Qlik Deployment Framework

Deployment Guide

March, 2015

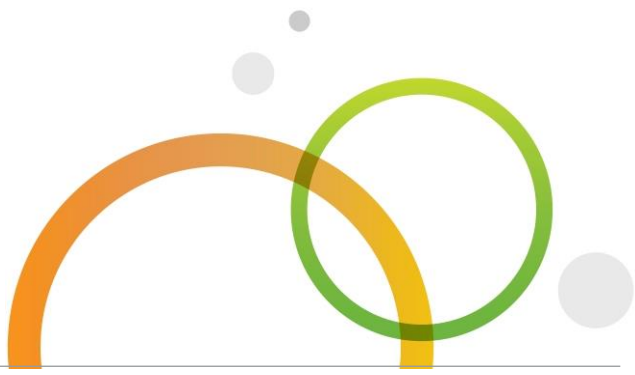


Table of Contents

Deployment Guide v1.5.1 3
Business Intelligence Competency Center (BICC) 4
Deployment process 4
DTAP acronym 4
Environment descriptions 5
Process Workflow 7
Container Workflow 8
Process identification Workshop 9
Deployment Framework implementation steps 10
Application Certification Process 11
QVD Strategy 13
QVD Strategy Overview 13
Indexing using Qlik Deployment Framework 1.5 13
QlikView Index Monitor in Administration container 13
Self Service BI 14
QlikView Self –Service flavors 14
Data Discovery Self Service by using Containers 16
QVDMigration Sub Function 17
Self Service process example 17
Container Security 18
Qlik IDE Development environment 19
Overview 19
Automatic Performance Tests 20
Tool for easy creation of load/performance tests of QlikView 20
Customer Case 21
DTAP Process 21
Platform consolidation 22
Project Methodologies 23
SCRUM Methodology 23
RAD/DSDM Methodology 24
Development 26

Skill sets 26
Qlik Development Teams 27

Deployment Guide v1.5.1

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

Business Intelligence Competency Center (BICC)

For clearer governance best practices, processes, roles, responsibilities and ownership (QlikView platform, framework containers and applications) needs to be established. The group responsible for this task is usually called BICC (Business Intelligence Competency Center) and should be a cross-functional team to ensure involvement from all stakeholders. Gartner started advocating that companies need a BICC to develop and focus resources to be successful using business intelligence.

The need for a competency center arises when the sharing of central services, expertise and governance become practical and cost effective. This can happen with distributed development teams, business development teams and large development teams that are either cross-functional or not co-located.

Qlik Competency Center (QCC)

Qlik Competency Center (QCC) can be a component of a BI Competency Center (BICC), with a tool focus on Qlik. This allows the QCC to access the BI shared services and resources of the BICC, but still retaining a tool-focus on Qlik services and resources. A virtual or dedicated team that provides centralized Qlik services to teams developing, supporting and using Qlik applications. QCC is also responsible for Qlik Deployment Framework methodology implementation. This document provides the guidance and subjects for the QCC.

Deployment process

DTAP acronym

The acronym DTAP is short for Development, Testing, Acceptance and Production. It is a rather common acronym in ICT describing the steps taken during software development.

This is the sequence:

- The program or component is developed on a Development system. This development environment might have no testing capabilities.
- Once the developer thinks it is ready, the product is copied to a Test environment, to verify it works as expected. This test environment is supposedly standardized and in close alignment with the target environment.
- If the test is successful, the product is copied to an Acceptance test environment. During the Acceptance test, the customer will test the product in this environment to verify whether it meets their expectations.
- If the customer accepts the product, it is deployed to a Production environment, making it available to all users of the system.

The set of environments used for a DTAP cycle is often called a DTAP street. (Wikipedia)

Setting up DTAP environments are a step toward establishing controlled governance across all BI projects. In addition to the code artifacts, important information includes who requested a change, whom reviewed and approved the item, requirements and design discussions and decisions, and implementation and test plans and details.

Environment migration —The software change management process controls migration patterns (that is, how a given set of code moves from one environment to another). While large organizations may have automated solutions to control the migration process, others do it manually. In either case, it's critical that the transitions and approvals along the path be well-defined and that the data to support traceability and accountability be preserved.

Emergency changes —Programmer access to production for emergency or "hot" fixes is still found, but it is hazardous and difficult to manage. The access has to be temporary and revoked immediately on the completion of the fix. Alternatively, some organizations grant the temporary access to a separate fix-test environment or to the preproduction environment. Once the patch is ready, production control personnel can deploy it through standard mechanisms to production.

Complexity —The more environments you own, the more complex the process of making changes is, the more organizational challenges you have, and the more difficult it is to be flexible to changes in the project because shifting requirements or defects are discovered.

Environment descriptions

Development

This is the working environment for individual developers or teams. Working in isolation the developers can try
radical changes to the code without adversely affecting the production.

Test

Test Environment is a hardware and software environment in which tests will be run. Software that interacts with Qlik during the tests like database drivers must also be available.

Sandbox or Self-Service

These environments are used for distribution of non certificated applications to a smaller audience, like a group of controllers. Sandbox/Self-Service is also used for application concept validation. A sandbox application could either use live data or use test data depending on application validation purpose. Self-Service is usually more focused on business and Sandbox on IT. Read more under Self-Service section.

Acceptance (Pre-Production)

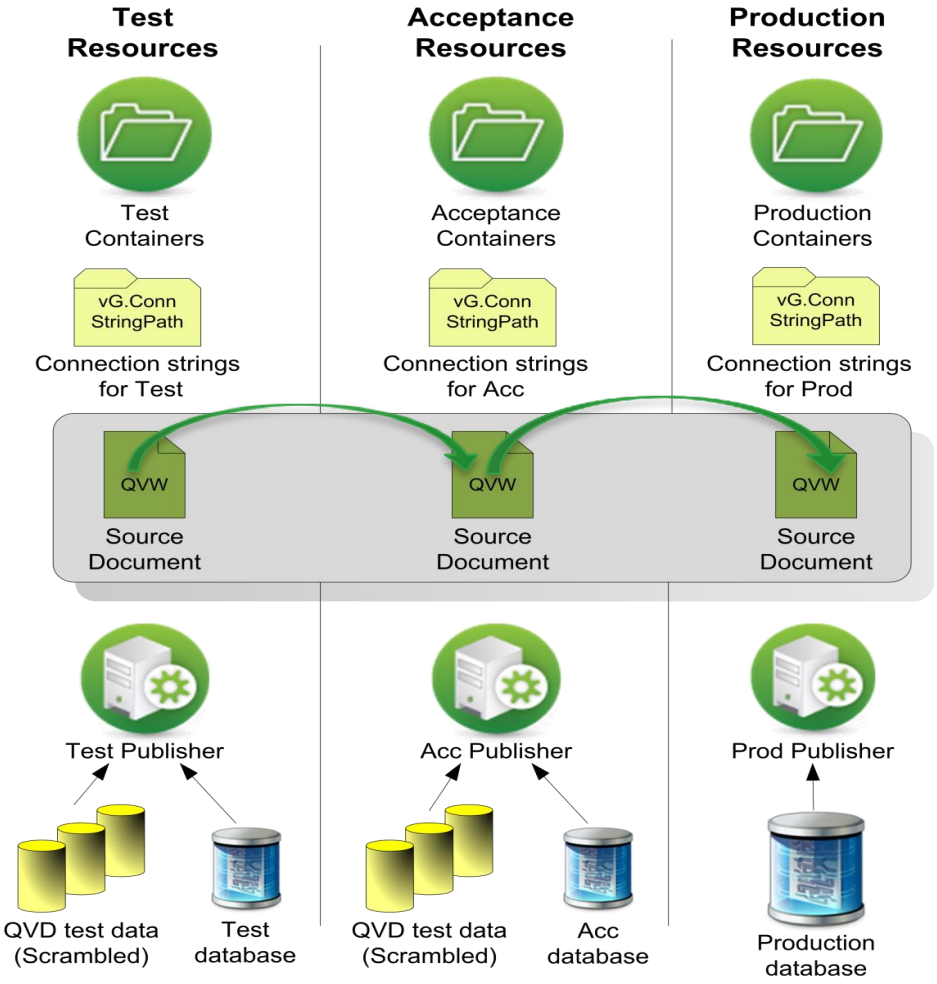
System Acceptance is the point in the lifecycle at which every aspect of the application, along with data connections and other routines and system utilities are thoroughly validated prior to proceeding with production Implementation. Acceptance server is a mirror copy of the production server; its primary purpose is to test the completed application on the mirrored copy of production to ensure that the application doesn't break the existing production server applications. No actual code development should ever take place here, only minor tweaking of OS parameters or application settings.

Production

The production environment is the set of resources and controls directing them to provide a "live" service, in our case QlikView and Qlik Sense.

Qlik deployment DTAP resources

This is an overview of what recourses a Qlik DTAP environment consists of. When moving applications between the environments application will automatically connect to different resources (like data sources) when moving applications between the DTAP environments.



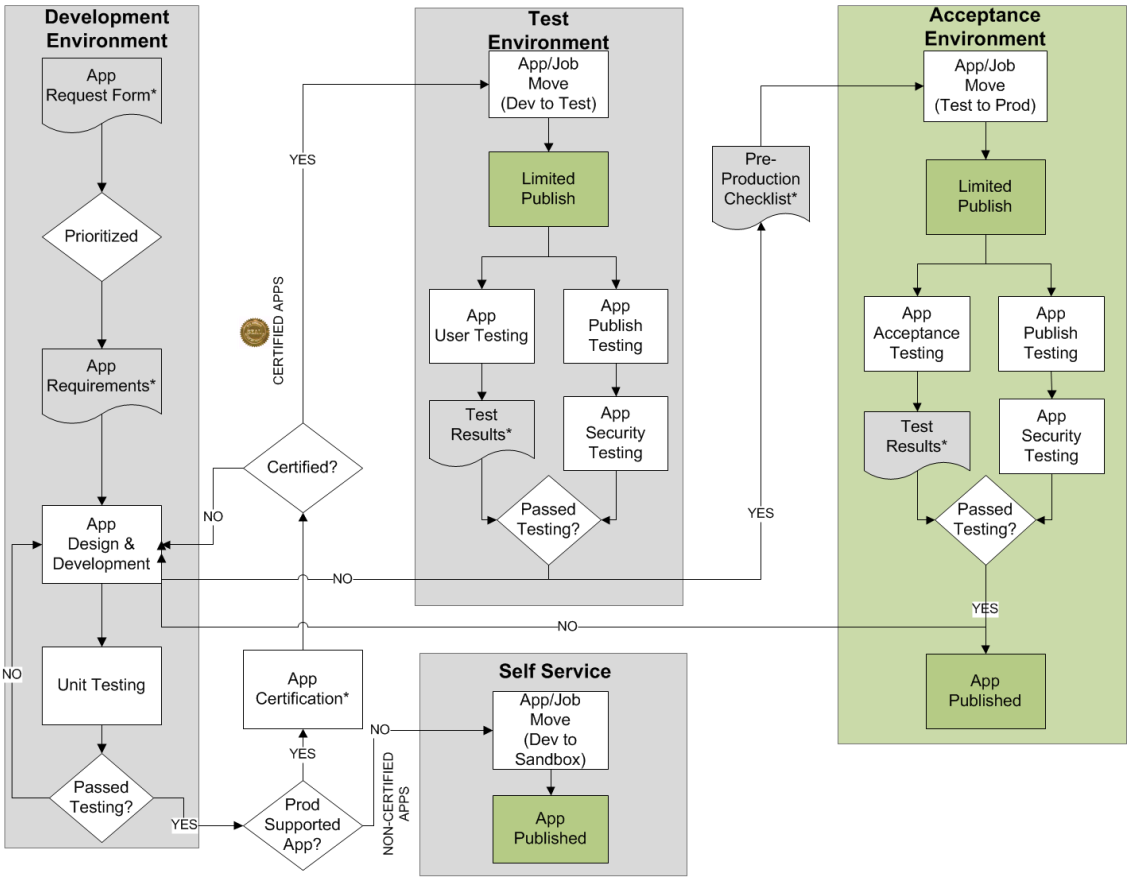
Resources needed in a DTAP promotion process

Resources:

- **Containers**
  Use different containers for test, sandbox, acc and production.
- **Connection string includes**
  For QlikView use vG.ConnStringPath folder, same name but connection strings inside that connects to different databases.
- **Data Sources**
  The DTAP process will require that database designs are equivalent for test, sandbox, acc and production. In lack of good test data create scrambled QVD files instead, use include-files for qvd loading.
- **Qlik Applications**
  The application is the only component that is moved between environments.
  When moving between the environments the application will connect to environmental specific data sources depending on the correlating connection string.
- **Security boundaries**
  Groups, roles and policies to separate the environments and responsibilities

Process Workflow

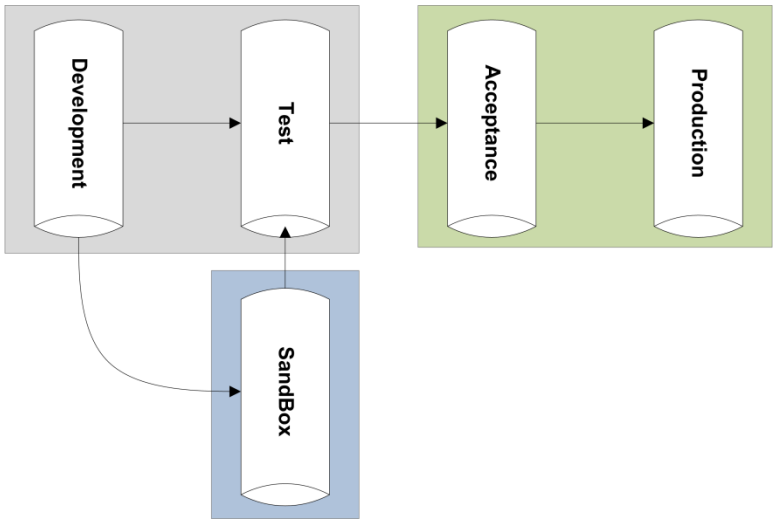
For a good DTAP process a workflow describing the handover steps and certification process is needed. This example describes a complex workflow with development, sandbox, acceptance and production environments.



Recommendation is to create a custom workflow depending on the company needs, see ***Process identification Workshop*** down below.

Container Workflow

Deployment Framework containers should be in alignment with the DTAP workflow. Security (groups) and/or physical (disk areas) boundaries added to containers/environments according to the workflow.



This container structure example is aligned with the development workflow in the example above.

In this example there are three security boundaries one for development and test (grey), one for acc and production (green) and one for the SandBox environment (Blue).

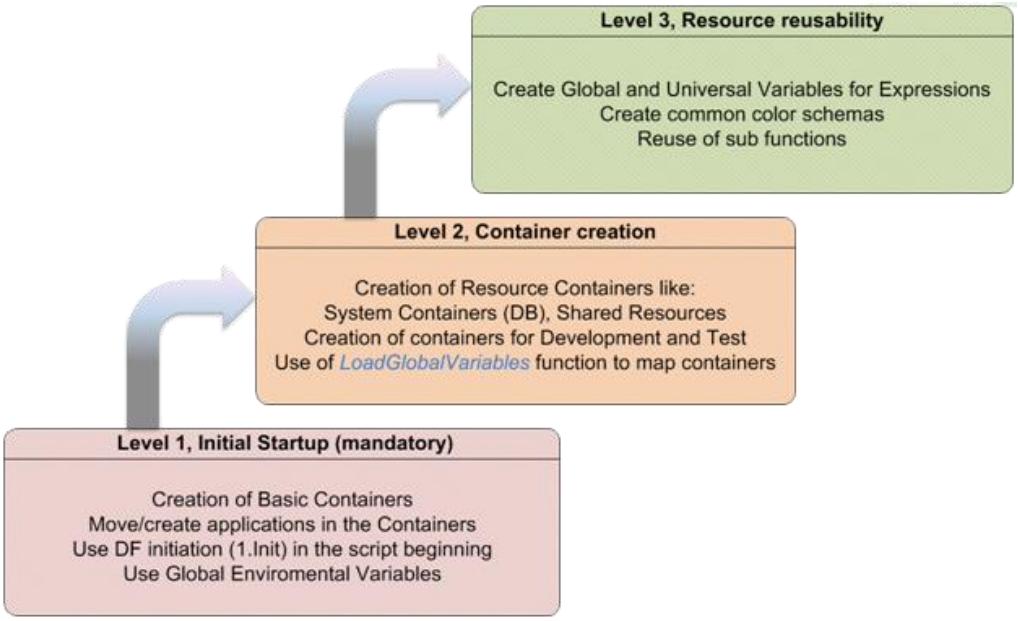
Process identification Workshop

To identify the development/deployment workflow a good idea is to get stakeholders involved in a Process identification workshop, aiming to identify the Qlik deployment process document and align it to QDF.
Below is a workshop content example.

- Identify the DTAP process
  1. Use White board or Post It to create a DTAP process
  2. Identify Application Certification Process that is aligned with the DTAP process
  3. Identify roles and responsibilities
- Align the framework to the processes
  1. DTAP Release process
  2. New projects/departments/self service/Sand Box
  3. Define relevant container setup
  4. Set up the initial container structure using the Variable Editor in Deployment Framework.
- Define the security model
  1. Define naming standard for container security groups in Active Directory based on the Identified roles and responsibilities.
  2. Define the initial Active Directory Groups to secure the initial setup.
- Exemplify Application development in the context of the framework
  1. Application Development Basics, using QDF environmental Variables is Mandatory
  2. Other possibilities,
     a. Global Variables
     b. Universal Variables
     c. Functions
- Exemplify System Administration in the context of the framework
- Task Naming Standards when creating tasks
- Exemplify Change Management in context of the Framework
- Renaming a container/moving a container
  - Review solution based on the Deployment Framework (in a later time)
    1. Based on the common understandings and documentation from first workshop and the guidelines in the Deployment Framework, review the ongoing development.
    2. Address new questions regarding the framework that may have arisen during application development.
    3. Use additional QDF functionality if needed
       - Global Variables
       - Universal Variables
       - Function library
       - System Variables and monitoring

Deployment Framework implementation steps

The Framework does not need to be implemented all at once, but there are some mandatory functions that are needed to start utilizing the power of the Framework. This is shown below Level 1, Initial Startup.



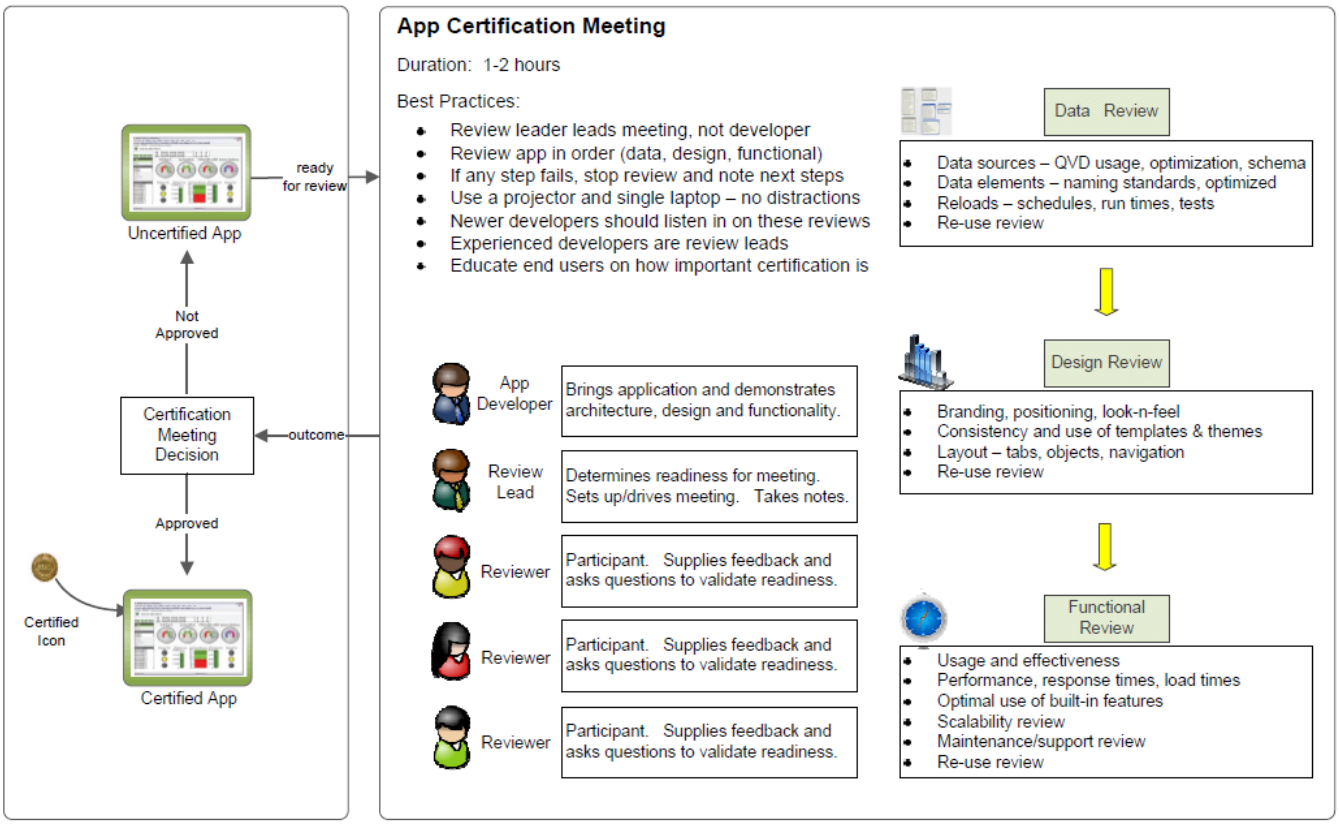
Deployment Framework CMMI model.

Steps two and three are not mandatory in the initial implementation phase, but highly recommended to get the full power out of the framework.

Application Certification Process

Application Certification Process serves as a tollgate to getting a Qlik application “certified”. Certification means an application has gone through this process and been approved. Certification should be a part of the development workflow.

A “Certified” icon is then placed in the title section of the application so that users and support teams know which applications are certified and which ones are not. This allows teams to place emphasis on this process by not providing the same level of support for non-certified applications.



Example of the Application Certification review process

Application Certification review

- The review leader leads the review process, not the developer.
- Review app in order (data, design, functional)
- If any step fails stop the review and note the next steps
- Use a projector and a single laptop – no distractions
- Newer developers should listen in on these reviews to learn the process
- Experienced developers or project managers are review leads
- Educate end users on how important certification is

Roles

- **App Developer**
  Brings application and demonstrates architecture, design and functionality.
- **Review Lead**
  Determines readiness for meeting. Sets up/drives meeting and take notes.
- **Reviewers**
  Participant. Supplies feedback and asks questions to validate readiness.

Review Criteria's

- **Data Review**
  Data sources – QVD usage, optimization, schema
  Data elements – naming standards, optimized
  Reloads – schedules, run times, tests
  Re-use of data
- **Design Review**
  Branding, positioning, look-n-feel
  Consistency and use of templates & themes
  Layout – tabs, objects, navigation
- **Functional Review**
  Usage and effectiveness
  Performance, response times, load times
  Optimal use of built-in features
  Scalability review
  Maintenance/support review
- **Compatibility review**
  Is the Deployment Framework best practice used
  Is DF variables used for easy DTAP promotion process

QVD Strategy

To handle infinite amounts of data using Qlik we need to store data into Qlik Data (QVD) files, Qlik then reads QVD data in millions of records per second. This means that instead of loading all available data into Qlik, we only load needed data. This seems simple at first, but for this we need to understand what data we have and data needed? This identification should also be done in a self-service way to keep time to delivery at absolute minimum. In Qlik Deployment Framework 1.5 (QDF) QVD index functionality has been added to support "self-service" delivery of data.

QVD Strategy Overview

So what is a QVD strategy and how do I create one? A strategy is documented definitions explaining how data is handled and identified (by Qlik) from extraction to delivery. Here is a simple strategy example: Orders data should be stored into container *Sales* under folder *2.QVD\Orders*, the QVD names should be based on dates and tables *2014-01-01-Orders.qvd*, all QVD's should have x,y,z tags attached to it and data files older than 24 month should be deleted. There are also more complicated strategies involving staging, aggregation, history and security. Without a QVD strategy maintaining, identifying and thereby reuse of data becomes much more cumbersome and self-service QVD delivery become impossible.

Indexing using Qlik Deployment Framework 1.5

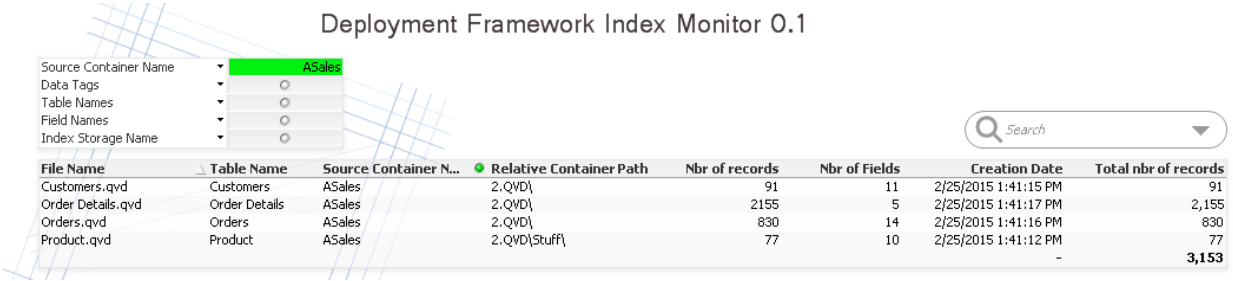
With help of the index functions the strategy examples above can easily be indexed and maintained. The index functions create and maintain a set of indexes for Qlik data files and works with both QlikView and Qlik Sense. These indexes are used when searching for data types across multiple QVD files this means that developers and power users select needed data using a simple command. Finding the data is done autonomously by the system in the background. The index is stored in one single location (*vG.SharedConfigPath*) while the qvd's can be spread out across the environment depending on security or organizational considerations. Index functions implemented are:

- **IndexAdd** Will create QVD indexes, should be done during qvd creation.
- **IndexLoad** Loads Qlik data based on combination of index criteria's like file name, tags, table, fiels…
- **IndexDel** Delete index and optionally referring QVD file.

Read more on the Index functions in Development Guide

QlikView Index Monitor in Administration container

A simple Index monitor application to monitor current Indexes is available under container *0.Administration/3.IndexMonitor.* The app will load in all index meta-data available in the environment (as long as index been added for the QVD's).



File Name	Table Name	Source Container N...	Relative Container Path	Nbr of records	Nbr of Fields	Creation Date	Total nbr of records
Customers.qvd	Customers	ASales	2.QVD\	91	11	2/25/2015 1:41:15 PM	91
Order Details.qvd	Order Details	ASales	2.QVD\	2155	5	2/25/2015 1:41:17 PM	2,155
Orders.qvd	Orders	ASales	2.QVD\	830	14	2/25/2015 1:41:16 PM	830
Product.qvd	Product	ASales	2.QVD\Stuff\	77	10	2/25/2015 1:41:12 PM	77
						-	3,153

Self Service BI

By utilizing Self Service BI business will become more self-reliant and less dependent on the IT organization. Answers to questions will be delivered faster than traditional BI where cubes/views or data marts need to be created by IT before finally creating and delivering “reports” to the business, this traditional process takes month and most important often will not provide the insight that business requires. Qlik Self Service model lets business them self-create applications while IT delivers controlled and secure data. Exact level of IT deliverables are different from case to case, this could be raw or cleansed data, QlikMart data model, predefined expressions and templates containing initial scripts (including DF) and graphical profile

QlikView Self –Service flavors

There are primarily three different QlikView Self-Service flavors.

- QlikView Applications using Dynamic Tables and Charts
- Shared Objects, meaning that users can create custom objects in the UI
- Data Discovery – “Sandboxing”

QlikView Applications using Dynamic Tables and Charts

One can definitively argue that a QlikView application itself is a self-service enabler. The tool has standard functionality for “cycling” dimensions and measures in a flexible way – a single chart can give a standard user with no special training the opportunity to combine a huge set of dimension attributes with multiple metrics.

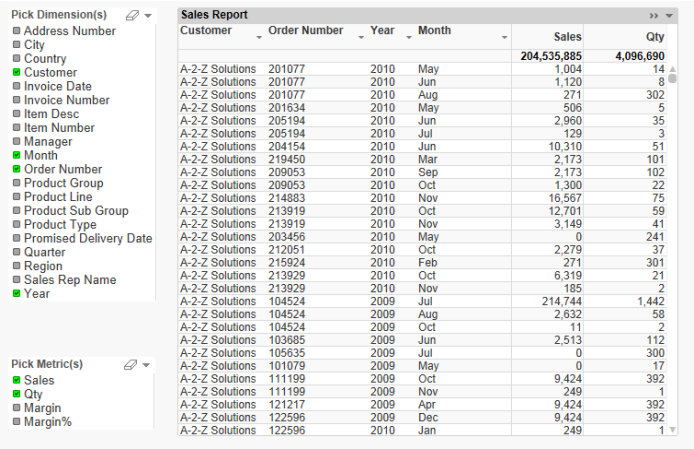
A chart or table containing a “cyclic” group with 10 different dimensions attributes and the possibility to “cycle” between 5 metrics can give the user 50 different views within the same chart.

Self-Service enablement should be considered also during QlikView application development.

The QlikView data model is extremely powerful since the number of dimensional attributes (fields) that can be loaded into a single QlikView application is huge without the typical challenges of data explosion. Any field in QlikView can be selected to slice the data in subsets. This is also a self-service enabler that is related to the tool itself and the QlikView application development.

A very common use case related to self-service is that users on a very ad hoc basis want to add a number of dimension attributes, for example Customer, Year, Month, Order No to a chart and combining it with one or more metrics - a type of “dynamic chart”.

This use case can be addressed during the QlikView application development where a developer needs to “enable” this scenario by building this functionality into specific charts or tables.



The benefit of this is that it does not require any training and only relevant combinations of attributes and metrics can be pre-defined by the developer – no risk of doing something wrong for the user.

It is a simple approach that in many cases covers requirements of self-service without the business users having to build or develop anything - everything is pre-built into the QlikView application and no new content is created that needs to be maintained.

All these examples show that thinking about self-service should start already during the QlikView application development.

Relevant role:

- QlikView Standard User

Self-Service location:

- Server Side

Shared Objects

The concept of “Shared Objects” in QlikView is also a self-service enabler. If allowed, users can create personal charts and tables using existing metrics or even defining personal metrics.

This type of self-service needs training and should not be for everyone.

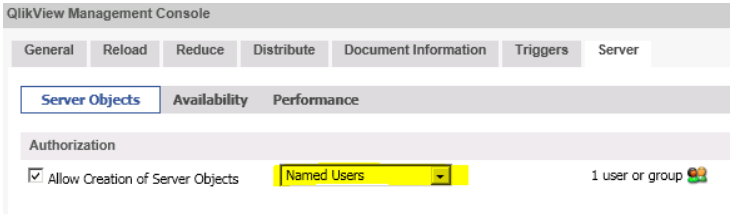
Compared to the scenario “Dynamic Chart” described earlier Shared Objects introduces much more complexity.

- The user needs to build something and access properties
- The user needs to write metrics or copy metrics
- The user can combine attributes and metrics that gives the wrong result
- Content needs to be stored outside the QlikView application itself (.shared file)

If the requirements are to combine pre-built metrics with any number of dimension attributes, the approach with Shared Objects represent an unnecessarily complex way of doing the same thing that could be achieved with the “dynamic chart”.

The creation of shared objects should require training and the feature should only be enabled for the users that qualify as “QlikView Advanced User” - A poorly built chart or table could consume

unnecessary high amount of server resources. *Shared Objects* restriction be set on each specific QlikView application in the QlikView Management Console, by only allowing “Named Users” to create shared objects or by completely disabling the option.



Adding these “advanced users” to an Active Directory group and authorizing the group when setting up the Publisher task would be a way of restricting the creation of Shared Objects only to trained users.

Relevant role:

- QlikView Advanced User

Self-Service location:

- Server Side

Data Discovery – Self Service (Sandboxing)

Data Discovery or “Sandboxing” is not a part of an actual development process. The objective is to give business users the opportunity to access, combine and analyze data to get new business insights. The process is about being creative, explore and test business theories.

Data Discovery Self Service by using Containers

The flexibility of Deployment Framework Container architecture enables Self Service BI capabilities. A container is *isolated and self-contained* and thereby perfect as a Self Service enabler and repository.

Distributing QVD files and data marts to dedicated Self Service containers the sandboxing will not have any impact on source systems and will gain from the fast load speed of the QVD and QVW binary load which enables short development iterations without having to wait – even with large data volumes (a QVD file can typically be loaded with 2-3 million records per second when optimized).

Create containers and security groups in a separated repository (by using *Alt root Path*) for each new area of Self Service, this will create isolation from data storage container. Recommendation is to first use a Self Service validation container, used for qvd and QlikView mart data quality validation so that IT does not distribute faulty data.

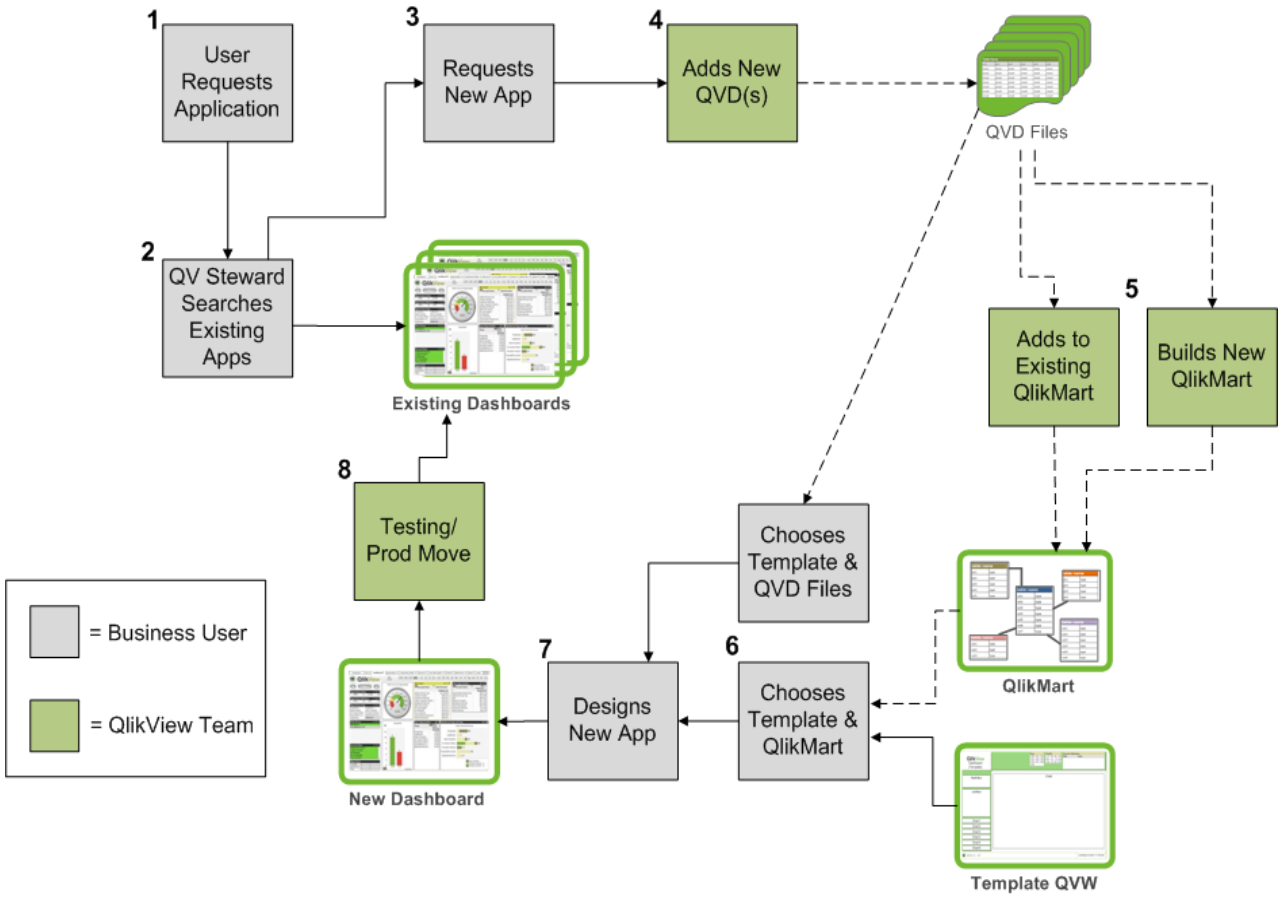
After data Validation Container path (*Alt root Path)* is changed to the front end storage container. The DF core will automatically redirect the back end qvd distribution to the new path.

Container Prefix	Container Folder Name	Comment	Alt root Path
Admin	0.Administration		
Shared	002.Shared		
DB1	*1.System\1.DB1*	*Data storage container push data to Self Service Containers*	
ValidationSelfService	*2.ValidationSelfService\1.SelfServiceContainer*	*Self Service Validation containers*	
.SelfService	*\\FrontEnd\SelfService\1.SelfServiceContainer*	*live Self Service containers in Front End*	*\\FrontEnd*

QVDMigration Sub Function

To simplify QlikView data (qvd) movement between data storage container and Self-Service containers the pre-defined sub function *QVDMigration* exists. The function makes it easy to copy multiple data files (qvd) from data storage containers to Self-Service containers and at the same time scramble selected fields, all this done with one single line of code. Read more regarding functions in the Development Guide

Self Service process example



Process flow

User requests an Application (1) and a steward (2) checks if a current application easily could be modified to fit the request, else a New App Request process/project (3) will start. A new Self Service Container will be created and user access granted for the new application process/project. The Qlik Team prepares Qvd load from back end system container or containers will push requested data to the Self Service Container (4). Optionally a Qlik Mart (4) will be built and added into the Self Service Container. The business will develop the application based on Template, mart and/or qvd files (6-7). Qlik Team will certify application before release (8).

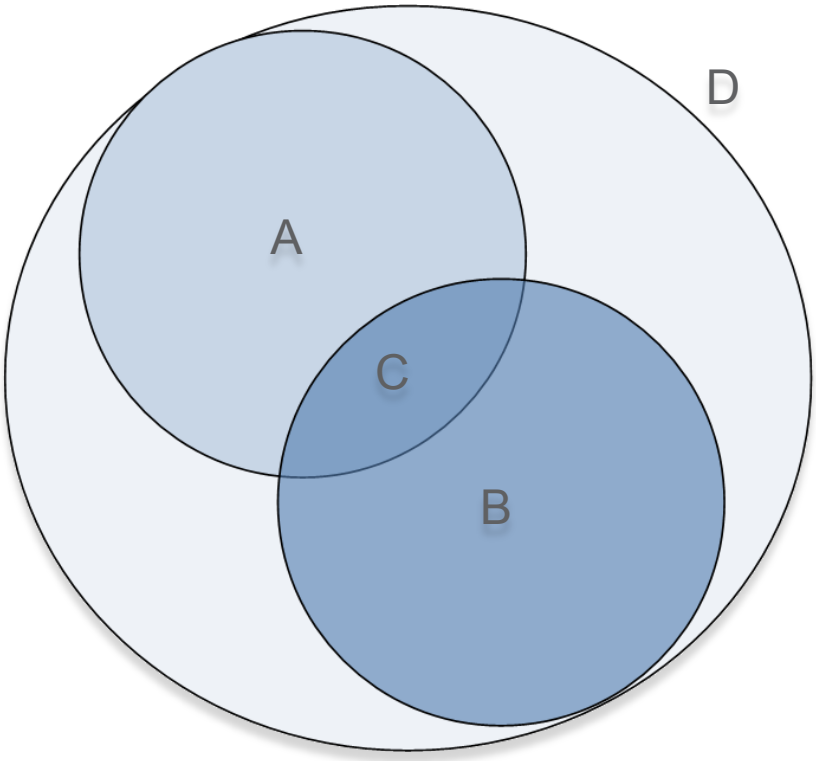
Container Security

Containers in the DTAP workflow should have different security settings so that the handover process will be

Respected and used.

All containers are independent from each other and could be in separate Active Directory administrative security groups. Important is that a global security group surrounds all the containers, super admin and Qlik service accounts must reside in this group.
Example:

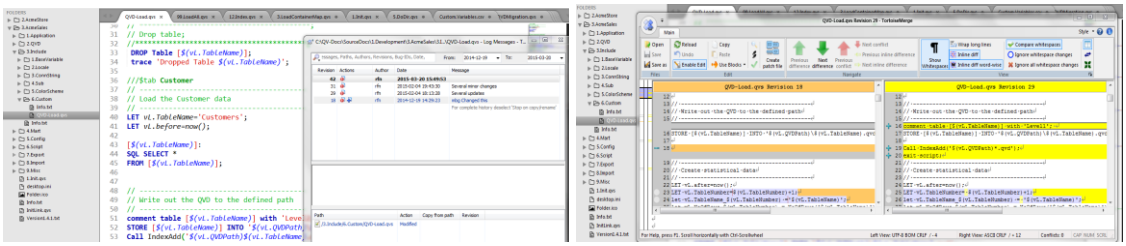
A) Security group for Container A (Development)
B) Security group for Container B (Sandbox)
C) Shared Folder security group where both Container A and Container B have access (Shared Folders)
D) Super Admin security group, this group has full access to the root folder above the containers. The QlikView service accounts must be member in this group. The Super Admin security group should be a member of the local QlikView Administrators group.



Qlik IDE Development environment

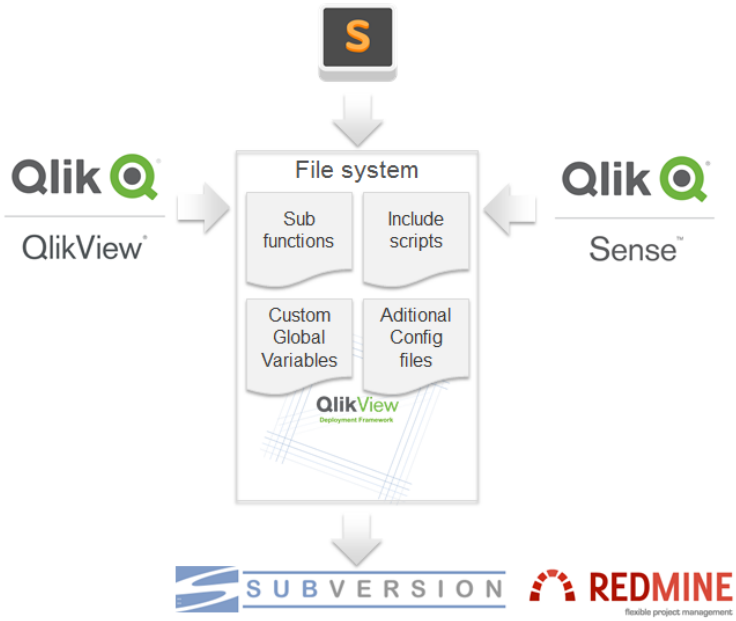
Overview

An integrated development environment (IDE) is a programming environment that has been packaged as an application program. The IDE may be a standalone application or may be included as part of one or more existing and compatible applications. The suggested environment in this document blends several components together into a complete development experience including version control and multi-development support.



The IDE environment could look something like this.

Development and version-control is based on Qlik Include scripts (qvs) that is reusable across multiple applications and can easily be moved between environments. The developer can create sub functions reused over and over again, script snippets for a single purpose or global variables reusable across several applications. Global variables can be used to store common formulas, expressions, color scheme or descriptive text. The enabler for this is Qlik Deployment Frameworks and its reusable code structure. Qlik script development is done outside Qlik's internal load editor instead we are using Sublime text editor. Sublime will use Subversion for version control and InQlik-Tools for syntax high lighting. The IDE integrates with both QlikView and Qlik Sense that will share scripts, functions and global variables.



More details how to create an IDE can be found in *Qlik IDE Development environment.pdf* available at Qlik Community.

Automatic Performance Tests

Scalability Center at Qlik have developed performance test methodizes and tools based on JMeter (an open source test platform). The main communication portal for Scalability Center is QlikCommunity under the Scalability group (http://community.qlik.com/groups/qlikview-scalability). The community group contains all necessary documents and tools to create a performance test matrix.

Tool for easy creation of load/performance tests of QlikView

This package (referred to as QVScriptGenTool) contains a complete set of tools for easy creation, execution and analyzing of load/performance tests.

QlikView version 10 and 11 are supported by this package, Qlik Sense is for now not supported.

Included parts are:

- Standalone application for creating a JMeter script
- Support files for launching the script by utilizing the JMeter engine (JMeter installed is a pre-requisite)
- QlikView applications for analyzing the results from a test session
- Documentation on how to use the package

The QVScriptGenTool package can be downloaded via the QlikView Scalability group at QlikCommunity.

http://community.qlikview.com/docs/DOC-2705

Customer Case

DTAP Process

This customer in the energy sector did a workshop together with Qlik Consultant Services before introducing QDF in the organization. The workshop purpose was to establish their DTAP workflow and align the QDF container structure with their processes.

During the workshop we discovered that the container structure in development should align to the development teams but in acceptance and production the container structure should be aligned on a departmental level. The benefits in alignment to development teams are the ease to remove "leftovers" after project end. Containers handling data sources (system Containers) are the same in test, acceptance and production.

Container Map in Test:

Container Prefix	Container Folder Name
Admin	0.Administration
Shared	002.Shared
Proj1	010. Project1
Proj2	011. Project2
Proj3	012. Project3
Proj4	013. Project4
ARD	001.System\01.ARD
SCCM	001.System\02.SCCM
AD	001.System\03.AD
EAMP	001.System\04.EAMP
OVPI	001.System\05.OVPI
DFM	001.System\06.DFM
Trio	001.System\07.Trio
Survey	003.Common\01. Survey

Container Map in Acceptance and Production:

Container Prefix	Container Folder Name
Admin	0.Administration
Shared	002.Shared
Dep1	010. Department1
Dep2	011. Department2
Dep3	012. Department3
ARD	001.System\01.ARD
SCCM	001.System\02.SCCM
AD	001.System\03.AD
EAMP	001.System\04.EAMP
OVPI	001.System\05.OVPI
DFM	001.System\06.DFM
Trio	001.System\07.Trio
Survey	003.Common\01.Survey

Noticeable is that Shared folder name is changed to *002.Shares*, as long as container path name is *Shares* the physical filename could be altered. The resource containers (001.System and 003.Common) are equal in all environments except the data content. Test only contains test data while production has production data in their resource containers.

Platform consolidation

The second example is also from the energy sector. In this case the customer was consolidating three different environments into one. These environments were developed and maintained by different development teams and had completely different designs. Before project start QlikTech Expert Services did a workshop together with stakeholders introducing DF and discussing how to align containers in the best way for the merge.

The merge have gone very well and faster than expected, here are some feedback from the developers:

- In the beginning it took some extra time to understand and implement the environmental variables in the application scripts.
- When we got the hang of it, the migration process became much easier and faster than without the
  framework.
- The script code is much easier to read and understand when using Deployment Framework
- In the containers we only use some of the folder structures, those not used are deleted after container creation. This is done so that developers won't use the wrong Global Variables. For example, we always delete *2.QVD* and use *7.Export* as qvd storage instead.
- By using the framework we reuse data and scripts much more efficient. Reusability is now by design.
- Some expressions are created as Global Variables that we reuse in every app, like this one:
  TextBetween(Replace('$(vG.BasePath)','$(vG.RootPath)',''),'.','\') &'_'&Left(DocumentName(),len(DocumentName())-4)
  This expression creates concatenated container and file name for every application. We use this variable when creating qvd files.

Production Container Map example after platform migration

Container Prefix	Container Folder Name
Admin	0.Administration
Worklist	0001.Worklist
Agreements	0002.Agreements
Economy	0003.Economy
SpotPrice	0004.SpotPrice
Order	0005.Order
Overview_SAP	0006.Overview_SAP
SAP_BW	1000.DataSources\001.SAP_BW
FlatFile	1000.DataSources\002.FlatFile
ETC	1000.DataSources\003.ETC
Shared	3000.Shared_folders
Restricted	4000. Restricted_folders
Common_scripts	9000. Common_scripts

Notice that in this case the customer has several shared resources. The default common Shared container, a restricted Share container and one container only used as script storage.

Project Methodologies

SCRUM Methodology

Works well with Qlik. Some important factors are

- Since Qlik projects are very rapid, SCRUM's methods of frequent project meetings work well with Qlik development
- A Notification method must be set up between concurrent developers when one of them are changing shared objects
- Define Processes for:
    - QA
    - What denotes an 'Error' when performing QA?
        - Incorrect data/totals are errors
        - Incorrect labels/descriptions are errors
    - What denotes an 'Enhancement'?
        - Changes to the layout (adding, changing items) are enhancements if the item/sheet already passed the initial acceptance by the end user
    - *It's important to denote between Errors and Enhancements because Errors must be fixed, Enhancements must get approved before they are implemented. We try to stay away from enhancements during QA since it may require us to re-QA a lot of work.
- Change Requests
    - What should we do when a user asks to change items? When do we have to ask permissions?
- Communication and Execution plan
    - When will the key stakeholders meet to go over scope changes, or enhancement requests?

RAD/DSDM Methodology

- RAD = Rapid Application Deployment
- DSDM = Dynamic Systems Development Method
  - RAD and DSDM methodologies highly recognized in North America and Europe as part of Agile Project Management Alliance
  - RAD and DSDM methodologies fit the typical Qlik project profile with minimal modification
  - RAD and DSDM methodologies can be referenced and researched to aide project governance and templates
  - RAD and DSDM methodologies provide a foundation for knowledge transfers
  - RAD and DSDM methodologies are resource requirement and documentation lean yet complete

DSDM methodology

- Based on RAD methodology
- One of the Agile methods; part of the Agile Alliance
- Similar to SCRUM in process and concept, HOWEVER:
  - Less jargon than SCRUM
  - No education into SCRUM roles and titles
  - Fewer documents required
- Globally recognized Agile RAD methodology
- Iterative and incremental
- Emphasis on continuous user involvement
- Focus on “on time, on budget” time-boxed, scope consciousness
- Adjustments for changing requirements built in to schedule
- Easily folded into over-arching customer projects and PMO’s
- “Plain language” project effort, roles, and documentation.
- Cyclical back to additional sales & revenue opportunities

RAD/DSDM Elements

Project Phases	3 Phases: • Pre-project • Project Development Life-cycle • Post-project
Project Team Resource Roles	6 Roles • Project Owner/Sponsor • Technical Analyst • Project Manager/Business Analyst • Expert Services Consultant • Qlik Service Partner Developer • Customer Project Team
Documents Required	*8 Documents* • Project Charter with Scope • Requirements: Business, Functional, Non-functional, Technical • Test Plan & Summary • Project Schedule & Plan • Design & Development Summary • Knowledge Transfer & Support Summary • Team Post-project Interview Summary • Customer Satisfaction Interview Summary
Engagement Document	*"Project Charter"* Single Document with tables Documentation: • Charter Purpose, Executive Summary, Project Overview, Scope with goals, objectives, and deliverables, Conditions with assumptions, communication plan, issue tracker, risk tracker, constraints, and escalation path, Structured Approach, Team Organization Plan, Team Contact Directory • Appendix Documents: project schedule (spreadsheet), SOW, change requests, milestone summaries (requirements, design & develop, test, deployment)
Project Schedule	Excel spreadsheet exportable to MS Project, et al project management software, based on MS Project formats

Shown below is a sample project plan for a Qlik project. Qlik recommends that a project plan be created and followed for Qlik projects, and that the help of a qualified Project Manager be sought out for larger projects.

Development

Read more details in Deployment Framework-Development Guide

Skill sets

The development process can be split into two overall groups, *Front End* and *Back End* development. One notice, an individual developer is seldom expert in all the skill sets, try to find and utilize the developer's sweet spot. The Deployment Guide sections are bases on the skill sets below.

Back End developers skill set

- Typically DBA knowledge like
  - Data source expert
  - QlikView data modeling
  - QlikView data model optimization
  - Good understanding in ETL process
  - Data security (Section Access) models

Front End developers skill set

- Typically a BI developer
  - Business specific understanding
  - KPI and measurements
  - QlikView Front End optimization
- Typically a designer skill set
  - Design skills
  - Visualization
  - Usability
- In Qlik Sense Java script and CSS knowhow can be useful

Qlik Development Teams

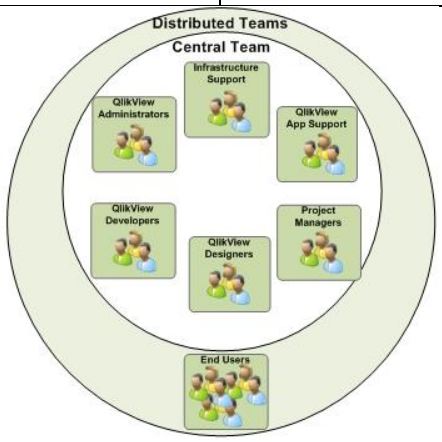
Qlik is an extremely flexible and easily adapted BI tool. As such, development teams can organize around several models for support, administration, back and front end development, training and management. These scenarios help guide discussions about possible configurations for Qlik roles in a development environment.

It is recommended that the client consult its own IT standards for development, as they may drive this decision, or at least narrow the allowed choices. Qlik does not expressly promote one of these scenarios over the others, but asks that clients determine for themselves which of these configurations might work best, given the nature of the Qlik development and the skills sets that exist.

On a continuum from Fully Centralized to Fully Decentralized, the following are 5 options for Qlik team structures:

1) Fully Centralized

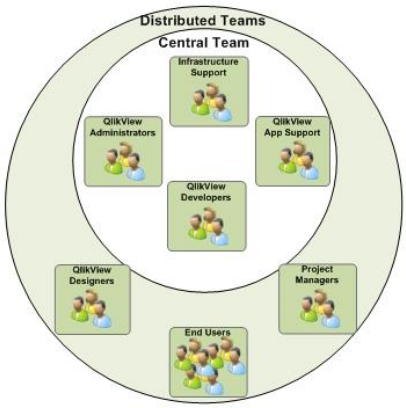
Central Team	**Distributed Team(s)**
Infrastructure Support	End Users
Qlik Administrators	
Qlik Application Support	
Qlik Back end Developers	
Qlik Front End Developers and Designers	
Project Managers	



In this option, departments don't need to supply developers, support personnel or administrators to use applications. They request new applications and then consume them along with central Qlik services. Strengths in this approach are control, skill set sharing, consistency and governance, since all services are contained to one team.

2) Co-Development (v1)

Central Team	**Distributed Team(s)**
Infrastructure Support	End Users
Qlik Administrators	Qlik Front End Developers and Designers
Qlik Application Support	Project Managers
Qlik Back End Developers	

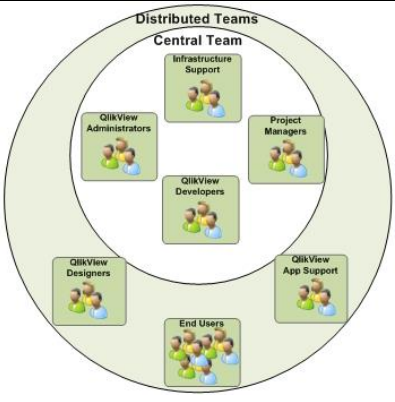


In this option, enterprise development is retained as a central function, allowing for the scripting and data modeling to be handled by expert Qlik developers and data professionals. Departments are responsible for all training, project mgmt, application design, testing and support.

The strengths of this approach are that the back-office BI work is still centralized, but the design and project management are done in the business teams, so that they can move at a faster pace, partially independent from IT resources.

3) Co-Development (v2)

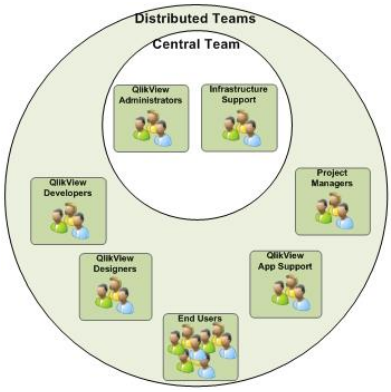
Central Team	**Distributed Team(s)**
Infrastructure Support	End Users
Qlik Administrators	Qlik Application Support
Qlik Back End Developers	Qlik Front End Developers and Designers
	Project Managers



In this option, support has been moved to departments, but Project Mgmt is retained in the central team to better allow for Qlik expertise and control of designs. Departments are responsible for all training, application design, and support. Strengths of this approach are similar to the v1 Co-Development model, with the exception of the application support now also being distributed to the business teams. This frees up more IT resources and places some responsibilities on the distributed teams to provide support and training to their end users.

4) Mostly Decentralized

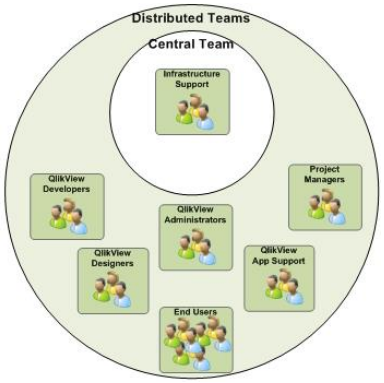
Central Team	**Distributed Team(s)**
Infrastructure Support	End Users
Qlik Administrators	Qlik Application Support
	Qlik Back End Developers
	Qlik Front End Developers and Designers
	Project Managers



In this option, departments are responsible for all training, project mgmt, application design, scripting, development, testing and support. The Central team is still providing infrastructure support and Qlik Administration. This allows for a small IT team (usually less than 2 people) to administer rights, server settings, and batch processing (reloads). Strengths of this approach are that the Centralized (IT) team is very small.

Fully Decentralized

Central Team	**Distributed Team(s)**
Infrastructure Support	End Users
	Qlik Administrators
	Qlik Application Support
	Qlik Front End Developers
	Qlik Front End Developers and Designers
	Project Managers



In this option, infrastructure is still maintained centrally, but all other aspects of Qlik development, testing, support, training a usage are distributed to departments. This requires distributed teams to be trained on all aspects of Qlik. The strength of this option is that there is no software-specific resource needed in a central team. However, the challenge of this approach is that those software-specific resources will need to be present in several distributed teams, possibly overlapping.

Choosing a development team structure is an important step in an enterprise deployment of Qlik. While the Co-Development options (#2 & #3) are the most popular today, the right option for each client is the one that matches their needs and strengths.