

AQLIK

A QLIK FOCUSED BLOG

ABOUT ME



Back end, English

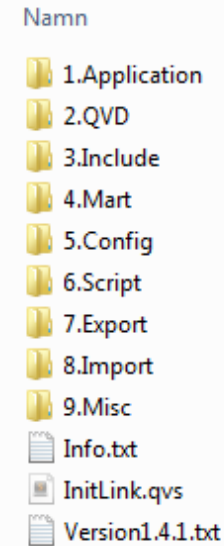
THE 9 FOLDERS OF QLIKVIEW DEPLOYMENT FRAMEWORK

Written by **Vegar Lie Arntsen** on **November 18, 2014**. **2 Comments**

The main building block of the QlikView deployment framework (qdf) is the resource container. Every environment using QDF will be build using only one to many instances of this

homogeneous structured entity.

When first introduced to the QDF the first thing you'll notice is the container folder and it's 9 folders structure. I'll explain which they are and what their purpose in the section below.



Namn

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

Info.txt
InitLink.qvs
Version1.4.1.txt

THE 9 FOLDERS

The folder names in a QDF container give you a good indication on what they contain. They are sense quite self-explanatory and after reading the section below you'll be ready to start working with them.

1. APPLICATION

You will place all your QlikView (.qvw) or Qlik Sense (.qvf) applications in the root or in sub folders of

1. Application. Only these two file types you should be kept in this folder. Prj-folders and version control files are exceptions that will reside inside 1.Application.

2. QVD

The path for QlikView data files. This folder will work as the source path where your application reads qvd files and/or the folder where you store qvds into.

3. INCLUDE

The folder where all .qvs files that is to be included into your QlikView or Qlik Sense script resides. This could be qvs files you create yourself or the pre-made scripts that are bundled with the framework.

The key to QDF initiation is found in a sub folders of 3.Include It is called `init.qvs` and is absolutely necessary to start using and take advantage of the QDF framework. I recommend all developers to click through these folders to get an impression of all shipped features that comes with the QDF.

Variable files and connections strings are typically placed in another sub folder to 3.Include. If you are using QlikView Components (QVC) then the `Qvc.qvs` file would be placed within this folder.

4. MART

If you want to create a data mart to use in further data discovery or as the base in a power user sandbox then you store mart files in the 4.Mart. I would recommend to pre-model your mart in a qvw file for this folder. A set of carefully modelled qvd files could also do the trick, if a `Select * From $(4.Mart)*` would give me a correct data model.

5.CONFIG

The 5.Config is the home of custom configuration files such as language setups and control documents. The Local config.qvs of QVC is a good candidate for this folder.

6.SCRIPT

The place to store script files. It could be script files used by the QlikView publisher or by Windows scheduled tasks. This is also a good place for scripts that you manually run for making adjustments or manipulations to your container content.

7.EXPORT

The 7.Export path is where you put files generated by QlikView but is to be consumed by another system. Ex. csv or xml files for export to a web service.

8.IMPORT

The 8.Import folder like the 7.Export is intended for data transfer between QlikView and other systems. This folder is used by the other systems to store their output files. Eg. the response of a web service.

9.MISC

The path for miscellaneous files that you do not fit into one of the above. It could be documentation, document and object extensions used by applications in the container.

All folders and sub folders in the QDF contain a Info.txt file. This file contains a short description of the purpose of the folder and the global path variable that will be available while scripting.

GETTING STARTED

Lets say you got your empty container. How do you start?

The key for initiating the QDF is to include the mandatory `init.qvs` file into your script. in the `3.Include\BaseVariable` folder with relative paths like this.

```
$(Include=..\3.Include\1.basevariable\1.init.qvs);
```

Depending on which sub folder level your application is located within the 1.Application folder it is considered a good practice to expand the include statement to cover multiple alternative relative paths such as this.

```
$(Include=..\..\3.Include\1.basevariable\1.init.qvs);  
$(Include=..\..\3.Include\1.basevariable\1.init.qvs);
```

```
$(Include=..\..\3.Include\1.basevariable\1.init.qvs);  
$(Include=..\3.Include\1.basevariable\1.init.qvs);
```

By running this script you will initiate the QDF and you will have a set of variables generated into your application. Some of these applications are essential for the framework.

CONTAINER PATHS

The initiation will generate variables for all your container folders and standard sub folders mentioned above. If you use these variables instead of hard-coded paths in the script you won't need to be concerned about where your container is located at the moment you run your applications.

SHARED AND CUSTOM VARIABLES

If you or someone else have defined shared and/or container specific variables then these will be loaded automatically.

USING THE FOLDER STRUCTURE WITHIN A CONTAINER

If you have initiated the QDF and are ready to go. What's different when scripting?

Developing in a QDF environment doesn't need much change in your scripting methods. To get started you should learn to use the path variables instead of absolute or relative paths.

```
LOAD *  
FROM C:/QlikViewFiles/MyContainer/2.QVD/MyData.qvd;  
  
LOAD *  
FROM ../2.QVD/MyData.qvd;  
  
LOAD *  
FROM $(vG.QVDPATH)MyData.qvd;
```

The QDF also supports fetching variables from other containers. You will notice that per default you will have variables for the Shared container defined for you. You may also import variables from other containers using the following function.

```
Call LoadContainerGlobalVariables('TransformContainer');
```

Details on how to set this up is not covered here but is well documented in the QDF documentation.

SUMMING UP

The main difference between QDF and traditional deployment is in the structure. All containers in a QDF environment have the same folder structure and consists of folders with designated functions. This structure does not have to affect the way you script. You are safe within the QDF as long you


1. remember to initiate it with `init.qvs` in every application using relative paths and
2. use the generated container variables instead of absolute and relative paths in the rest of your script.

Following these directions will not make you a better QlikView or Qlik Sense developer it will make you survive in the QDF while familiarising yourself with the structures and the functionality that the QDF has to offer.

I'm convinced that the QDF can make improve you and your organization's QlikView and Qlik Sense developer and administration capabilities. Start of simple by

1. Following my two points above.
2. Start exploring the folder structure and its content.
3. Read the QDF developer guide that is included in the QDF.
4. Join and participate in the Qlik Community Group: [QlikView Deployment Framework](#)

I believe that you will come a long way with these measures. If you need more help or support contact an QDF expert. If you don't have one in your organisation then your Qlik Partner.

 architecture container deployment framework qdf qlikview qvc qvs sense

Blogroll week 46

QlikView 11.2 SR10B

2 COMMENTS ON “THE 9 FOLDERS OF QLIKVIEW DEPLOYMENT FRAMEWORK”

**Nishant Dialani** says:

January 10, 2016 at 07:47

Hi,

Thanks so much for sharing this. It seems to be helpful for me as we just adopted Qlikview in our organization.

I have a question regarding publishing the files created by developers. Do they have to copy the file from their local machine send it to administrator and then administrator will copy the file in the folders you suggested?

Or developers themselves will have access to qlik server machine to copy the files themselves.

Nishant

Reply**Vegar Lie Arntsen** says:

April 5, 2016 at 09:07

Hi Nishant,

I'm sorry, but your comment got stuck in the blog comment spam filter.

My suggestion is that you let the developers get a copy of the whole container. When the developer is

finished with the changes an administrator can move the changed files (or the whole container) over to the test, acceptance or production environment.

I often recommend my clients to use some kind of version control in their Qlik development environments to keep track on which files that have been changed.

Reply

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

9/18/2018

The 9 folders of QlikView Deployment Framework – aqlik

Website

Post Comment

Proudly powered by WordPress | Theme: Nordic by Sandpatrol.
