container named *0.Administration*. This container contains admin stuff like Variable Editor, System Monitor. This container is used for administration of all the other containers.
- **Shared folder** container named *99.Shared_folder*. This container is used for company shared applications, kpi's and common Qlik data files (QVD) like Active Directory data.
- **Example** container (Optional) (*1.Example*). Containing example applications to understand how QDF development works. When getting started browse through the example container, look at example scripts and read the information inside the folders.

Administrative Container (0.Administration) 0.Administration

The Administrative container is the only mandatory container (Blue) and is pre-installed. Administration is used for administrative duties and for the Qlik platform maintenance. Only system administrators need to have access to this container.

Administration container includes more folders than the other containers, like framework templates, batch scripts and admin tools. The folder *0.Templates* (zero in the beginning) stores templates and exists only in 0.Administration container.

Administrative applications

Administrative container is pre-configured with administrative applications:

- **Variable Editor** adds and modifies **Custom Global Variables** and **System Variables**. Variable Editor also edits, modifies and creates new containers based on the **Container Map**. Read more in Variable Editor Section. Variable Editor Resides under *6.Scripts\2.VariableEditor*
- **QlikView System Monitor** preconfigured for Deployment Framework. Is used for monitoring the QlikView Environment. Resides under *1.Application\2.QlikViewSystemMonitor*. QlikView System Monitor is depended on the settings in *SystemSettings.qvs* Include file.
- **QlikView Governance Dashboard** Initiation script *DF_Script_for_GD.qvs* to use GD in combination with QDF. Read more in Operations Guide.
- **Index Monitor** This tool will load in and monitor qvd meta-data index created by the index function built into QDF. Works for both Qlik Sense and QlikView.

Shared Folders Container (99.Shared_Folder) 99.Shared_folders

As default Deployment Framework contains the shared folders container (*99.Shared_Folders*) this is an (optional) repository for scripts and files that are shared between all containers. Shared Folder container is a “*single point of truth repository*”, a single view between all containers. Setup the Shared Folder security so that container owners have access to the shared container, read more in the security section of **Operations Guide**.

Shared Global Variables are loaded by the Initiation script *1.Init.qvs*, so the shared folder variables are in the applications without loading any special script. Shared Folders container does not need to have the physical name 99.Shared_Folders, as long as a container is defined (in the Variable Editor) with the Prefix (tag) *Shared* will the container be treated as a shared repository.

Shared Container variable names

The Shared Global Variables are identical to the ordinary environmental Global Variable except for the Shared prefix (*vG.SharedxxxPath*), as shown below:

.\	<i>vG.SharedBasePath</i>	Container root path
1.Application	<i>vG. SharedApplicationPath</i>	
2.QVD	<i>vG. SharedQVDPATH</i>	
3.Include	<i>vG.SharedIncludePath</i>	
1.BaseVariable	<i>vG. SharedBaseVariablePath</i>	
2.Locale	<i>vG. SharedLocalePath</i>	
3.ConnString	<i>vG. SharedConnStringPath</i>	
4.Sub	<i>vG. SharedSubPath</i>	
5.ColorScheme	<i>vG. SharedColorSchemePath</i>	
6.Custom	<i>vG. SharedCustomPath</i>	
4.Mart	<i>vG. SharedMartPath</i>	
5.Config	<i>vG. SharedConfigPath</i>	
6.Script	<i>vG. SharedScriptPath</i>	
7.Export	<i>vG. SharedExportPath</i>	
8.Import	<i>vG. SharedImportPath</i>	

Resource Containers 1.Project_HR

All containers (except Administration) are Resource Containers. Resource containers are created by the *Variable Editor* or *Deploy Tool* and have a green container icon.

Resource Container Purpose

These are the primary containers for an organization and can be utilized in many different areas:

- Load and store company QVD data by using the *vG.QVDPATH* Variable.
- Store company specific connection strings with *vG.ConnStringPath* Variable.
- Load and distribute end user applications.
- Load and share Qlik Marts to developers and/or power users with *vG.MartPath* Variable.
- Store configuration files in *vG.ScriptPath*.
- Export data from Qlik to *7.Export* folder by using *vG.ExportPath* Variable.
- Store import data in *8.Import* folder, load into Qlik with *vG.ImportPath* Variable.
- Use as Self-Service Containers

Example Container (1.Example)

This is an optional container that includes examples for demoing and learning Qlik Deployment Framework.

Examples available in 1.Example Container

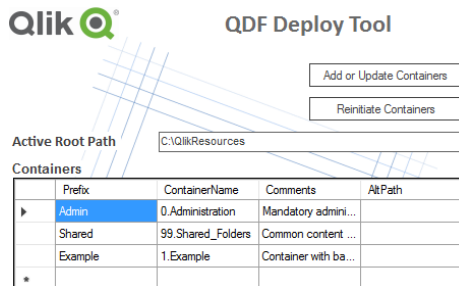
The optional Example container contains working examples, this for easy start and to understand how to use the Framework in the best way. These examples are:

- **QVD generator example**, loading from Northwind and writing qvd's into $$(vG.QVDPath)$
Resides under *1.Example\1.Application\1.QVD-Generator-example*
- **LoadIncludeExample** Basic example how to Load Deployment Framework Include Init scripts.
Resides under *1.Example\1.Application\2.LoadIncludeExample*
- **Calendar-Examples** that shows how the CalendarGen sub function works
Resides under *1.Example\1.Application\4.Calendar-Example*
- **QVD-Migration-example** demonstrates the power of using the QVDMigration sub function
Resides under *1.Example\1.Application\5.QVD-Migration-example*

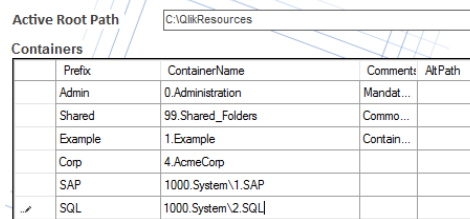
Manage and add containers using Deploy Tool

After the initial installation it's possible to enhance the framework (add containers) using the deploy tool. Active frameworks are identified automatically when adding a correct URL. Deploy tool will store last used URL to make it easy.

1. When active framework is found the Deploy path text will change to **Active Root Path**. Also *Documentation* and *Example Container* check boxed will be removed.



2. From the Containers grid additional containers can be typed and generated




3. Container properties:


- **Prefix** (Mandatory) this is the Container identifier, this is a **unique** short name for company department or project. This is used when mounting containers within the Qlik Scripts. Do not use long names or spaces
- **Container Name** (Mandatory) this is the physical container name shown on disk, could be company department or project. This could also include subfolders within the framework, examples:
 - **1.AcmeSales**
 - **100.DataSources\1.SQL-DB**
- **Comments** (optional) is used as descriptive meta-data.
- **AltPath** (optional) is used when creating a container in a different location. Example *C:\QlikTempContainers*

5. To add the new containers press, **Add or Upgrade Containers**.

Variable Editor

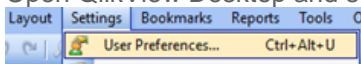
Variable Editor is a QlikView application that graphically controls Deployment Framework. System and Custom Global Variables can be added and edit within Variable Editor and all containers are plotted in a Container Map (master is stored in Administration container) this map is edited and containers created using Variable Editor, start by clicking on the Variable Editor Shortcut. If using User Access Control (UAC) right click on the *VariableEditor.cmd* and run as Administrator.

 VariableEditor Shortcut

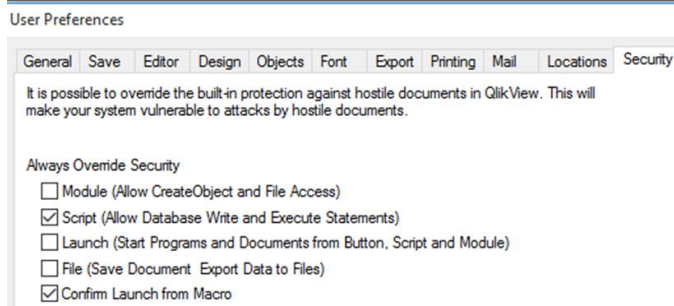
 VariableEditor.cmd

Security to run Variable Editor

1. Open QlikView Desktop and select Settings/ User Preferences. A User preferences box will appear



1. Change to Security tab and check in Script. And press Apply



Container Map Editor

Is used to administrate and populate containers within the framework. Press Go to Container Map to change to container view. Container Map is used by Deployment framework to identify containers.

Deployment Framework Container Map Editor 1.4.0
Container Map config in Admin Container

Go to Variable Editor Settings and Templates

Update Map and create Containers Reset Input and create Backup

Container Folder Name	Variable Prefix	Alt Root Path	Container Comments
0.Administration	Admin		Default Administration Container
1.Example	Example		Container containing Examples
2.AcmeStore	AStore		AcmeStore Container
3.AcmeSales	ASales		AcmeSales Container
98.Systems\1.Oracle	Oracle		First System Container containing Oracle Data
98.Systems\2.SAP	SAP		Second System Container containing SAP Data
98.Systems\3.SQL	SQL		Third System Container containing MS-SQL Data
99.Shared_folders	Shared		Default Shared Container

Active Container Layout

- C:\
 - QV-Docs
 - SourceDocs
 - 1.Production
 - 0.Administration
 - 99.Shared_folders
 - 2.AcmeStore
 - 3.AcmeSales
 - 1.Example
 - 98.Systems
 - 1.Oracle
 - 2.SAP
 - 3.SQL

Container Design View

- C:\
 - QV-Docs
 - SourceDocs
 - 1.Production
 - 98.Systems

Edit or modify container map in the table, remember that it's only the container Map that is changing not the physical container structure.

Help

There is a help button in the VariableEditor available when needed.

Container Input Fields

- *ContainerFolderName* contains the Container folder Name. To create or add in a sub container structure type *folder name\container name*. Example 1: *1.Oracle* to create a container under file root. Example 2: *98.System\1.Oracle* to create a container under 98.System folder.
- *ContainerPathName* enter prefix share variable names in *ContainerPathName* field, example *Oracle*.
- *Alt root path*, add a container in another file system, add a UNC path in *alt root path* field.
- *Container Comments* Is descriptive Meta Data regarding the containers, very good to use for documenting the solution.

Update Container Map

Use this button to apply the new Container map after adding and/or modifying the container layout.



Create New Containers option

Create New Containers will create containers based on the current container Map. This button is only shown after Update Container Map is applied and accepted.



Reset Input and Create Backup

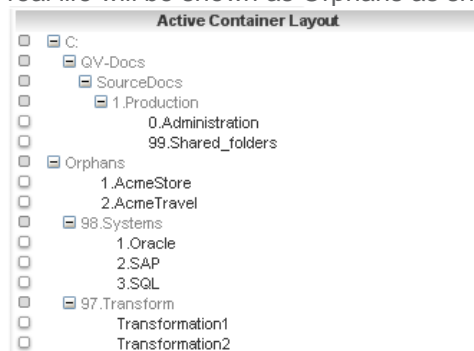
Will reset (revert) all inputs and also create a backup of the Container Map.

Retrieve Backup

Use Retrieve Container Map Backup to get back to the backup stage.

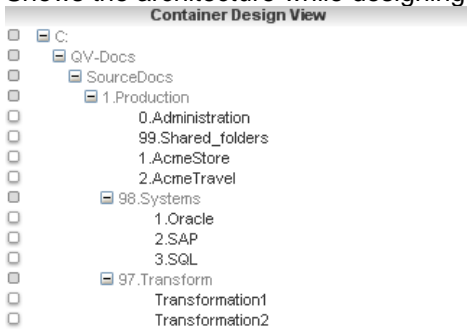
Active Container Layout

Shows physical containers that exist within the Container Map Container that exists in the Map and not in real life will be shown as Orphans as shown in the example below:

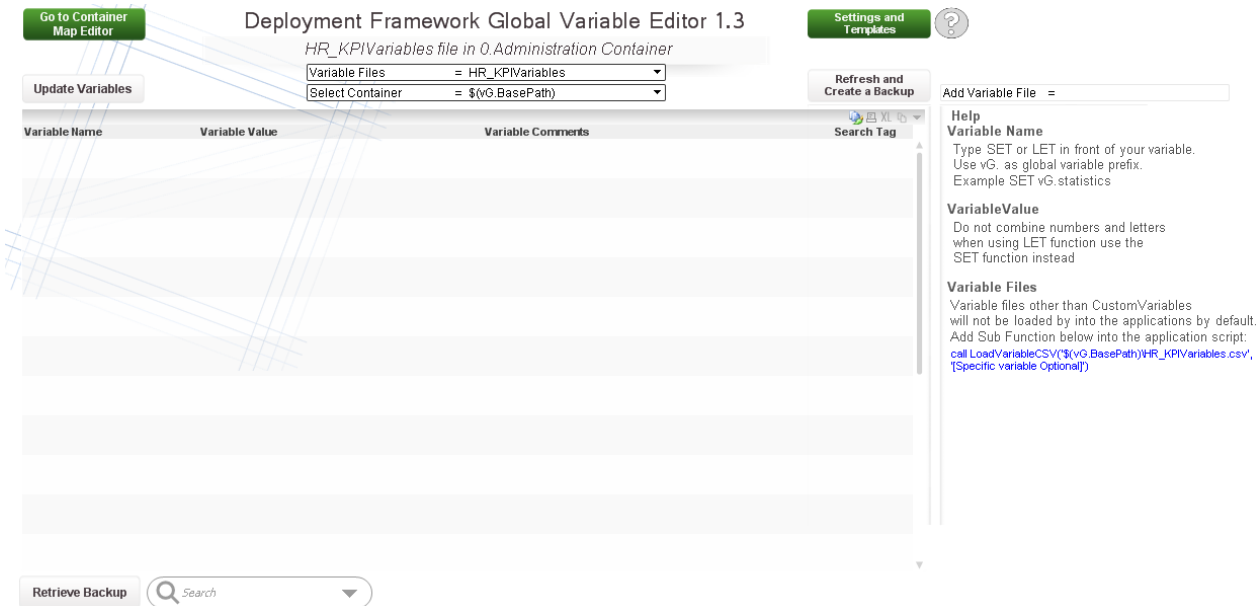


Container Design View

Shows the architecture while designing, in this view no Orphans is shown and no reload/refresh is needed.



Variable Editor Tab



Global Variables

The Global variables are modified by the Variable Editor and I stored in $$(BaseVariablePath)\CustomVariables.csv$ files in each container. Global variables (with the prefix *vG.*) are loaded by default into Qlik during the framework initiation process in the beginning of the script (read more in using Deployment Framework Containers). Global variables should only be used when a variable is shared by several applications in a Container.

Universal Variables

By using Universal Variables that are stored in $$(SharedBaseVariablePath)\CustomVariables.csv$ files in the Shared Folders Container, we get “single point of truth” across all containers. Universal Variables are by default loaded during the framework initiation process, have the prefix *vU* and is also modified by the Variable Editor application.

System Variables

System Variables are actually also Global Variables that start with (vG.), the difference is that System Variables are predefined variables used to store system settings like QlikView Server log path. System Variables are also not preloaded, 3. *SystemVariables.qvs* include script needs to be run to load in the System Variables into QlikView.

System Variables are modified by the Variable Editor and are stored in *\$(BaseVariablePath)\SystemVariables.csv*. There is usually only need for one System Variable version, the main is stored in 0.Administration container and is by default replicated out to the other containers.

Variable Input Fields

- *VariableName* Type *SET* or *LET* in front of your variable name. Use *vG.* or *vU.* as Global or Universal Variable prefix. Example1 *SET vG.statistics*. Example2 *SET vU.statistics*.
- *VariableValue* Type value or text, when entering text do not use brackets (") this is done automatically. Do not combine numbers and letters when using LET function, use the SET function instead for this.
- *Comments* Used for comments like author and creation date
- *Search Tag* Used only for easy search

Variable Files, Custom Global Variables

Custom Global Variables will automatically be loaded into Qlik applications when using Deployment Framework. Each Container has its own Custom Global Variable file that the applications use.

For Global Variables that need to be used across containers modify Shared Custom Variable file with Variable editor.

Refresh Create a Backup

Will refresh the view without updating Variable files and at the same time create a backup.

Retrieve Backup

Use Retrieve Backup to get back to the backup stage created by Change Variable File and Create a Backup button.

Update Variables

Use this button to apply the new variables after adding and/or modifying.

Add and Remove Variable Files

Variable Editor has the possibility to add variable files into the selected container in addition to the default *Custom Global Variables*. Type the variable filename into the *Add Variable File* input box and press enter like example below:

Add Variable File = HR_KPI

The *Refresh and Create a Backup* box will now change to a *Create Variable File* box

Create Variable File

When pressing apply the new csv file (empty) will be created as *HR_KPIVariables.csv* and stored under selected container *3.Include\1.BaseVariable*.

To remove a Variable File add the command **del** before the filename and run the script like example below:

Add Variable File = del HR_KPI

The box will change to *Delete Variable File*.

Delete Variable File

Variable files other than Custom Variables will not be loaded by *1.Init.qvs* into the applications by default.

Add Sub Function below into the application script instead:

```
$(Include=$(vG.SubPath)\2.LoadVariableCSV.qvs);
```

```
call LoadVariableCSV('$(vG.BaseVariablePath)\HR_KPIVariables.csv', '[Specific variable Optional]')
```

More detailed documentation on Variable Editor can be found in **Operations Guide**.