# *TSEEQ*: The Structured ETL Engine for Qlik

Version 4.0.9

## Qlik Consulting Services

Document Author:     Jeff R. Robbins

Document Date:     July 20th, 2020

Email:     Jeff.Robbins@qlik.com; jr@qlikperf.com


*Ask, and it shall be given you; seek, and ye shall find; knock, and it shall be opened unto you.*

    (Matthew 7:7, King James Version)


We provide *TSEEQ* (pronounced "seek") in the hope that it will be useful, but without any warranty or guaranteed level of support.

Please note that *TSEEQ* was formerly spelled *SEEQ* (no initial "*T*"); some screen shots, file names and variables may still reflect this older spelling in the near term.

# Contents

# Introduction

*TSEEQ* (pronounced "seek"), The Structured ETL Engine for Qlik, implements centralized management of Extract, Transform and Load (ETL) operations that provide data to QlikView and Qlik Sense applications. Primary benefits of *TSEEQ* are:

- **Governance:** ETL operations are defined in external (and therefore very manageable) rule sets.
- **Self-Service:** Business users may easily define and modify ETL operations in sandbox environments.
- **Productivity:** The core ETL engine and its surrounding utilities automate time-consuming tasks.
- **Performance:** A profiler enables efficient ETL execution.
- **Migration:** ETL rule sets for QlikView can be used without modification for Qlik Sense (and vice versa).
- **Integration:** Data from multiple sources is conformed and integrated to create a consolidated data model.
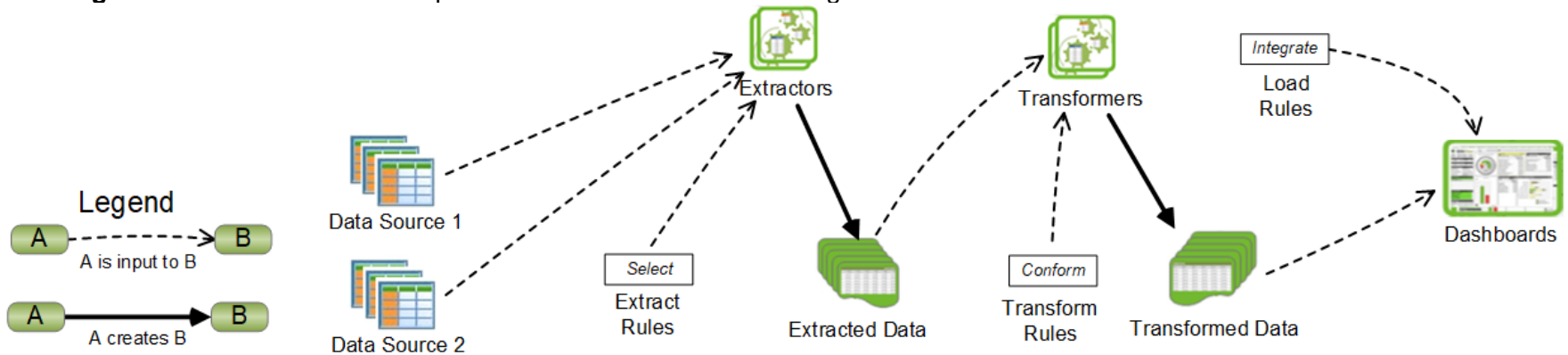
## *TSEEQ* In Comparison to Traditional Embedded Scripting

The distinguishing characteristic of *TSEEQ* versus Traditional Embedding Scripting (**TES**) is that in *TSEEQ*, externalized rules (diagram above) provide a structured source of ETL control; in **TES**, free-form textual ETL script is embedded within Qlik application files (**QVWs** in QlikView and **QVFs** in Qlik Sense). We can conceptualize **TES** as shown in the diagram below:
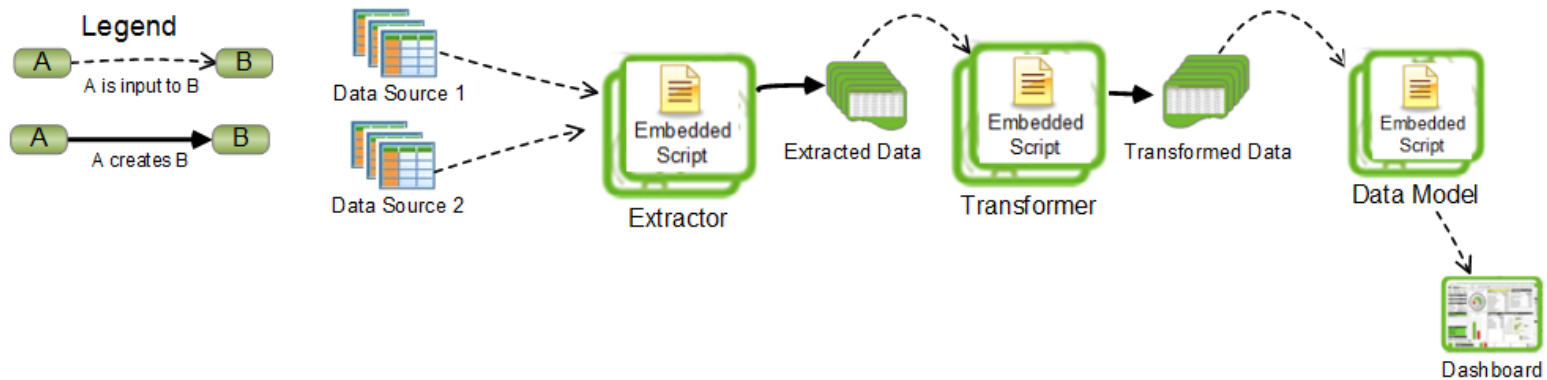
Note: *TSEEQ* and **TES** are not mutually exclusive; a hybrid approach is useful in many cases. Another approach, TeXas, is also useful.

# Technical Architecture

**TSEEQ** reads ETL rules from a metadata store and then automatically generates and executes Qlik script to build QVDs and QlikView or Qlik Sense data models.

1. **TSEEQ** implements the concept of **common** components, including both data and code, that are shared among multiple applications. Common components from a typical **TSEEQ** deployment are shown in the left and center of the diagram at right.

2. **TSEEQ** does not provide a graphical drawing tool for data flows; rather, the Qlik Developer defines ETL rules in a tabular metadata store.

   a. The metadata store may be persisted in a relational database, or as files (XLS, CSV, QVD, etc).

   b. Despite the lack of a drawing tool, **TSEEQ** ETL rules are easily created by modifying sample rules provided in the **TSEEQ** *Sales Sample* (detailed on the next page).

3. Field transformation rules are Qlik expressions; **TSEEQ** is an abstraction layer on top of the Qlik script.

4. Since **TSEEQ** is a set of script routines executing in the context of a QVW or QVF, standard QlikView Publisher or Qlik Sense QMC tasks are the scheduling mechanism.

## Sample *TSEEQ* Data Flow
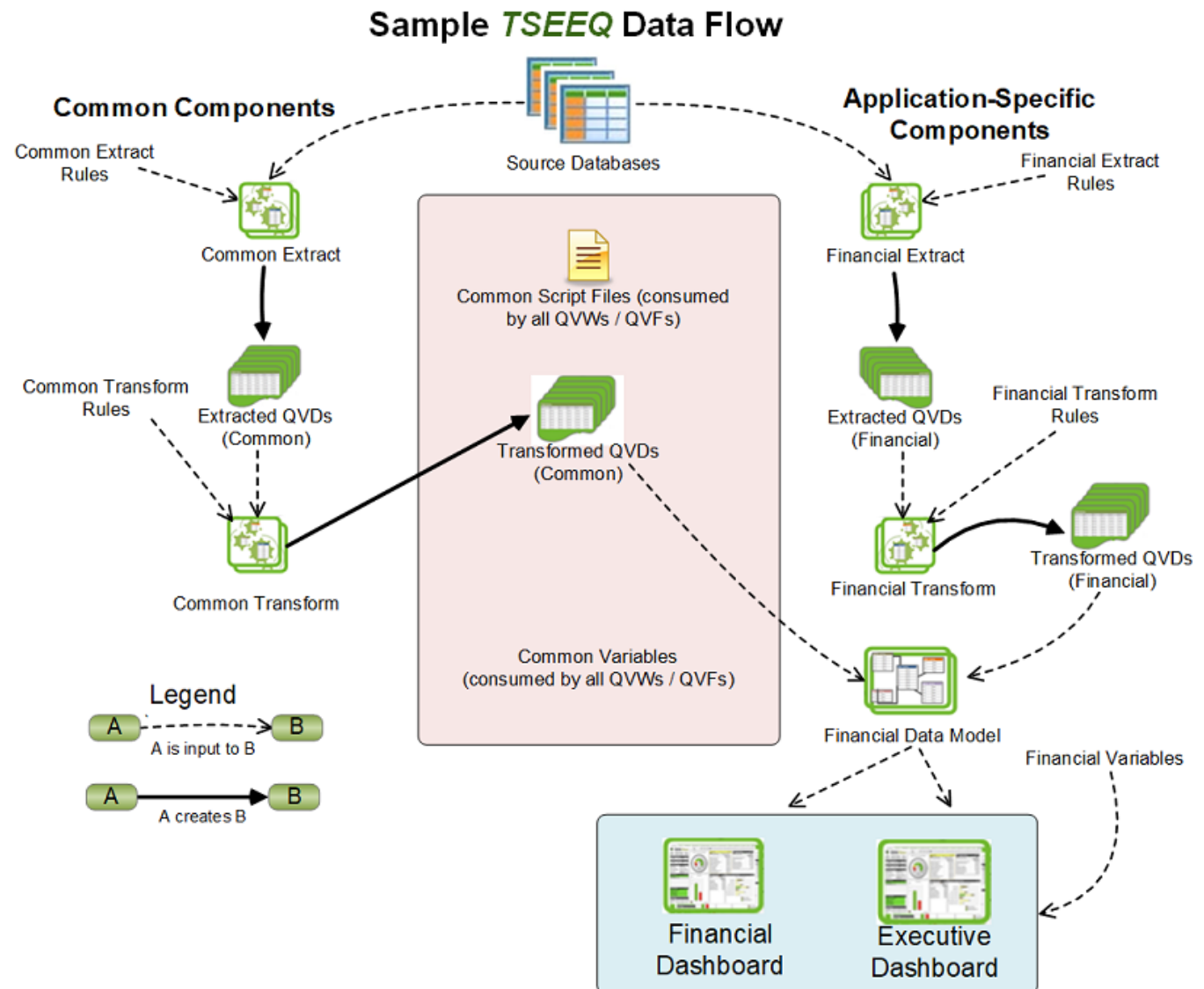
## Instructions for the *TSEEQ Sales Sample*:

The *TSEEQ Sales Sample* provides an end-to-end working example of a **TSEEQ** ETL flow that can be executed and modified for learning purposes, as well as serve as a template from which to implement **TSEEQ** ETL flows for additional applications.

### ZIP File Download and Unblock Instructions (for Both Qlik Sense and QlikView)

Download the **Source code (zip) file** for the latest **TSEEQ** release from https://github.com/qlikperf/TSEEQ/releases.



After you download the file, please check **Unblock** in the ZIP file properties and then click **OK**.

Note that the screenshot at right shows a ZIP file named **TSEEQ-4.0.4.zip**; if you have downloaded a newer release, your **TSEEQ ZIP** file will have a higher release number.

The remainder of this document simply refers to the downloaded file as "the **TSEEQ ZIP** file".
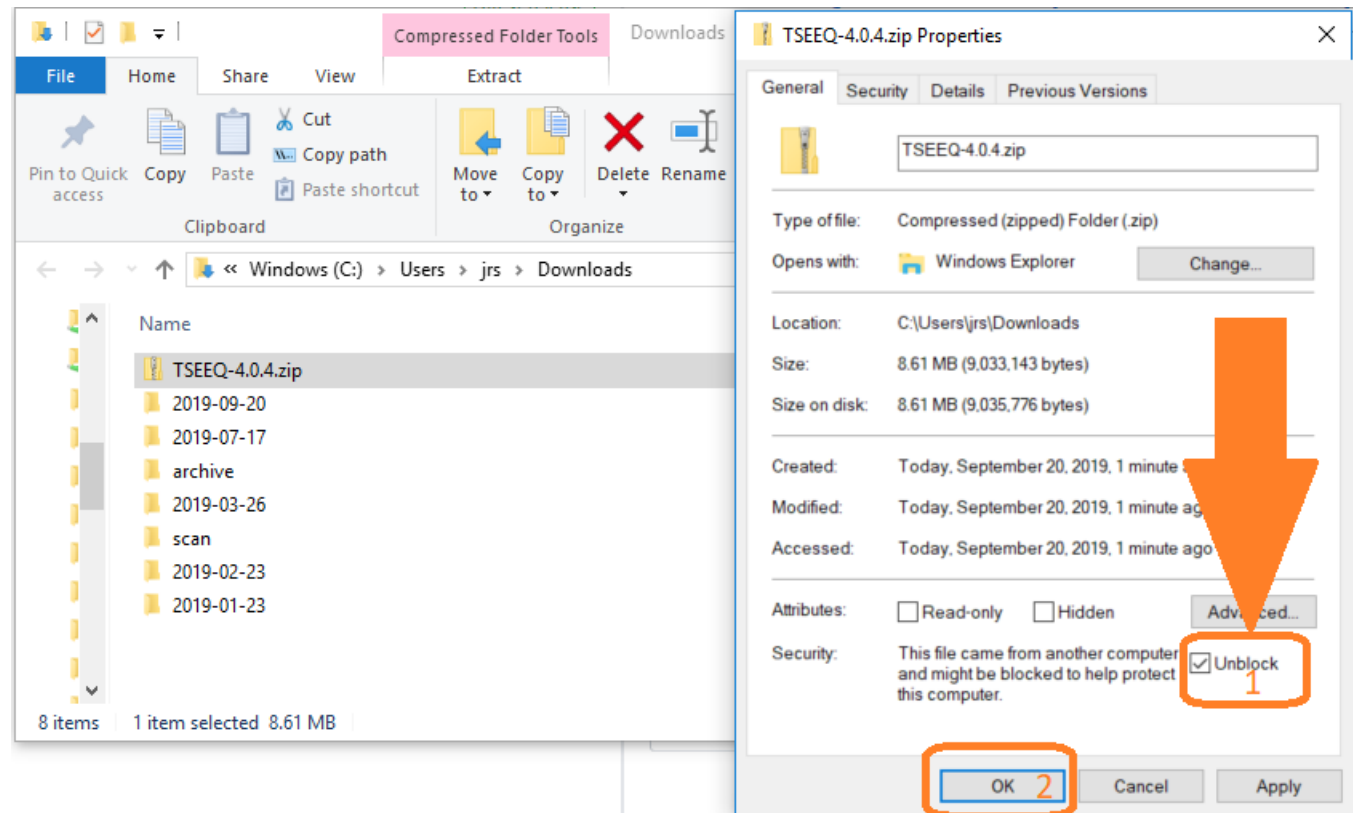
# ZIP File Extraction Instructions (Both Qlik Sense and QlikView)

1. For **Qlik Sense Desktop** users, we recommend extracting the contents of the **TSEEQ ZIP** file to the path of *C:\TSEEQ*, since this will allow use of pre-built data connections within the provided sample QVFs.

    a. Any alternate folder location may be used; however, an alternate location will require modification of the pre-built folder data connections to point to the alternate location.

2. For **QlikView** and **Qlik Sense Enterprise,** we recommend extracting the contents of the **TSEEQ ZIP** file to a sub-folder of a network share to which your development team has read and write access.

    a. The screen shot at right shows an example, a *\\w2008001\share* with a *TSEEQ* sub-folder contained within.

3. For the rest of this document, ~\ refers to the local folder or sub-folder within a network share to which you extracted the **TSEEQ ZIP** file contents
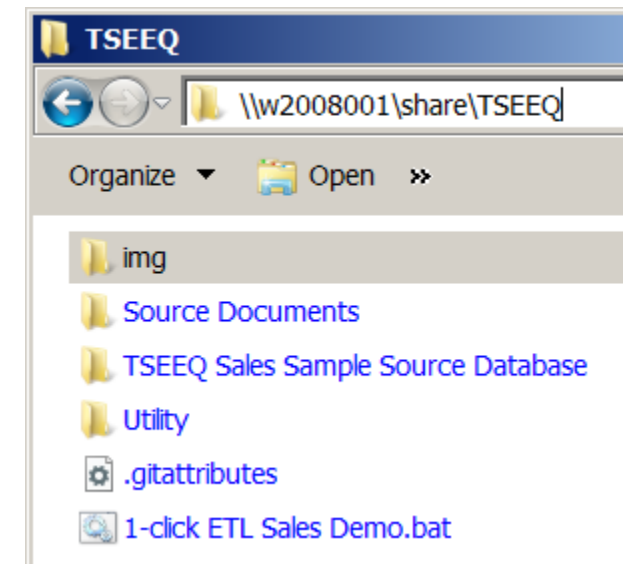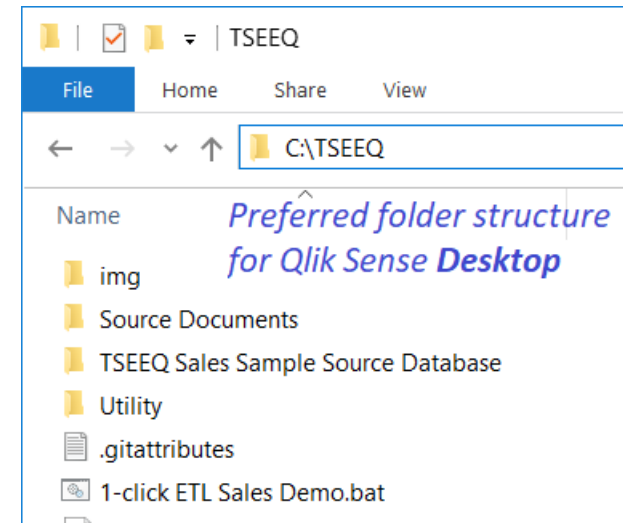
## ETL Execution Instructions (QlikView)

1. For **QlikView** <mark>November 2018 and earlier</mark>, to run the whole ETL flow for the Sales demo application, simply double-click the **1-click ETL - Sales Demo.bat** file that is in the top-level ~\\*TSEEQ* folder.

    a. The Extract, Transform and Load QVWs for the *TSEEQ Sales Sample* will then all be reloaded in sequence by the **bat** file.

        i. If prompted with a *Save As* dialog, click **Save** to overwrite the old file and click **Yes** when prompted if you want to replace the file.

        ii. For additional insight into the QlikView ETL flow, please view the **bat** file in Notepad or any other text editor.

If Windows will not allow the **bat** file to execute, please ensure that the **TSEEQ ZIP** file is unblocked as described on the prior page, and then extract the **TSEEQ ZIP** file contents again.

2. For **QlikView** <mark>April 2019 and later</mark>, execution of the **bat** file may not result in the desired reload behavior, since the **/r** option to **qv.exe** has been problematic in some newer releases.  In which case, open and reload each of the following QVWs in sequence:

    a. ~\\*TSEEQ\Source Documents\TSEEQ Sales Sample\1_Extract\Sales Extract.qvw*

    b. ~\\*TSEEQ\Source Documents\TSEEQ Sales Sample\2_Transform\Sales Transform.qvw*

    c. ~\\*TSEEQ\Source Documents\TSEEQ Sales Sample\3_Load\Sales DataModel.qvw*

    d. ~\\*TSEEQ\Source Documents\TSEEQ Sales Sample\4_App\Sales Dashboard.qvw*

After successful reload of the 4 QVWs, please see the section in this document about Validating Successful ETL Execution.

## ETL Execution Instructions (Qlik Sense)

Below, you only need to complete Step 1 OR Step 2, depending on your Qlik Sense product:

1. For **Qlik Sense Desktop**, copy each of the 3 ETL QVFs to your Qlik Sense Apps folder. The default location of this folder is *C:\Users\<user id>\Documents\Qlik\Sense\Apps*. The 3 ETL QVFs to copy are:

   a. *~\TSEEQ\Source Documents\TSEEQ Sales Sample\1_Extract\Sales Extract.qvf*

   b. *~\TSEEQ\Source Documents\TSEEQ Sales Sample\2_Transform\Sales Transform.qvf*

   c. *~\TSEEQ\Source Documents\TSEEQ Sales Sample\4_App\Sales Dashboard.qvf*

Note that that step **c** references a folder pre-fixed with the number 4, rather than the number 3; this omission of "3" is intentional. Previous versions of *TSEEQ* used a 4-stage ETL architecture; however, *TSEEQ* now defaults to 3-stage ETL, in which the data model LOAD (step 3) and the front-end UI APP (step 4) are combined into a single QVF stored under the *4_App* folder. 4-stage ETL is still available in the QVFs that contain "deprecated" within their names; however 3-stage ETL is preferred, for reasons detailed later in this document.

2. For **QlikSense Enterprise**, use the QMC to import each of the following 3 QVFs:

   a. *~\TSEEQ\Source Documents\TSEEQ Sales Sample\1_Extract\Sales Extract.qvf*

   b. *~\TSEEQ\Source Documents\TSEEQ Sales Sample\2_Transform\Sales Transform.qvf*

   c. *~\TSEEQ\Source Documents\TSEEQ Sales Sample\4_App\Sales Dashboard.qvf*

3. Next, create the folder data connections[1]. If you are using **Qlik Sense Enterprise**, you will need to create the connections in the **Data Load Editor.** Qlik Sense Enterprise users should also use the **QMC** to strip out the *(<directory>_<userid>)* postfix from each connection name[2]. Connections should be defined as follows:

   a. **Admin**: folder connection pointing to *~\TSEEQ\Source Documents\Admin*.

   b. **Sales**: folder connection pointing to *~\TSEEQ\Source Documents\TSEEQ Sales Sample*.

   c. **Common:** folder connection pointing to *~\TSEEQ\Source Documents\Common*.

The next page describes how to create the **QWT** OLE DB connection.

---
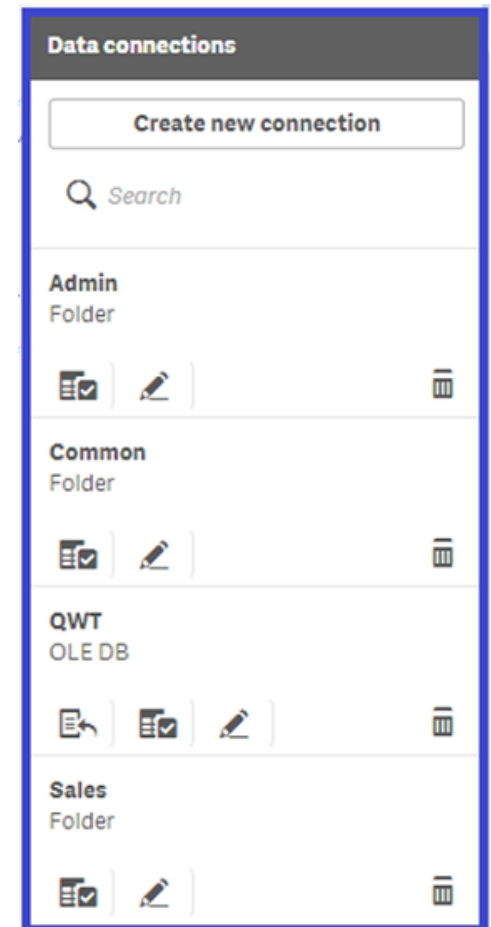
[1] **Qlik Sense Desktop** users should be able to use the pre-built connections in each QVF, unless a folder other than *C:\TSEEQ* is being used.

[2] If the connection names cannot be edited due to QMC access restrictions, each of the ETL QVFs provides the option to use post-fixed connection names; please see the **01-Main** script tab within each QVF for more information.

4. Next, create a new OLE DB connection, named **QWT**, configured as shown in the screen shot at right.
   a. **Provider** set to Microsoft Jet 4.0 OLE DB Provider (32-bit).
   b. **Data source** set to *~\TSEEQ\TSEEQ Sales Sample Source Database\QWT.mdb*.
   c. Choose **Specific user name and password**, and leave the **Username** and **Password** blank.
   d. Specify a **Name** of QWT.
   e. Click **Create**.

**Create new connection (OLE DB)**

Provider

Microsoft Jet 4.0 OLE DB Provider(32-bit) ▼

Data source (file path or server name)

:\TSEEQ\TSEEQ Sales Sample Source Database\QWT.mbd

◯ Windows integrated security

⦿ Specific user name and password
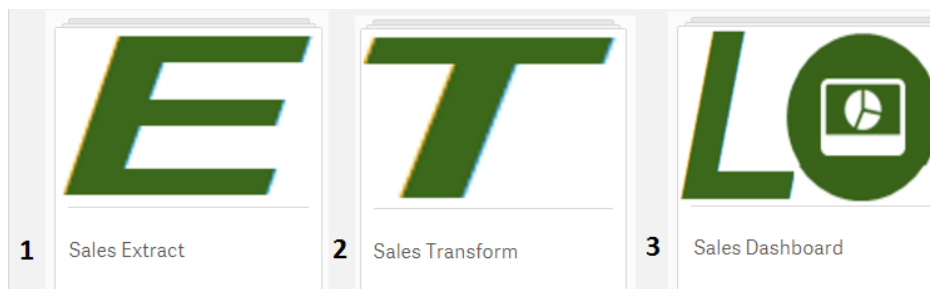
Username

**leave blank**

Password

**leave blank**

Test connection

5. Open the **Data Load Editor** and press **load data,** for each of the following QVFs in sequence:
   1. *Sales Extract*
   2. *Sales Transform*
   3. *Sales Dashboard*

Database

Load      Select database...      ▼

Name

QWT

Cancel      Create



| 1 | Sales Extract | 2 | Sales Transform | 3 | Sales Dashboard |

# Validating Successful ETL Execution

Successful **TSEEQ** execution can be validated within both QlikView and Qlik Sense by viewing the **Performance Profile** and **Execution Trace** tables within the ETL QVWs and QVFs.

The screen shot at right shows the **Execution Trace** from the sample **Sales Transform** QVF for Qlik Sense (a similar table is provided in the **Sales Transform** QVW for QlikView, as well as the **Sales Extract** and **Sales DataModel** QVWs and QVFs).
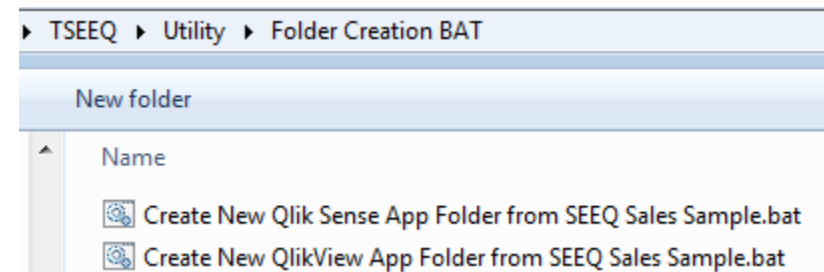
**Execution Trace**

| Execution Order | Executed Script | Execution Start Time | Execution Completion Time | Execution Time Elapsed |
|---|---|---|---|---|
| 1 | [Accounts]:<br>LOAD<br>    BillingCity AS [Billing City],... | 07-22-17 01:51:11 PM | 01:51:13 PM | 00:00:02 |
| 2 | [Contacts]:<br>LOAD<br>    AccountId AS [Account Id],... | 07-22-17 01:51:13 PM | 01:51:17 PM | 00:00:04 |
| 3 | [CaseOwners]:<br>LOAD<br>    FirstName & ' ' & LastName AS [Case O... | 07-22-17 01:51:19 PM | 01:51:20 PM | 00:00:01 |
| 4 | [Timecards]:<br>LOAD DISTINCT<br>    Id AS [Timecard Id],... | 07-22-17 01:51:20 PM | 01:51:20 PM | 00:00:00 |
| 5 | [Projects]:<br>LOAD<br>    (... | 07-22-17 01:51:20 PM | 01:51:20 PM | 00:00:00 |

# Creating New App Folders from the *TSEEQ Sales Sample*:

After you have successfully run and validated the end-to-end ETL flow provided within the *TSEEQ Sales Sample* as discussed on the prior pages, you may wish to replicate the folder structure of the *TSEEQ Sales Sample* for use with additional applications[3]. To facilitate this folder structure replication, you may run the batch files under ~\TSEEQ\Utility\Folder Creation BAT.

Please note that:

1. **TSEEQ** does not require any specific folder structure be used; **TSEEQ** can be adapted to an existing folder structure if one is already in place.

2. With Qlik Sense Enterprise, the folder structure is less relevant than with QlikView; Qlik Sense Enterprise stores all "apps" (QVFs) in a repository and therefore the concept of a folders does not apply for QVFs (with the one exception being a BINARY LOAD of a QVF via a folder connection). In Qlik Sense, the concept of folders only applies to data files (such as XLSX and QVD) and externalized script that is brought in with an **INCLUDE** statement.

> TSEEQ ▸ Utility ▸ Folder Creation BAT
>
> New folder
>
> Name
>
> 📄 Create New Qlik Sense App Folder from SEEQ Sales Sample.bat
> 📄 Create New QlikView App Folder from SEEQ Sales Sample.bat

---

[3] A later section of this document defines terms such as "app" and "application" a bit more explicitly.

# Enabling Additional Rules & The GENERATE ONLY Mode

Please note the included sample rules files under the following folder:  ~\*Source Documents\TSEEQ Sales Sample\ETL_Rules*.

By default, *TSEEQ* generates and executes ETL script for those rules where **ENABLED** is set to **Y**.  The currently enabled rules are those which operate upon the included sample database (~\*TSEEQ Sales Sample Source Database\QWT.mdb*).



*TSEEQ* will not generate and execute ETL script rules where **ENABLED** is set to *N*, since a corresponding sample database is not included in the *TSEEQ* zip file. However, you may optionally enable those rules by

1. changing **ENABLED** to **Y**, and then
2. uncommenting the following line in on the *01-Main* script tab: `// SET vTSEEQ_Mode = 'GENERATE ONLY';`

*TSEEQ* will then generate, but **not e**xecute, ETL script*,* thereby allowing you to see how *TSEEQ* creates ETL script for a wider variety of ETL rules.

# Documentation Within the ETL QVFs and QVWs

As shown in the screen shot, the script within each of the *TSEEQ Sales Sample* QVFs (Qlik Sense) and QVWs (Qlik View) is extensively commented (comment lines outnumber actual code lines by over 3-to-1). Information from those intra-QV* comments is not replicated in its entirety within this document.  As such, developers are referred to the intra-QV* comments for additional information on the script within the QV* files.
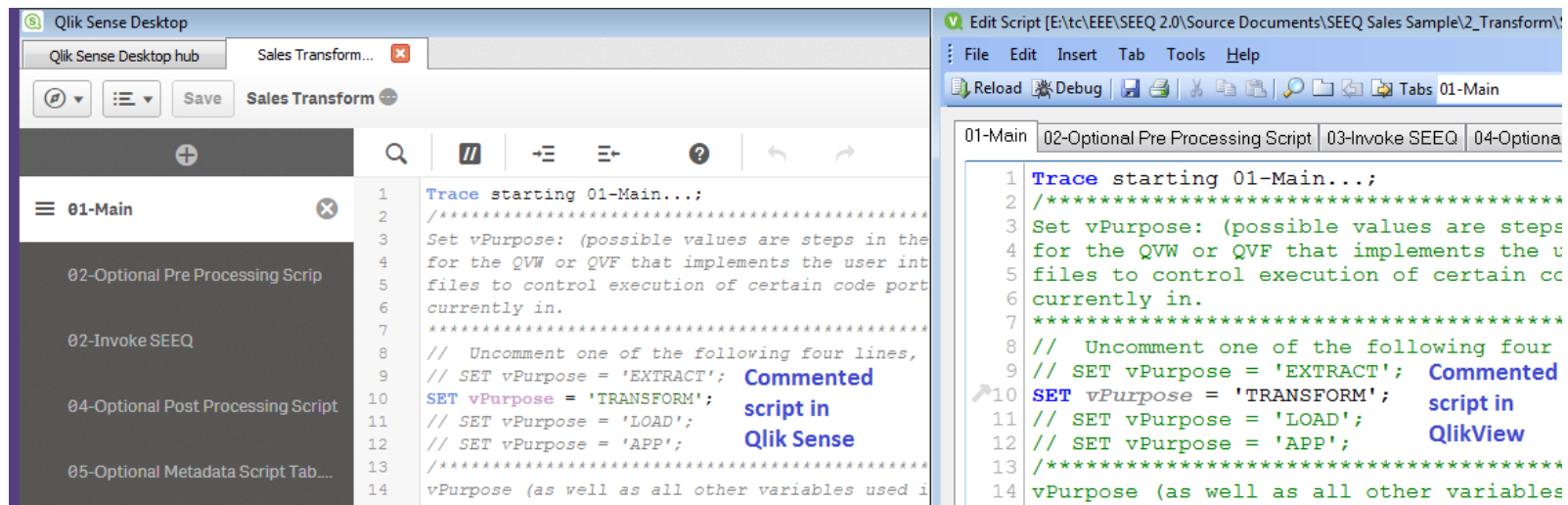


# Documentation Within the XLS Files

As shown in the screen shot, each column and row within the XLS files in the *TSEEQ Sales Sample* contains embedded documentation. This documentation is not replicated in its entirety within this document.  As such, Qlik developers are referred to the intra-XLS documentation within the *TSEEQ Sales Sample*:



**Column headers includes comments that are displayed on mouse hover.**

**Each row contains a comment.**

# ETL Statistics (Most Recent Execution)

In the *TSEEQ* *Sales Sample*, ETL Statistics for the most recent execution shown in within the respective **Extract, Transform and Load** QVFs (Qlik Sense) and QVWs (QlikView).

The screen shot immediately below shows a view of the Qlik Sense statistics on the left and the QlikView statistics on the right.

# ETL Statistics (Historical)

ETL statistics for the past 100,000 ETL operations are analyzed in the **~\Admin\4_App\ETL Analysis QVW** and **QVF.**

The screen shot shown below is from the **ETL Analysis** QVW in QlikView; a similar sheet is provided in Qlik Sense by the **ETL Analysis** QVF.

Note that for Qlik Sense Enterprise, **~\Admin\4_App\ETL Analysis.qvf** will need to be imported via the QMC prior to first use.

# Considerations on 3-Stage vs 4-Stage Data Architectures

*TSEEQ* allows for both 4-Stage (separate QVFs for data model and dashboard) and 3-stage (single QVF for data model and dashboard) architectures. At first glance, the 4-stage approach seems to provide more modularization; however, in Qlik Sense, the **binary** load required in the 4-stage approach is not always feasible[4].

As such, the 3-stage approach may be preferred in some cases. Note that in the 4-stage architecture, any single data model is a re-usable asset that can consumed by multiple dashboards. In the 3-stage approach, the consumable data model concept is not used; however, a single set of **Load Rules** (which define a data model) may be consumed by multiple dashboards. All dashboards consuming any specific **Load Rule** set will contain identical table structures and data sets; as such, a 3-stage ETL based on an on an externalized **Load Rule** set provides the same level of modularization that a 4-stage ETL does.
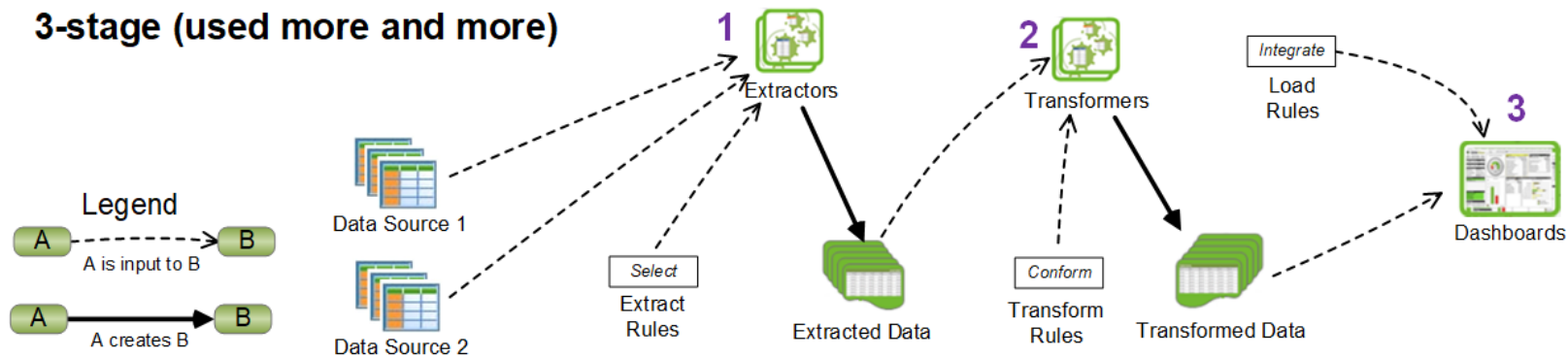


---
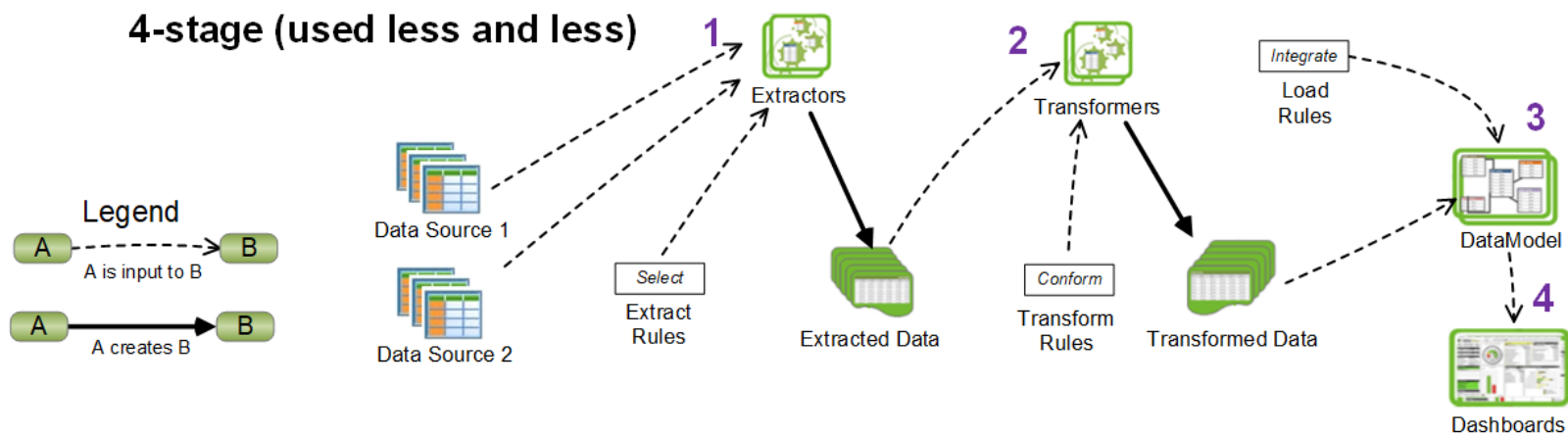
[4] As per the documentation since September 2017, "You cannot use **binary** to load data from an app on the same Qlik Sense Enterprise deployment by referring to the app ID." Technically, you can still can **binary** load by app id, but this technique is definitely in the realm of "not supported."

## ApplyMap Example

*TSEEQ* includes an example Transform rule set implementing the Qlik ApplyMap() functionality:

    *~\Source Documents\TSEEQ Sales Sample\ETL_Rules\ApplyMap Transform Rule Example.xls*

1. The mapping table can be loaded from either a QVD or from an EXTERNAL file (XLS*, CSV, etc). The example uses an XLS file (*EmpOff.xls*).

2. **MAPPING LOAD**, not simply **LOAD**, is the **Load Command** for the mapping table.

3. **INCLUDE_SUBSET** is required in the **Fields** column, if the source has more than 2 fields. If the source only has two fields, you could just use **ALL** in the **Fields** column, if those two fields are ordered with the key first and the mapped value second.

4. The **Keep or Drop** option does not apply, since Qlik will automatically drop the mapping table.  You can specify **KEEP**, but Qlik will still drop the table.

5. A **TRANSFORM_FIELDS** rule applies the map.

## Notes on the *vApp* Variable

Qlik Sense developers should note that the **TSEEQ** *vApp* variable is not equivalent to the Qlik Sense concept of "app".

- In **Qlik Sense,** an "app" corresponds to a single QVF on disk. In Qlik Sense Desktop, this QVF is visible in the developer's local file system; in Qlik Sense Server, the QVF is within a centralized repository and therefore somewhat hidden.

- We should precisely define what "application" refers to in the following paragraphs: "application" is deployed software that the end user interacts directly with, for example, the **Financial Dashboard.** Technically speaking, an "application" is implemented by a set of files (including one or more QVFs or QVWs), but the end user does not have file-level visibility.

- In **TSEEQ**, *vApp* is not a single file, but rather the prefix to multiple file names. *vApp* does not refer to an "app" in the sense of a QVF; rather, *vApp* refers to the set of files that implement a group of applications.



**Sample *TSEEQ* Data Flow**
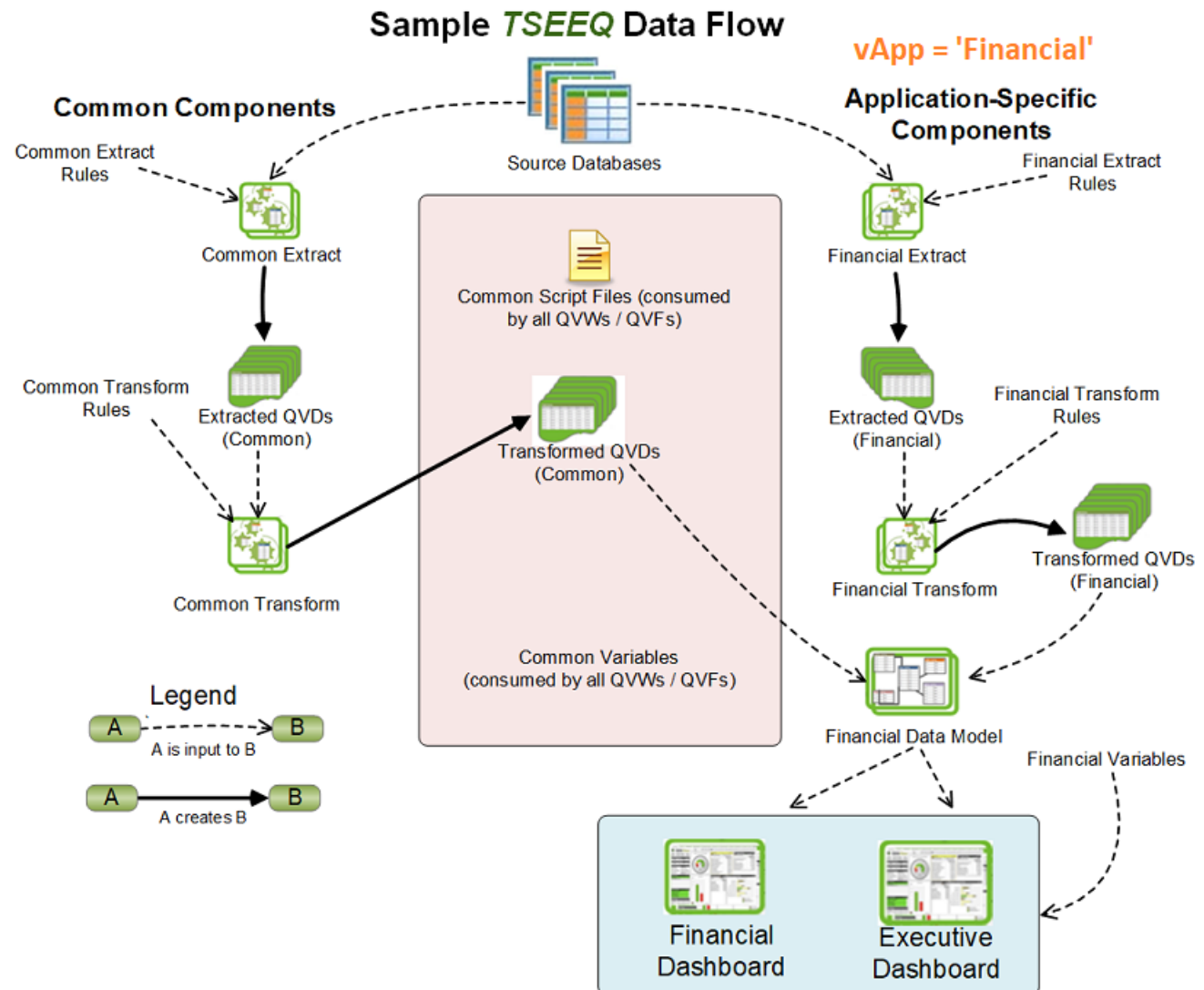
In the **Sample *TSEEQ* Data Architecture** (diagram at right), *vApp* = 'Financial'; the corresponding group of applications includes the **Financial Dashboard** and the **Executive Dashboard.** All back-end file names in the build chain (which end users do not see) are prefixed with '**Financial',** the value of *vApp*.

In this example, the **Financial** and **Executive** dashboards were initially conceived of as two separate initiatives; after some analysis, we realized that the **Executive** dashboard had high requirements overlap with the **Financial** and therefore based the **Executive** on the **Financial** build chain.

Note that the **Extract**, **Transform,** and **DataModel** QVFs in the **Example** *TSEEQ* **Data Architecture** (diagram on previous page) are all persisted as QVFs and are therefore considered "apps" from the Qlik Sense perspective. However, these back-end QVFs are not exposed to end users and are therefore not what we would refer to as "applications".

So, to put this to code:  suppose that there is a set of variables that every Financial-related QVF must read in, regardless of the QVF's *vPurpose* (**EXTRACT**, **TRANSFORM**, data model **LOAD**, or front-end dashboard **APP**).   Then, we could create a file, **FinancialVariables.xls,** and each of the financial QVFs could read in the variables with the following code:

```
/*******************************************************************
    Load financial-specific variable definitions:
********************************************************************/

call Load_Variables_from_XLS('$(vVariablePath)FinancialVariables.xls', 'Sheet1');
```

We could further genericize this code:

```
/*******************************************************************
    Load app-specific variable definitions:
********************************************************************/

call Load_Variables_from_XLS('$(vVariablePath)$(vApp)Variables.xls', 'Sheet1');
```

And the same single line of code immediately above could conceivably be used in every QVF that needs to read in *vApp*-specific variables: All Sales QVFs would read the Sales variables; all Financial QVFs would read the Financial variables.

We wish your success with *TSEEQ*; please contact us with any questions!  And if you're still reading, there's some additional info in the Appendixes on the following pages!

# Appendix 1: More About TES, TeXas and *TSEEQ*

## Starting out with TES (Traditional Embedded Script)

Many Qlik apps are built with a 3-stage data architecture as shown in this diagram:



**TES (Traditional Embedded Script), 3-Tier**
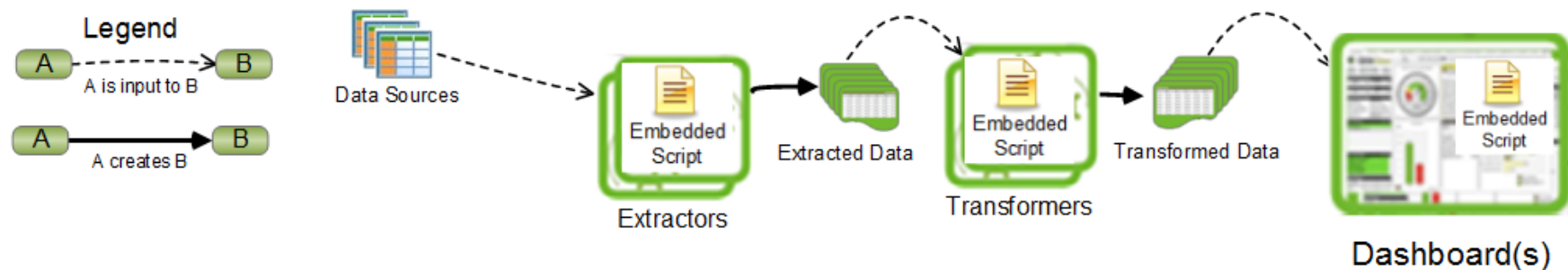
In this 3-Stage **TES** architecture, Extractor "apps" query data sources to create the first QVD layer ("Extracted Data"). Transform apps then aggregate, cleanse and/or de-normalize the Extracted Data to create the second QVD layer ("Transformed Data"), which is read directly by dashboards.

This 3-Stage architecture promotes data re-use and efficiency; a table need only be pulled from the database once to populate the Extracted Data layer. And data typically needs to be only read once from the Extracted Data layer to be cleansed and then added to the Transformed Data layer, from where it can feed multiple Dashboards. *Voila*, we have data re-use!

However, embedding script in the Extractors and Transformers results in cross-environment promotion complexity:

- In **Qlik Sense** Server, the Extractors, Transformers and Dashboards are all "apps" that reside in a Qlik Sense repository. If for a given Dashboard, the script in the associated Extractors and Transformers changes, then in a cross-environment migration (ex **Dev** to **Test**) scenario, all Extractors and Transformers associated to a migrating Dashboard must be manually exported from the source environment (**Dev**) and then manually imported into the target environment (**Test**).

- With **QlikView** Server, the "apps" are QVW files that are readily visible on the file system. However, QVWs are stored in a binary format; version control of the embedded ETL scripts is therefore a challenge[5].

Luckily, we can greatly mitigate these concerns by supplementing the traditional embedded script with more a manageable, externalized approach, as discussed on the next page.

---

[5] The Project Folder technique allows decomposition of a QVW into XML and text files that are better suited to version control than are binary QVWs. However, discussion of that technique is outside the scope of this document; on the next page, we discuss "TeXas", a very light weight and efficient way to enable version control of the ETL script exclusively.

# Streamlining the SDLC with TeXas (Traditional Externalized Script)



**Texas (Traditional Externalized Script)  3-tier**
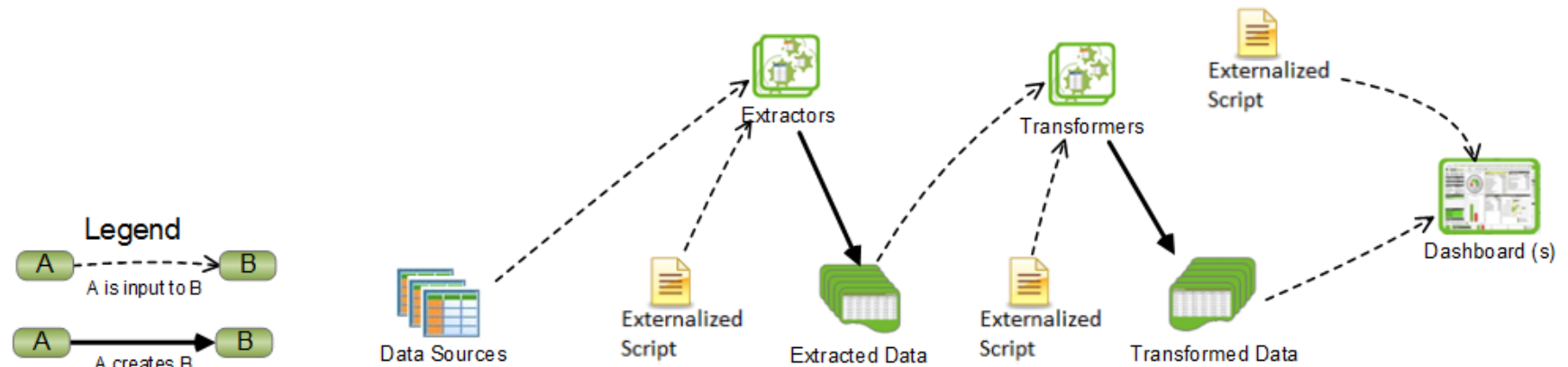
With **TeXas**, the ETL scripts that are executed are the same as with **TES**, it's just that the ETL scripts are persisted in text files that are *external* to Qlik apps; in the case of Qlik Sense, the scripts are therefore external to the Qlik Sense repository.

Recall that with **TES** (Traditional Embedded Script) as discussed on the prior page, the ETL script is embedded within "apps", each of which is a binary filed stored within the Qlik Sense repository.  With **TES**, cross-environment Dashboard promotion requires, for each associated Extractor and Transformer app, an export from the source environment followed by an import into the target environment.

With **TeXas**, cross environment promotion of a given Dashboard rarely requires cross-environment promotion of the associated Extractors and Transformers.  Rather, the externalized scripts referenced by the Extractors and Transformers are the assets to be promoted.

Since the externalized scripts are simply text files in a Windows file system, they do not require the export-import step that is required for Qlik Sense app promotion. Rather, the externalized script files can be easily managed in a source control system (SubVersion, TFS, etc) and promoted from one environment to another with a simple file copy.

# Stepping Beyond TeXas with *TSEEQ,* The Structured ETL Engine for Qlik

*TSEEQ* (pronounced "seek"), The Structured ETL Engine for Qlik, implements centralized management of Extract, Transform and Load (ETL) operations that provide data to Qlik Dashboards.

As with **TeXas**, the assets that define *TSEEQ* ETL operations are external to Qlik Sense apps; however, *TSEEQ* uses tabular ETL rule definitions, rather than the free-form scripting used in **TeXas**.

*TSEEQ* **(Structured ETL Engine for Qlik) 3-stage**

Note that **TES, TeXas,** and *TSEEQ* are not mutually exclusive; a hybrid approach is feasible and useful in many cases. Any single "app" (QVW or QVF) can use all three approaches; a single app might contain embedded script, reference externalized script, and execute a *TSEEQ* rules file[6].



---
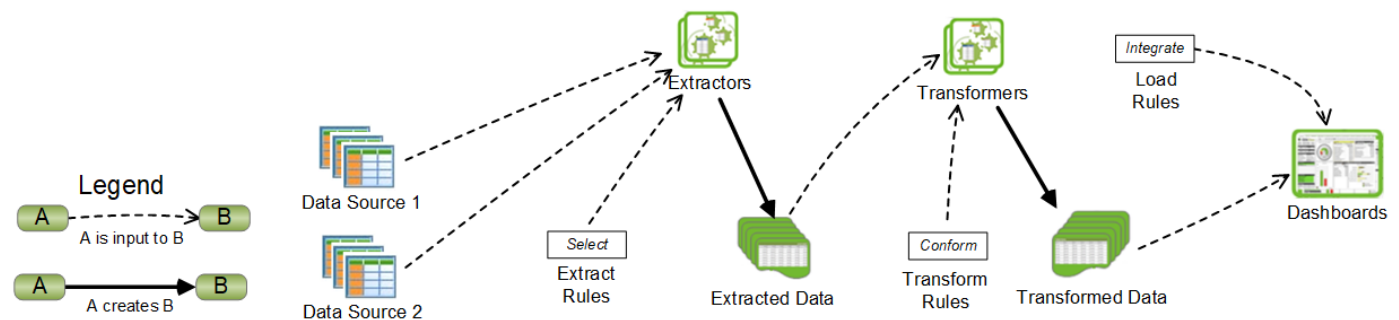
[6] Astute readers might point out that both **Texas** and *TSEEQ* use embedded script, specifically just enough embedded script to reference the required external files. However, this small amount of embedded script rarely, if ever, changes. **TES** by contrast, typically entails a significant amount of frequently changing embedded script.

# Appendix 2: Utilities Included with *TSEEQ*

Several utilities are included under **~\TSEEQ\Utilities.** Documentation for these utilities is largely embedded with the utilities themselves, in the form of intra-script comments.

Where available, more formalized documentation for each of these utilities is linked immediately below.

## The **QVD LooQer**

The **QVD LooQer** enables fast and convenient viewing of QVD contents, with no additional associated software licensing costs. Documentation for this utility can be found On GitHub at this link: https://github.com/qlikperf/TSEEQ/blob/master/Utility/QVD%20LooQer/Readme%20QVD%20LooQer.pdf