






Getting Started Guide

Getting Started Guide

March, 2015



Table of Contents

Getting Started Guide V 1.5.1	3
Why a Deployment Framework?	4
Why Standards?	4
Deployment Framework Resource Containers	6
Environmental Global Variables	6
Inside each container	7
Environmental Global Variables	8
Variable Naming Convention	8
Container architecture	9
Get started with QDF in QlikView Developer	10
Get started with QDF in Qlik Sense	10
Linking (mount) Containers	12
Sub Functions	12
Administrative Container (0.Administation)  0.Administration	13
Shared Folders Container (99.Shared_Folder)  99.Shared_folders	14
Resource Containers  1.Project_HR	15
Moving application between containers	15
Example Container (1.Example)	16
Variable Editor	17
Help	17
Container Map Editor	17
Variable Editor Tab	19

Getting Started Guide V 1.5.1

The deployment framework consists of several correlating documents. This document explains how to govern the Qlik platform by using the Deployment Framework. Deployment Framework is initially created for QlikView but has been enhanced for Qlik Sense.

The Deployment Framework documents are:

- **Getting Started Guide** Get an overall understanding of the framework basics and how to start installing and develop.
- **Operations Guide** for QlikView Administrators maintaining the platform and administrating security, tasks and containers.
- **Development Guide** for Developers how to work with DF in an efficient way, naming conventions, data modeling, optimization tricks/tools and other guide lines regarding development.
- **Deployment Guide** Project management guide on how to govern and manage Qlik deployment (DTAP) process, how to create Qlik projects and development teams and skill sets

Deployment Framework is designed for easy and basic deploys as well as complex deployment scenarios, this **Getting Started Guide** will not go into all the possibilities and scenarios, for this read the listed guides above.

When using QlikView, Publisher is recommended to separate Source Documents (Back End) and User Documents (Front End). The **Deployment Framework** core folder structures and scripts are stored under the Source Document folder (or folders) in the Back End.

Why a Deployment Framework?

Without frameworks and standards our world would have looked very different.



The world of frameworks

According to Wikipedia, A software framework is a universal, reusable software platform used to develop applications, products and solutions.

Deployment Framework is based on field experience from customers, partners and previous mistakes. The main purpose of Deployment Framework is to get manageability of the Qlik platform. An unstructured platform is more difficult to maintain and upgrade than if its structured and organized.

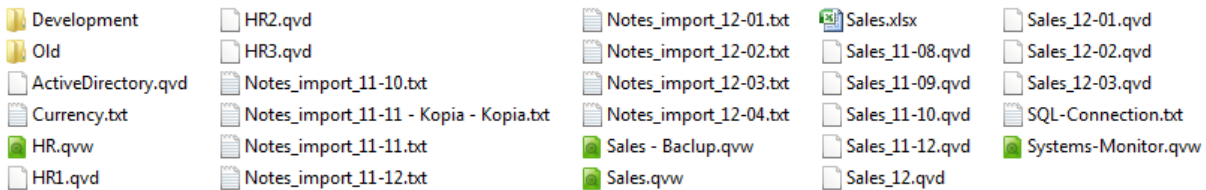
Why Standards?

It's important to have and use standards when developing and maintaining the Qlik platform. There are many ways of getting the same result, but not all of them are efficient and understandable. By use of Deployment Framework structure in combination with the methodology and standards we create a consistent multi development environment. Standards are needed for:

- Consistent way of developing and maintaining
- Reuse of data
- Reuse of code
- Reuse of expressions
- Multi development
- Create a Governed platform
- Creating and collecting understandable metadata
- Simplifying DTAP (Dev -> Test -> Acc -> Prod) process

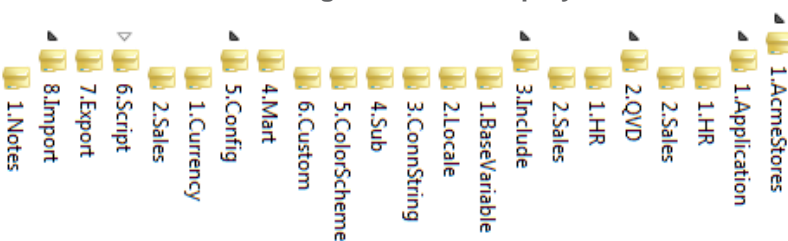
Example

Here is a simple unstructured QlikView environment where applications, qvd files, configuration files and import files are stored in a single folder. There is also a development folder where the development applications are placed.



Without structure when adding applications and generating QVD files the complexity will soon increase. These kinds of environments could become difficult to maintain. When using Qlik Deployment Framework, each function and file has its own place. The use of *Environmental Global Variables* makes it easy for QlikView and Qlik Sense to find and retrieve content.

Here are the same files organized with Deployment Framework.



By using Qlik Deployment Framework all the different document types and functions have their own place in the folder structure:

- The QlikView documents (qvw) are stored under *1.Application* folder in sub folders for HR and Sales
- QVD files is stored under *2.QVD* in sub folders for HR and Sales
- SQL Connection string is stored in *3.Include\3.ConnString*
- Configuration files like *Currency.txt* and *Sales.xlsx* are stored under *5.Config*
- Note import files are stored under *8.Import* in Notes sub folder.
- Development is in a separate but identical structure, making it easy to move into production.
- The *ActiveDirectory.qvd* that contains Active Directory data could either be stored under the QVD folder or in Shared Folders storage for easy reuse from other departments.

Deployment Framework Resource Containers

Qlik Deployment Framework is based on a **Resource Container Architecture**. Containers are identical but isolated folder structures placed side by side. A container can be moved and/or renamed without changing any Qlik script or logic inside it. Each container has identical file structures and includes the same base script functionality.

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 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.

Environmental Global Variables

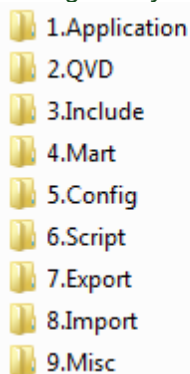
Qlik applications stored inside the containers need to execute an initiation script during reload. The initiation will dynamically create variable called **Global Variables** they are associated to the container you are in. When moving applications or container the initiation script will identify the 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.

Benefits using Resource Container Architecture

- Containers can be in different security groups, and is thereby a security separator.
- Containers are used for different purpose, like Administration, company data, departmental data and shared resources.
- A container can be specialized for a single purpose, like SAP data loading that IT want control over.
- Consolidation is easy when using container architecture. Just move containers from different locations and place them side by side.
- Container file structures are identical so Documents can be moved between containers without changing script parameters.
- In a DTAP (Develop, Test Acceptance, Production) environment moving from *Test* to *Acceptance* is when using QDF only to move applications from one container to another.
- It is easy to share Qlik Data files (qvd) between the containers, but still have the data segmented.
- Scalable, a small company probably only need one container, a big company may use 10+ containers.
- Metadata, the framework will provide metadata regarding containers and variables. It will be much easier to understand and reuse your metadata.

Inside each container

As mentioned earlier, all the containers have identical file structures and pre-configured scripts making it easy to develop and move applications between the containers.

- 
- 1.Application
 - 2.QVD
 - 3.Include
 - 4.Mart
 - 5.Config
 - 6.Script
 - 7.Export
 - 8.Import
 - 9.Misc

Container Naming Convention

The folder names inside the containers are simplified to fit as many languages and companies as possible.

Before each container and subfolder there are a sequential number that makes it easier to identify containers and subfolders, especially when using the Publisher.

Default container folder content

A Container structure consists of:

0.Template	Folder to keep custom examples and templates for easy reuse. Only exists in 0.Administration Container.
1.Application	QlikView Applications are resided in subfolders under 1.Applications
2.QVD	Qlik Data files are stored in subfolders under 2.QVD
3.Include	Folder where Qlik Include files are stored. These are script parts that are called from the main Qlik script.
1.BaseVariable	Stores all the variables needed to use the framework, like paths inside the container
2.Locale	Locale for different regions, used for easy migration between regions
3.ConnString	Stores connection strings to data sources
4.Sub	Store for sub routines, this is a way to reuse code between applications
5.ColorScheme	Company standard Color Scheme would be placed here
6.Custom	Store for custom include scripts
4.Mart	Resides QlikView Qvw marts (in subfolders) for data discovery usage, these folders could be shared.
5.Config	Configuration and language files like Excel and txt. This folders could be shared to make configuration changes easier
6.Script	Store for special scripts run by the publisher or scheduled tasks
7.Export	Folder used to store from Qlik exported data, probably txt or qvx
8.Import	Folder used to store import data from external systems
9.Misc	Storage for documentation, extension objects or whatever needed
Info.txt	Information files describing the folder purpose and Path variable. There are Info files in every folder.
Version.xx.txt	Version Revision list

Environmental Global Variables

Environmental Global Variables that are generated automatically when Qlik Deployment Framework is initiated. These variables are links to the default container file structure in the container where the application is executed from. If a folder inside container is removed the Global Variable correlating will not be created.

0.Administration		
1.AcmeStore		
1.Application	<u>vG.ApplicationPath</u>	Application Folder
2.QVD	<u>vG.QVDPPath</u>	QlikView Data files (QVD) repository
3.Include	<u>vG.IncludePath</u>	QlikView Include files repository
1.BaseVariable	<u>vG.BaseVariablePath</u>	Initiation script and Global Variables storage
2.Locale	<u>vG.LocalePath</u>	Regional / language Locale
3.ConnString	<u>vG.ConnStringPath</u>	Connection string repository
4.Sub	<u>vG.SubPath</u>	Sub Function Library
5.ColorScheme	<u>vG.ColorSchemePath</u>	Color and Picture templates
6.Custom	<u>vG.CustomPath</u>	Repository to keep custom include scripts
4.Mart	<u>vG.MartPath</u>	Qlik mart repository
5.Config	<u>vG.ConfigPath</u>	Stores QlikView configuration files
6.Script	<u>vG.ScriptPath</u>	Folder to store special scripts used by publisher
7.Export	<u>vG.ExportPath</u>	Data Export repository, probably txt or <u>qvx</u>
8.Import	<u>vG.ImportPath</u>	Data Import from external systems
2.AcmeHR		
99.Shared_Folders		

Variable Naming Convention

It's important to have a variable naming convention so that existing variables doesn't collide with the framework variables. Variable types in the framework are called **Global** and **Universal** and have the name standard **vG.xxx (Variable Global)** and **vU.xxx (Variable Universal)**. Variables only used in a single application are called Local and named **vL.xxx (Variable Local)** or use the original application variable standard usually **vXXX**.

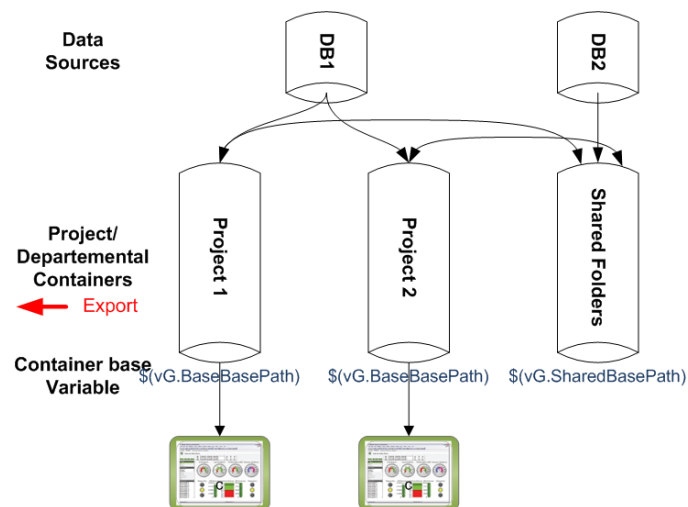
Environmental Global Variables are automatically created by the framework when running the initiation script *1.Init.qvs*. Read more below.

Custom Global Variables are Global Variables that are added manually by administrators or developers using the Variable Editor application. Custom Global variables (with the prefix **vG.**) are loaded default into QlikView or Qlik Sense during the framework initiation process in the beginning of the script (read more in *using Deployment Framework Containers*). Custom Global variables should only be used when variables are shared between several applications within the Container.

Universal Variables (prefix **vU**) are added manually by administrators or developers using the Variable Editor application and stored in *Shared Folders* Container that is used as "single point of truth" across all containers. Universal Variables are also loaded during the framework initiation process.

Container architecture

By using the Variable Editor Tool containers can be created and organized according to development and company processes, read more in *Deployment and Operations Guide*.



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 the picture above. Shared folders have its unique Environmental Global Variables (*vG.SharedxxxPath*) that is common and used by all other containers to reach Shared Folders.

More container architecture design solutions can be found in the **Operations Guide**.

Get started with QDF in QlikView Developer

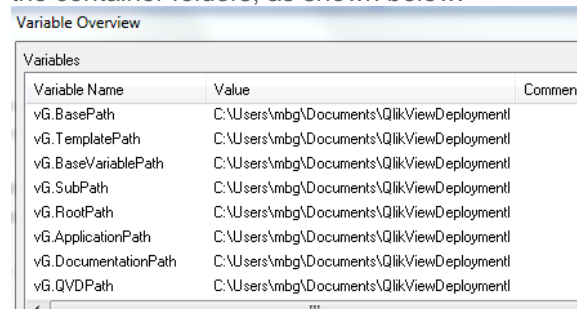
As mentioned earlier Qlik Deployment Framework applications need to have an initiation include statement in the beginning of the QlikView script. This initiation creates environmental *Global Variables* like container folder path and *Custom Global Variables* (Manually created Variables). The initiation script is named *1.Init.qvs* and resides under *3.Include\1.BaseVariable* in all containers.

1. Create or save QlikView application (preferably in a subfolder) under 1.Application folder in the container.

Create a first Tab called **Qlik Deployment Framework** and Paste the script code down below in. Use relative path for *1.init*.

```
$(Include=..\..\..\InitLink.qvs);  
$(Include=..\..\..\InitLink.qvs);  
$(Include=..\..\InitLink.qvs);  
$(Include=..\InitLink.qvs);  
$(Include=InitLink.qvs);
```

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



The screenshot shows the 'Variable Overview' window in QlikView. It contains a table with three columns: 'Variable Name', 'Value', and 'Comment'. The table lists several variables, all of which have values pointing to the same directory: 'C:\Users\mbg\Documents\QlikViewDeployment\'. The variables listed are vG.BasePath, vG.TemplatePath, vG.BaseVariablePath, vG.SubPath, vG.RootPath, vG.ApplicationPath, vG.DocumentationPath, and vG.QVDPPath.

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.QVDPPath	C:\Users\mbg\Documents\QlikViewDeployment\	

3. 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.

Get started with QDF in Qlik Sense

Please read the *Qlik Deployment Framework-Qlik Sense additional notes.pdf* document

Adding additional Include scripts

Below are initiation Include scripts executed after *InitLink.qvs* all of them are **optional** and used if needed.

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);
```

Connection strings

Best practice is to keep the connection strings in a separate Include file. This behavior is supported by Deployment Framework. 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);
```

Linking (mount) Containers

By using `LoadContainerGlobalVariables` function it's possible to create Global Variable links (mounts) between containers, this depending on security access. The short name `LCGV` can also be used and will work the same, example: `LCGV('SQL','QVD');`

Example: `call LCGV('AcmeTravel')` Will create all Global Variables linking to 2.AcmeTravel container. Variables created will have similar name as home container but with the additional *AcmeTravel* prefix, 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).

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. 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. As of v1.4 all sub functions included with QDF are preloaded *during the initiation process*.

Sub 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-Development Guide***.

Administrative Container (0.Administration) 0.Administration

The Administrative container is the only mandatory container (Blue) and is pre-installed during installation. 0. 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 QlikView Documents

Administrative container is pre-configured with administrative QlikView 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.
- **Governance Dashboard** Initiation script *DF_Script_for_GD.qvs* to use GD in combination with QDF. Read more in Operations Guide.

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>	
..\	<i>vG. SharedRootPath</i>	The root folder for all containers

Shared Folders Purpose

Shared folder container is easy implement and could be used for many different purposes:

- Store for *Universal Variables* that is automatically loaded by all applications (by the initiation script). Edit and add Universal Variables in the Variable Editor.
- Load and store common QVD data by using the *vG.SharedQVDPath* Variable
- Reuse Include scripts across containers by using the Shared Folder Variables:
 - *vG.SharedBaseVariablePath* pointing to 1.BaseVariable in Shared folders.
 - *vG.SharedLocalePath* pointing to 2.Locale in Shared folders.
 - *vG.SharedConnStringPath* pointing to 3.ConnString in Shared folders.
 - *vG.SharedSubPath* pointing to 4.Sub in Shared folders.
 - *vG.SharedColorSchemePath* pointing to 5.ColorScheme in Shared folders.
 - *vG.SharedCustomPath* pointing to 6.Custom in Shared folders.

- Import data into *8.Import* folder and use this data the other containers with the *vG.SharedImportPath* variable.
- Store common connection strings with *vG.SharedConnStrinPath* Variable
- Store common configuration files in *vG.SharedScriptPath*

Resource Containers 1.Project_HR

All containers (except Administration) are Resource Containers. Resource containers are created by the *Variable Editor* application 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

Moving application between containers

Moving applications between containers are necessary during DTAP promotion or environment consolidation. When moving between containers there are some considerations that sometimes need to be addressed.

What in the framework are applications depended on?

All container base structures are identical and applications usually work flawless across containers but there could be dependencies in the containers that need to be addressed, like:

- Connection string, resided in *vG.ConnStringPath*
- QVD and other sub folders, example *\$(vG.QVDPath)MetaData*
- Custom Global Variables (created by **Variable Editor**) need to move to the new container.
- If running a *Binary* load statement, (if needed) change relative path in the script.
- Connections to Shared Folders container could need scripts/data from the original Shared Folders container
- Global Variable links (created by *LoadContainerGlobalVariables*) to other containers need to be added in the Container Map if promoting to a new environment.

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*
- **NorthWindExampleMartApp** Is an end user application that binary loads from *0.Example_Northwind_Mart*
Calendar-Example application that shows how the CalendarGen sub function works
Resides under *1.Example\1.Application\4.Calendar-Example*
- **5.QVD-Migration-example** demonstrates the power of using the QVDMigration sub function
Resides under *1.Example\1.Application\5.QVD-Migration-example*
- **Qlik Mart** based on Northwind and QVD generator qvd's from $\$(vG.QVDPath)\0.Example_Northwind$
Resides under *1.Example\4.Mart\0.Example_Northwind_Mart*
- **Connection String** Include example to Northwind Access database
 $\$(Must_Include=\$(vG.ConnStringPath)\0.example_access_northwind.qvs);$
- Northwind Access database, resides under $\$(vG.ConfigPath)\0.Example_Northwind$

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 can also be edited with Variable Editor.

Variable Editor is found under *6.Script\2.VariableEditor\VariableEditor.qvw* in the *0Administration* container.

But can also be executed by using the Variable Editor Shortcut.

 **VariableEditor Shortcut**

Variable Editor with Qlik Sense

There is for the moment no variable editor for Qlik Sense. As Variable Editor uses an imbedded license it works with QlikView personal edition without the need of a QlikView license, personal edition can be downloaded for free. This means that Qlik Sense administrators can maintain QDF without additional costs.

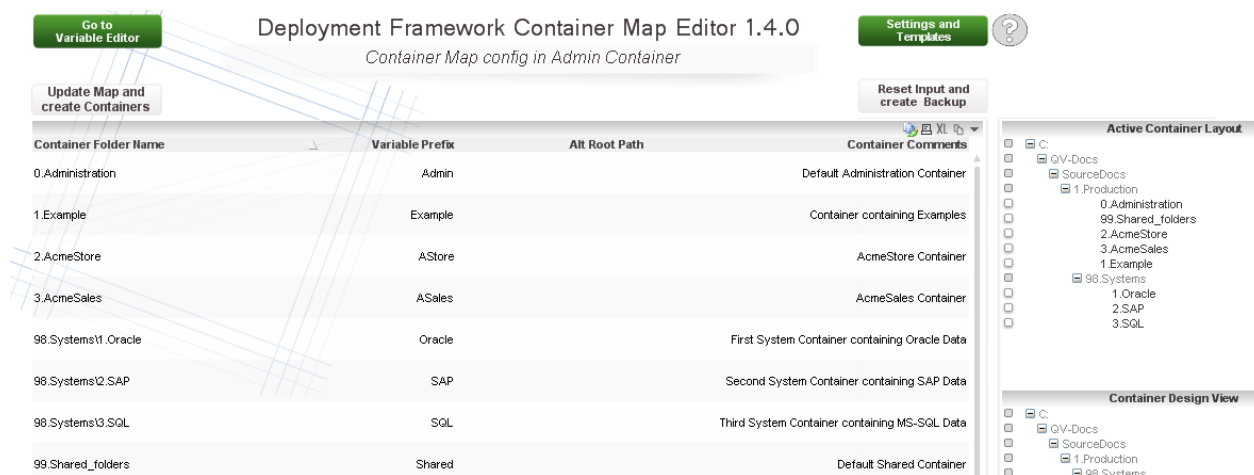
Help ?

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

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 find the containers and to create container connection variables. These variables are created when using the *LoadContainerGlobalVariables* function, example:

call LoadContainerGlobalVariables ('HR'); Will connect to 1.AcmeHR container (if exist).



Container Folder Name	Variable Prefix	Alt Root Path	Container Comments
0.Administration	Admin		Default Administration Container
1.Example	Example		Container containing Examples
2.AcmeStore	ASore		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

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

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 in the same level as *0.Administration*
Example 2: *98.System\1.Oracle* to create a container under a system folder. To add a container in another file system.
- *ContainerPathName* , enter prefix share variable names in *ContainerPathName* field, example *Oracle*.
- *Alt root path*, Edit an optional container path in *alt root path* field. A container could also be copied in a subfolder structure the subfolder name will be created automatically.
- *Container Comments* Is descriptive Meta Data regarding the containers, very good to use for documenting the solution.

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.

Update Container Map

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

Update Map and
create Containers

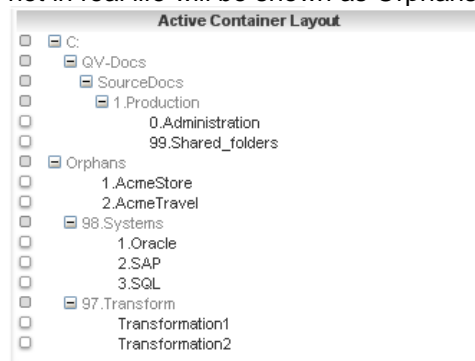
Create New Containers option

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. New Containers can only be created from the *0.Administration* container this means that the selected and applied container either is *vG.BasePath* or *vG.AdminBasePath*.



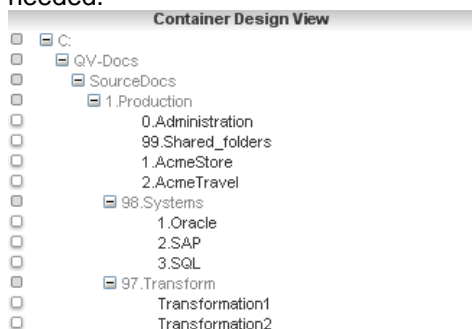
Active Container Layout

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:

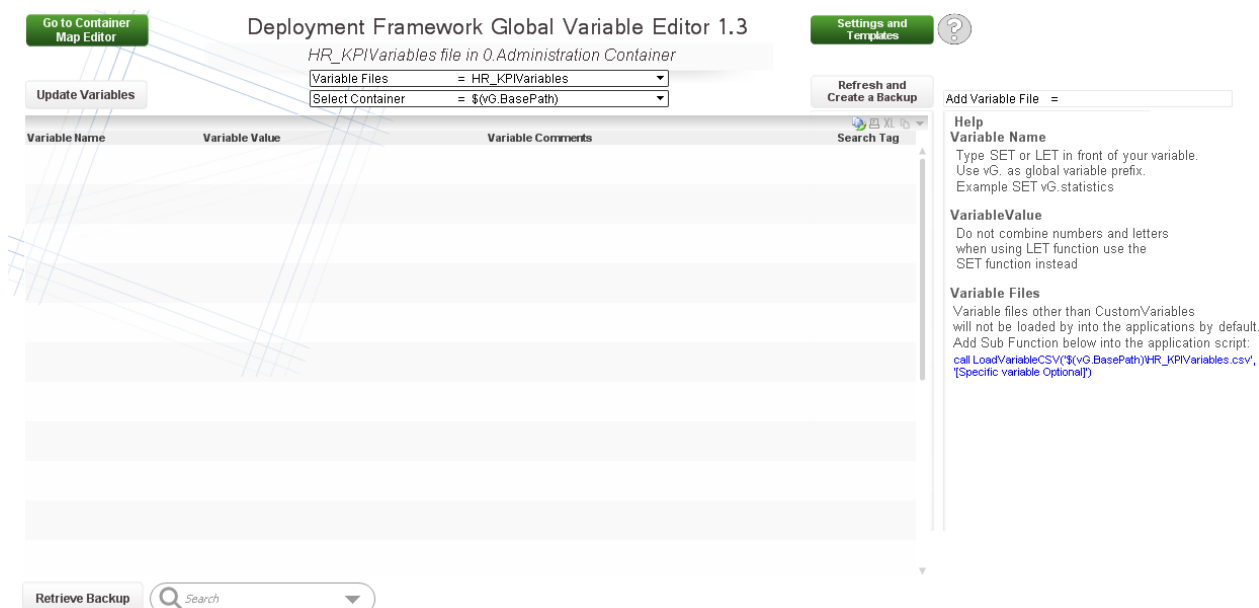


Container Design View

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**.