TSEEQ: The Structured ETL Engine for Qlik

Version 3.1.2

Qlik Consulting Services

Document Author: Jeff R. Robbins
Document Date: March 15, 2018
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.

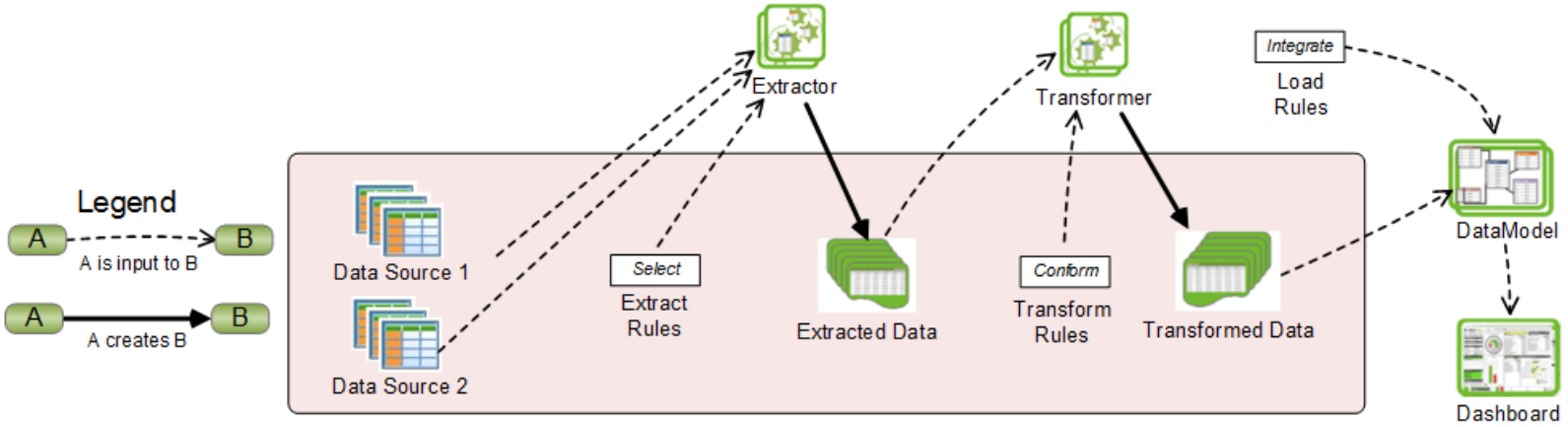
Contents

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.

Introduction

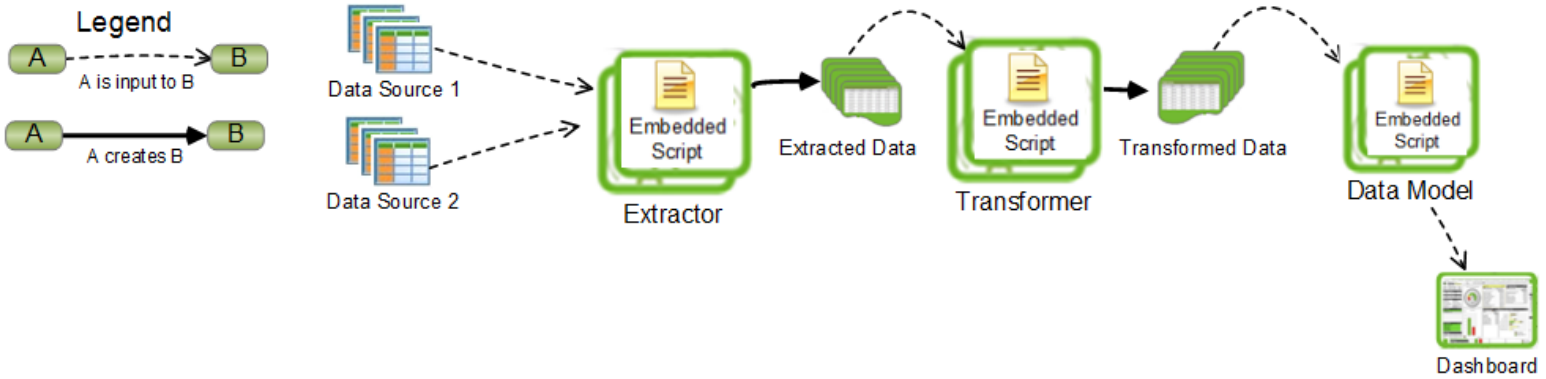
TSEEQ (pronounced “seek”), The Structured ETL Engine for Qlik, implements centralized management of Extract, Transform and Load (ETL) operations that provide data to QlikView and Qlik Sense applications. Primary benefits of ***TSEEQ*** are:

- **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.
- **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.
- **Productivity:** The core ETL engine and its surrounding utilities automate time-consuming tasks.



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.

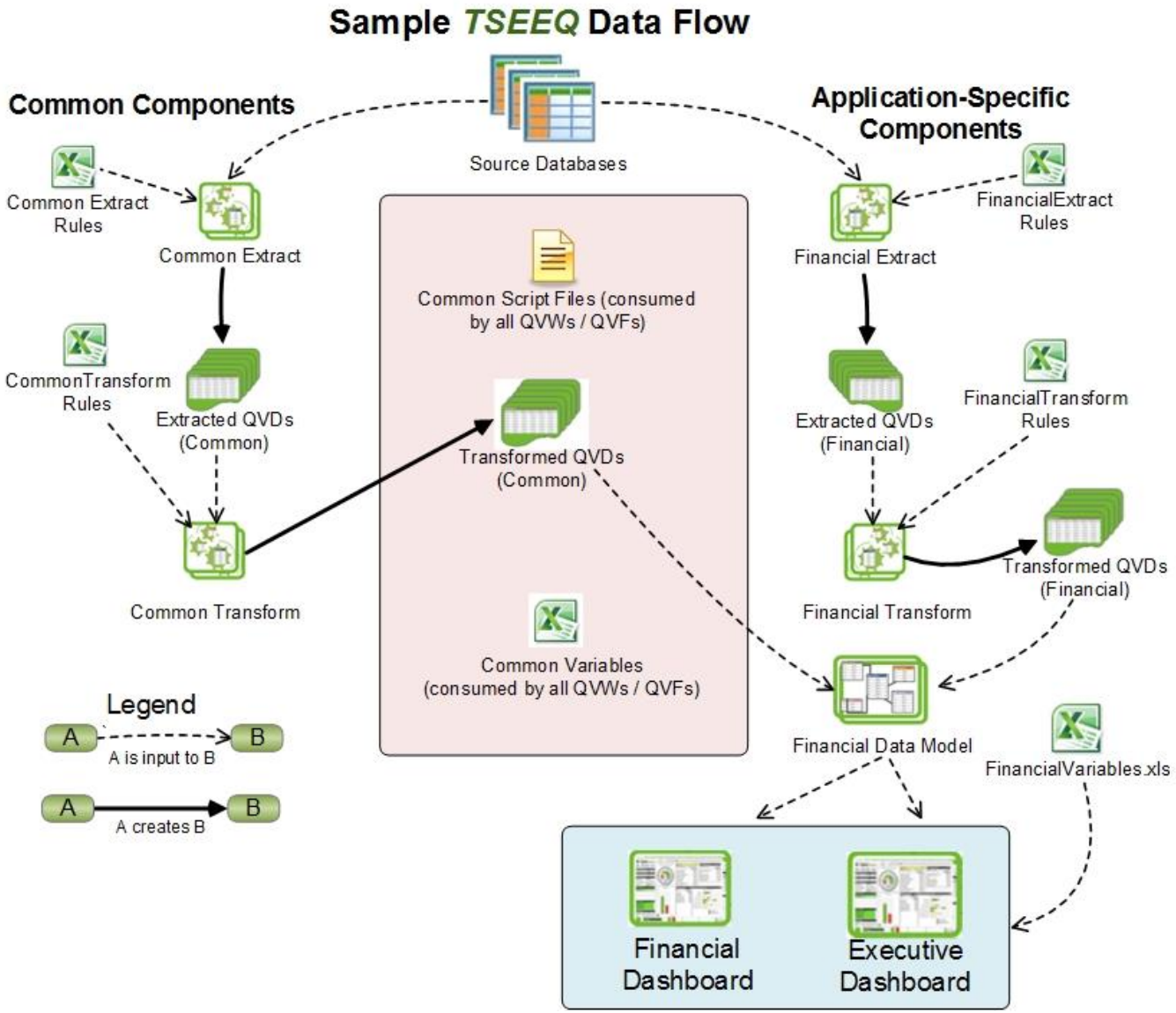
Technical Architecture

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 the tabular metadata store.
   a. The metadata store is by default a set of Excel spreadsheets; a relational database may be used instead of Excel.
   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 next page).

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



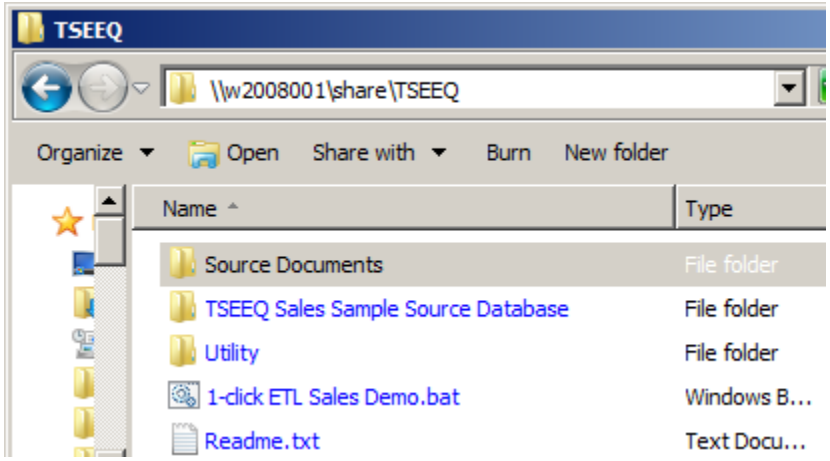
4. Since ***TSEEQ*** is a set of script routines executing in the context of a QVW or QVF, standard Publisher or Qlik Sense tasks are the scheduling mechanism.

Instructions for the ***TSEEQ*** *Sales Sample*:

The *TSEEQ Sales Sample*, contained within *TSEEQ.zip,* 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 with which to implement ***TSEEQ*** ETL flows for additional applications.

File Extraction Instructions (Both Qlik Sense and QlikView)

1. Identify a network share to which your development team has read and write access.
   a. Note that for single developer scenarios (such as prototyping in a private sandbox using QlikView or Qlik Sense Desktop), a local path, ex *C:\TSEEQ*, can be used in lieu of a network share. However, the remainder of this document assumes the use of a network share.
   b. The screen shot at right shows a share named *\\w2008001\share*; your actual share name will likely differ. You can use a sub-folder within a share if preferred.
2. For the rest of this document, ~\ refers to the share or preferred sub-folder within a share that you identified.
3. Extract the TSEEQ ZIP file to ~\.
4. Under ~\, you should then see a ***TSEEQ*** folder with constituent sub-folders, as shown in the screen shot at right.



ETL Execution Instructions (QlikView)

1. For **QlikView**, to run the whole data flow for a 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 process for the *TSEEQ Sales Sample* will then automatically execute.
   b. (Hint: if prompted to with a Save As dialog, just click **Save** to overwrite the old file and click **Yes** when prompted if you want to replace the file.)

ETL Execution Instructions (Qlik Sense)

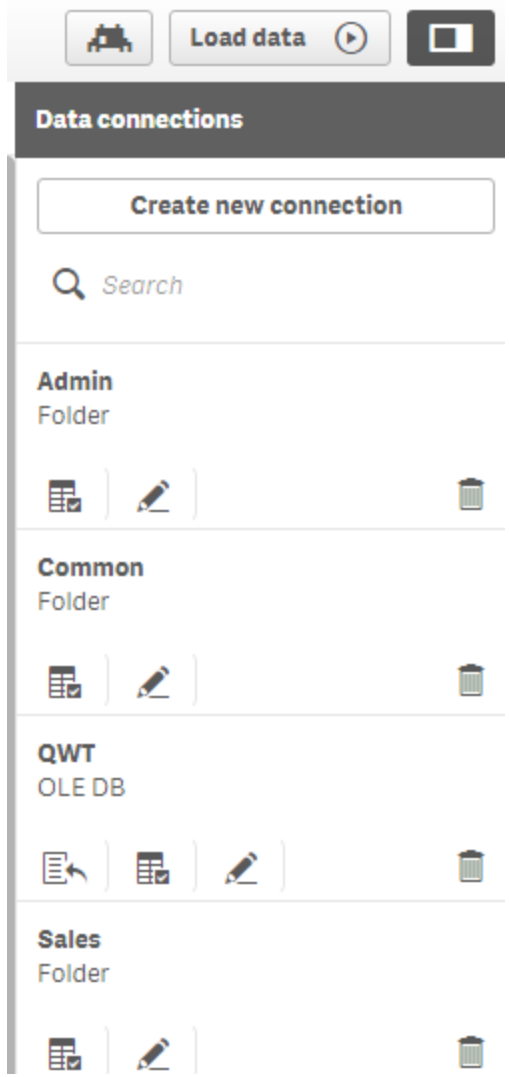
Below, you only need to complete Step 1 OR Step 2, depending on the Qlik Sense product you are using:

1. For **QlikSense Server**, use the QMC to import each of the 3 QVFs under ~*TSEEQ\Source Documents\TSEEQ Sales Sample*, from each of the step-specific sub-folders (*01_Extract*, *02_Transform* and *03_Load*).

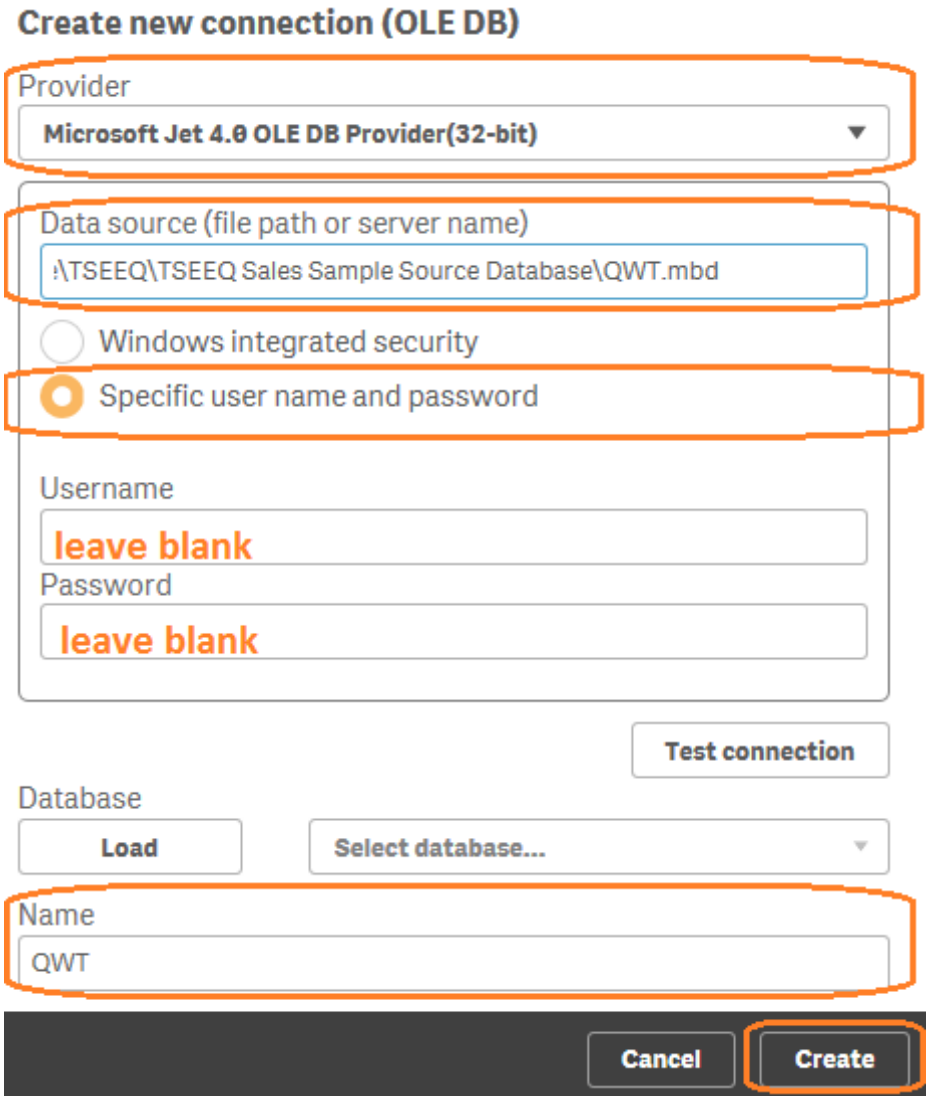
2. For **Qlik Sense Desktop**, you can either:
   a. Copy the ~*TSEEQ* folder to *c:\users\<user id>\Documents\Qlik\Sense\Apps*.
   OR
   b. Point *c:\users\<user id>\user folder>\Documents\Qlik\Sense\Settings.ini* to ~*TSEEQ*.
      - Please see this link for instructions: https://community.qlik.com/thread/158503

      OR
   c. Use a junction point to re-direct *c:\users\<user id>\Documents\Qlik\Sense\Apps* to ~*TSEEQ*.
      - Please see this link for instructions: https://en.wikipedia.org/wiki/NTFS_junction_point#Creating_or_deleting_a_junction_point

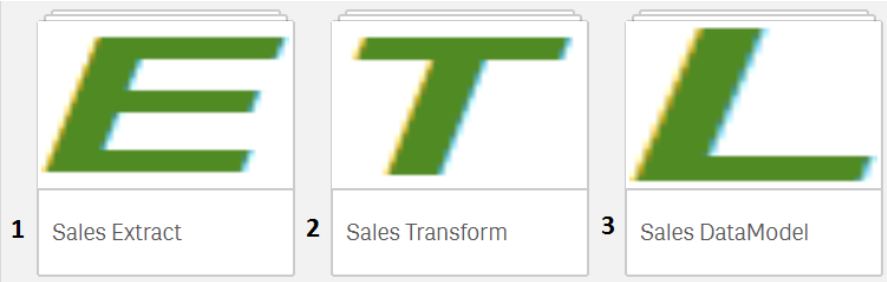
3. Next, create the folder data connections. You will need to create the connections in the **Data Load Editor** (whether in Qlik Sense Desktop or Server). Qlik Sense server users should also use the **QMC** to strip out the *(<directory>_<userid>)* postfix from each connection name**.** Connections 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***.



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**.



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



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 included **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

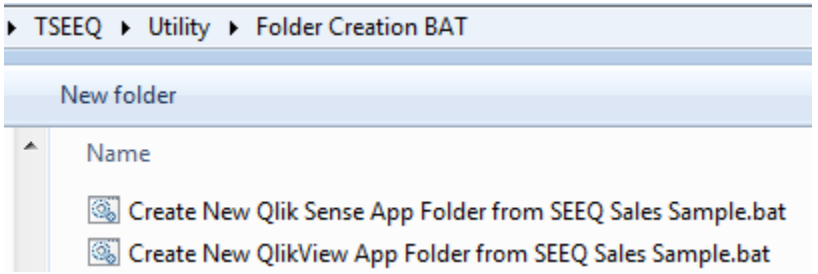
Execution Order	Executed Script	Execution Start Time	Execution Completion Time	Execution Time Elapsed
1	[Accounts]: LOAD BillingCity AS [Billing City],...	07-22-17 01:51:11 PM	01:51:13 PM	00:00:02
2	[Contacts]: LOAD AccountId AS [Account Id],...	07-22-17 01:51:13 PM	01:51:17 PM	00:00:04
3	[CaseOwners]: LOAD FirstName & ' ' & LastName AS [Case O...	07-22-17 01:51:19 PM	01:51:20 PM	00:00:01
4	[Timecards]: LOAD DISTINCT Id AS [Timecard Id],...	07-22-17 01:51:20 PM	01:51:20 PM	00:00:00
5	[Projects]: LOAD (...	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[1]. 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 Server, the folder structure is less relevant than with QlikView; Qlik Sense Server 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.

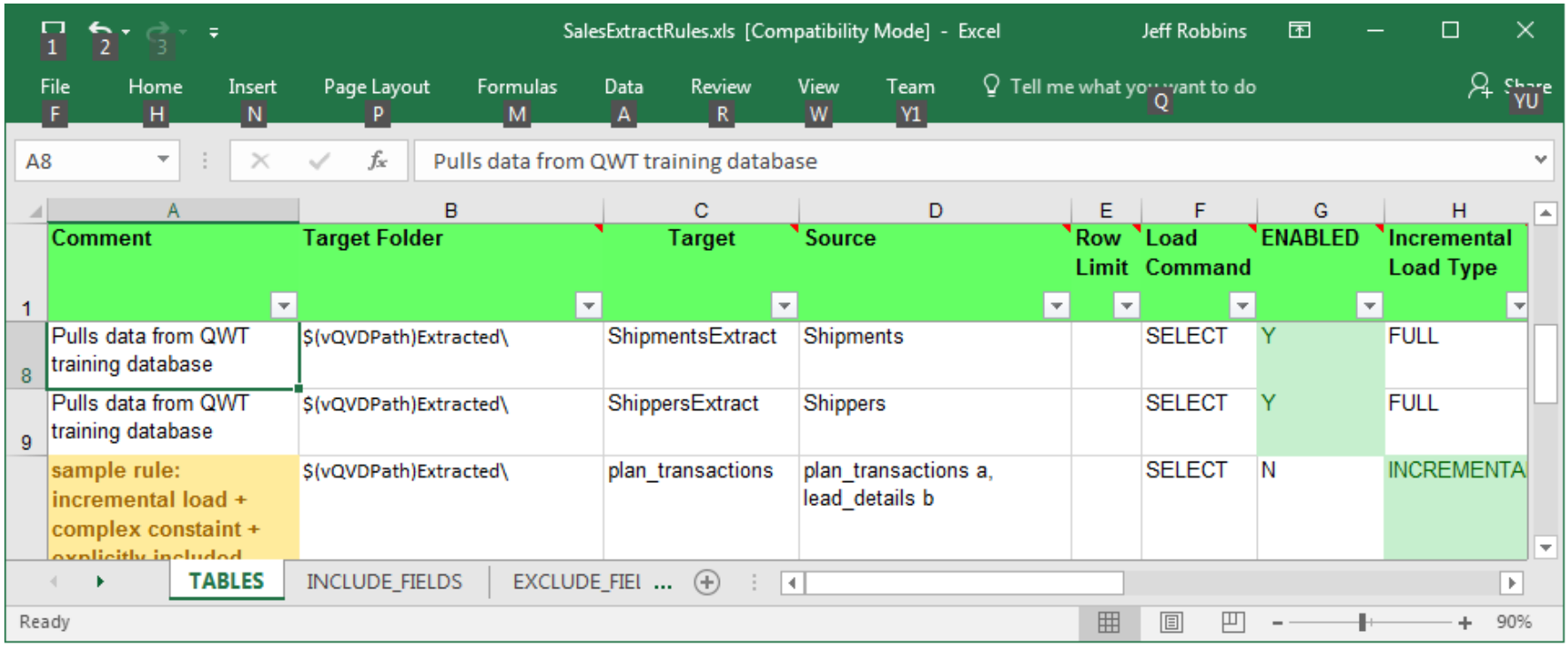


[1] A later section of this document defines terms such as "app" and "application" a bit more explicitly.

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*).



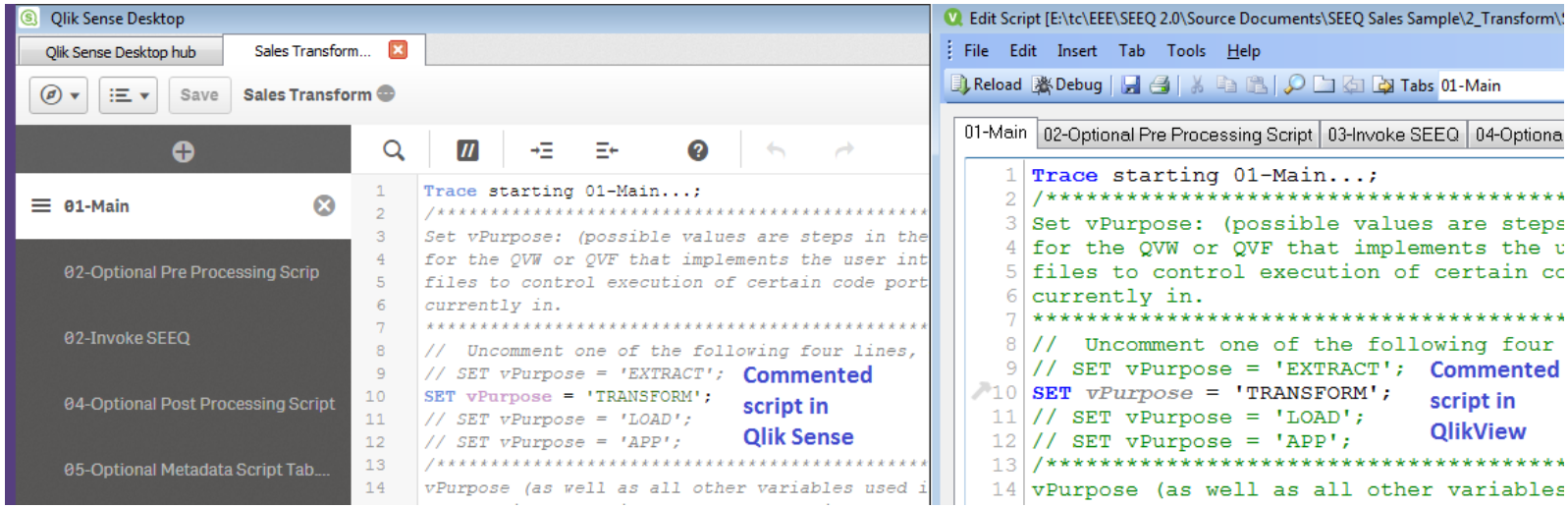
TSEEQ will not generate and execute ETL script rules where **ENABLED** is set to *N*, since a corresponding sample database is not included in the ***TSEEQ*** zip file. However, you may optionally enable those rules by

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.

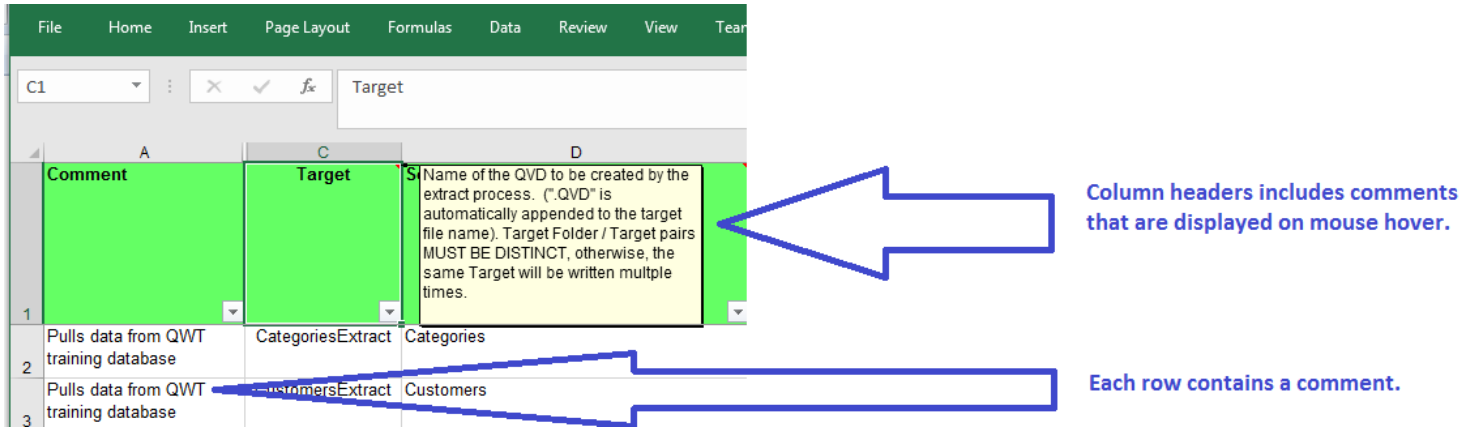
Documentation Within in the ETL QVFs and QVWs

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 in the XLS Files

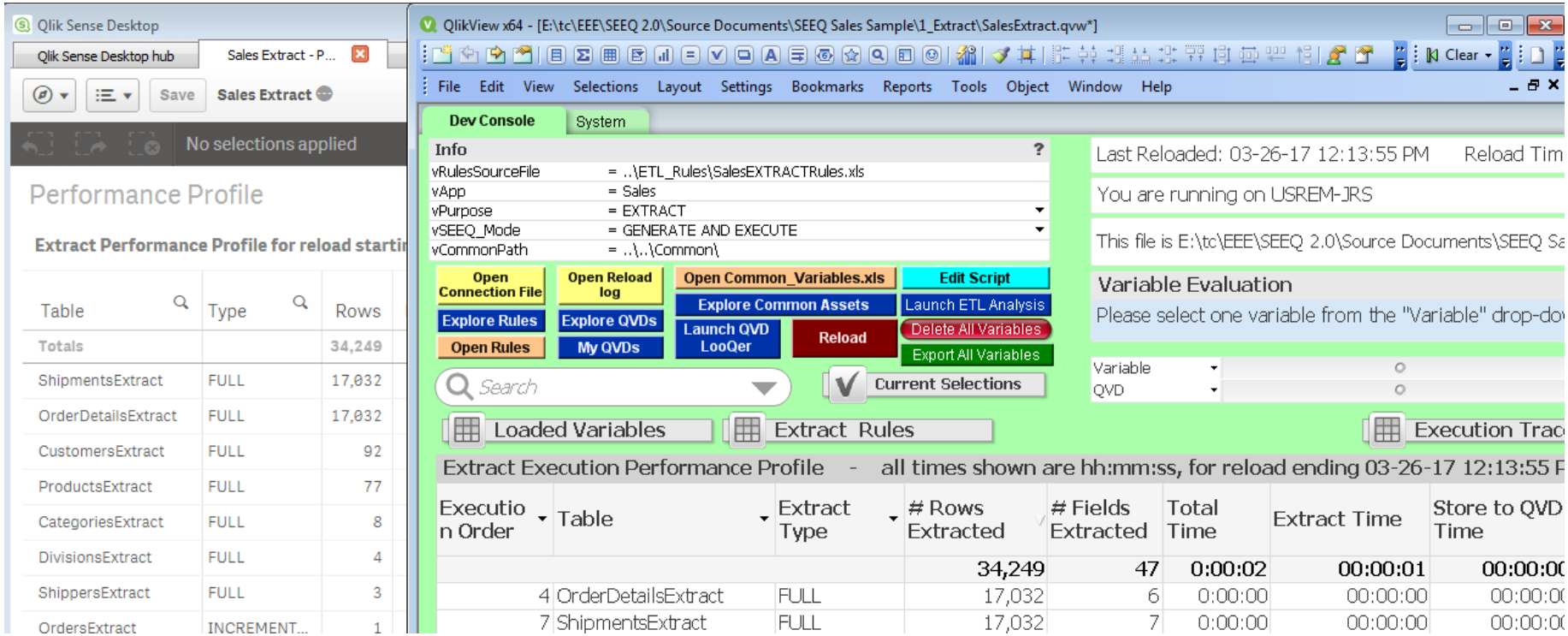
As shown in the screen shot, each column and row within the XLS files in the ***TSEEQ*** *Sales Sample* contains embedded documentation. This documentation is not replicated in its entirety within this document. As such, Qlik developers are referred to the intra-XLS documentation within the ***TSEEQ***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



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 QlikView; a similar sheet is provided in Qlik Sense by the **ETL Analysis** QVF.

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

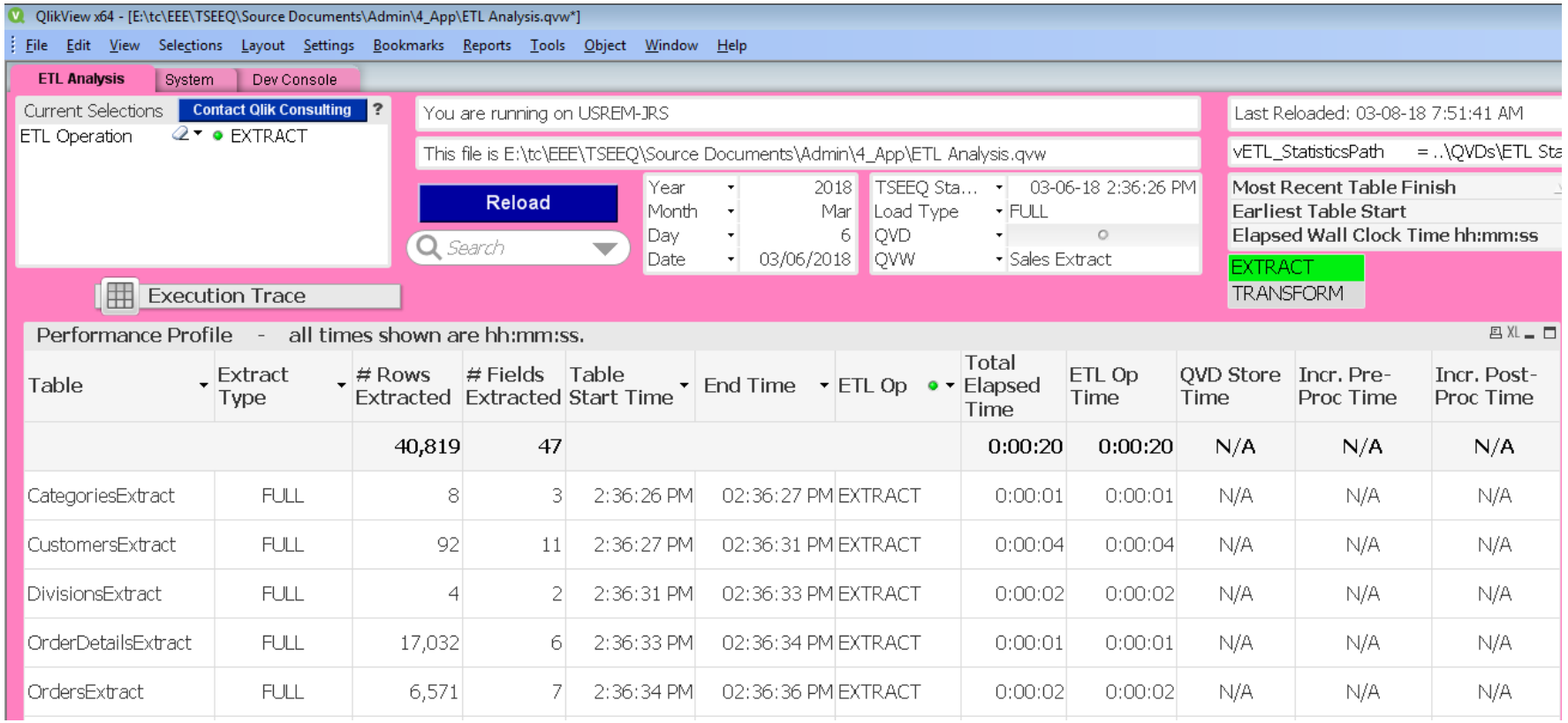
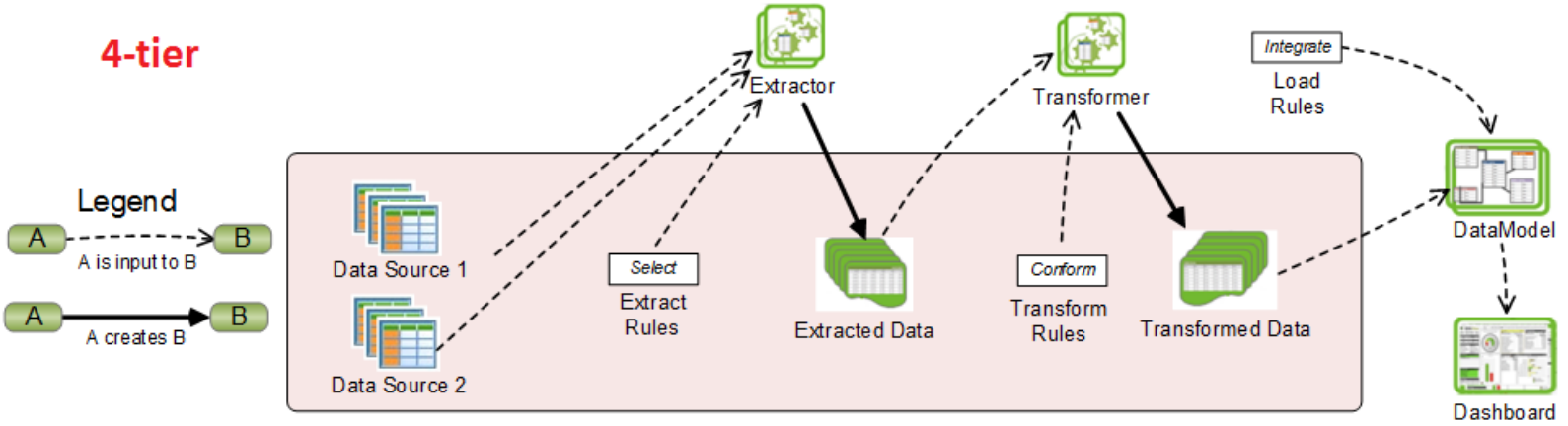


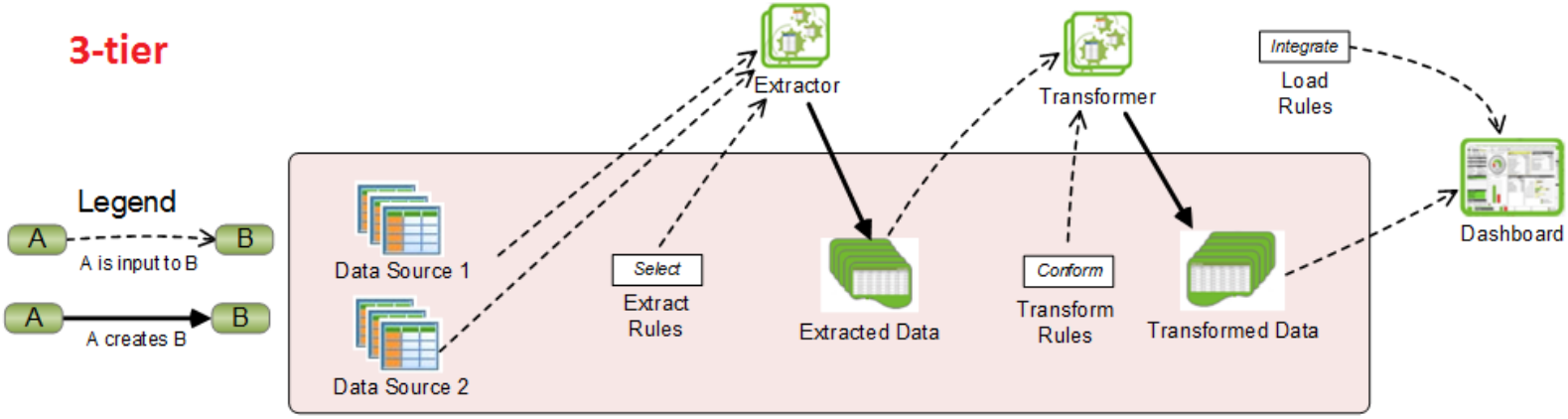
Table	Extract Type	# Rows Extracted	# Fields Extracted	Table Start Time	End Time	ETL Op	Total Elapsed Time	ETL Op Time	QVD Store Time	Incr. Pre-Proc Time	Incr. Post-Proc Time
		40,819	47				0:00:20	0:00:20	N/A	N/A	N/A
CategoriesExtract	FULL	8	3	2:36:26 PM	02:36:27 PM	EXTRACT	0:00:01	0:00:01	N/A	N/A	N/A
CustomersExtract	FULL	92	11	2:36:27 PM	02:36:31 PM	EXTRACT	0:00:04	0:00:04	N/A	N/A	N/A
DivisionsExtract	FULL	4	2	2:36:31 PM	02:36:33 PM	EXTRACT	0:00:02	0:00:02	N/A	N/A	N/A
OrderDetailsExtract	FULL	17,032	6	2:36:33 PM	02:36:34 PM	EXTRACT	0:00:01	0:00:01	N/A	N/A	N/A
OrdersExtract	FULL	6,571	7	2:36:34 PM	02:36:36 PM	EXTRACT	0:00:02	0:00:02	N/A	N/A	N/A

Considerations on 3-Tier vs 4-Tier Data Architectures

TSEEQ allows for both 4-tier (separate files for data model and dashboard) and 3-tier (single file for data model and dashboard) architectures. The 4-tier approach provides more modularization, but in Qlik Sense, 4-tier typically requires that the repository **Apps** folder be mapped to a folder data connection.

As such, the 3-tier approach may be preferred in some cases. Note that in the 4-tier approach, any single data model is a re-usable asset that can consumed by multiple dashboards. In the 3-tier 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.



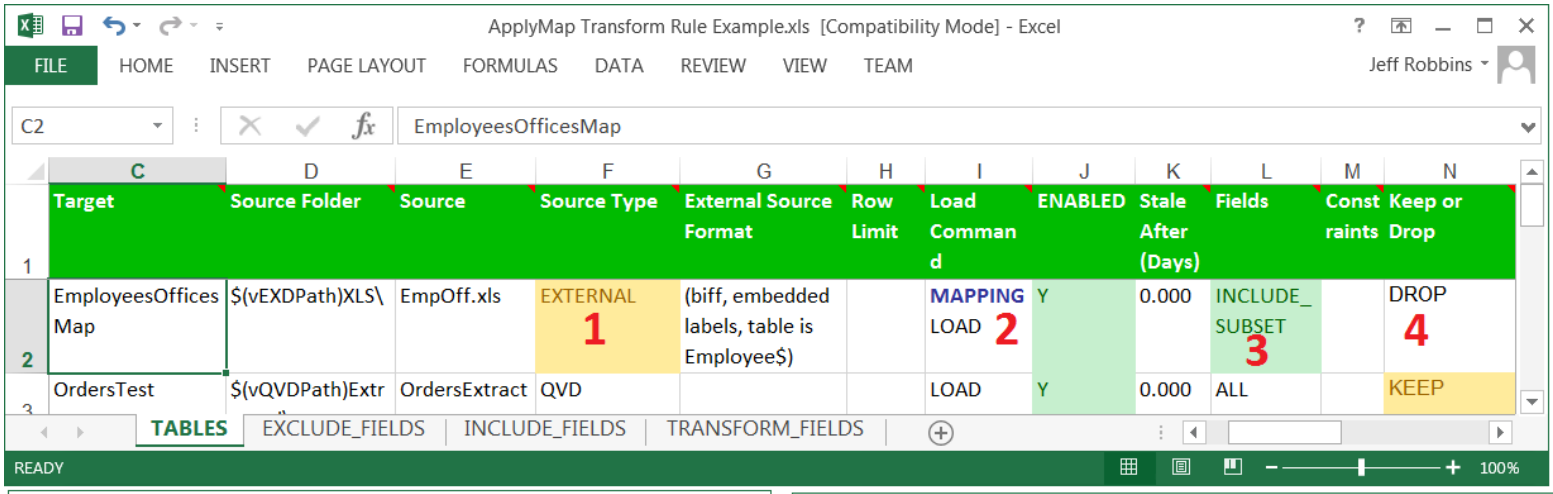


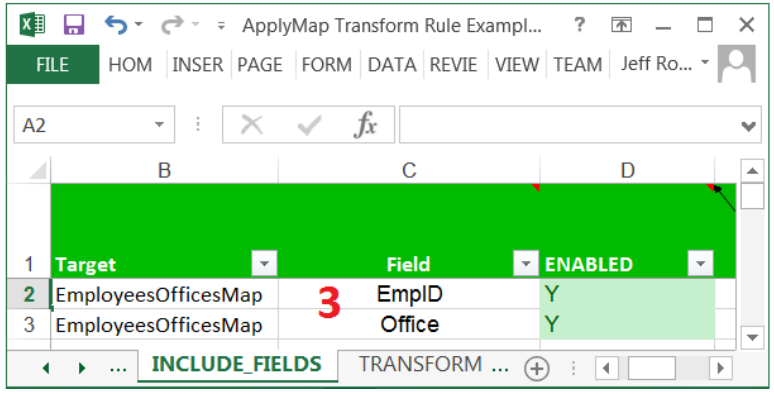
ApplyMap Example

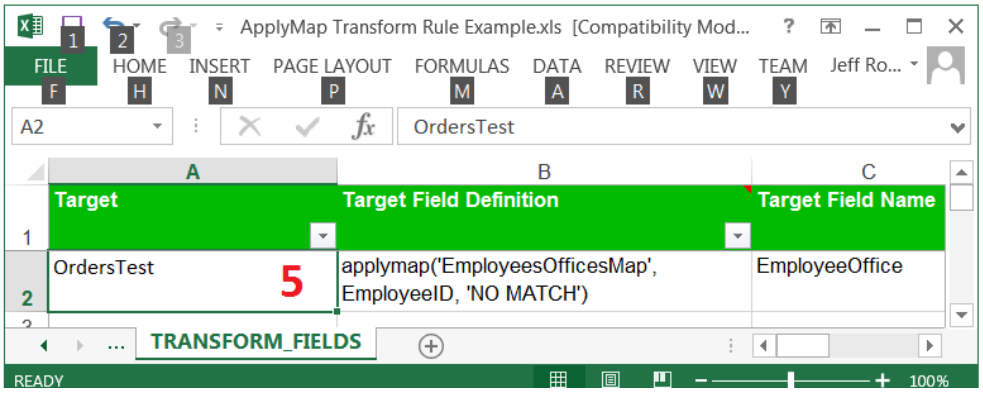
TSEEQ includes an example Transform rule set implementing the Qlik ApplyMap() functionality:

~*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.







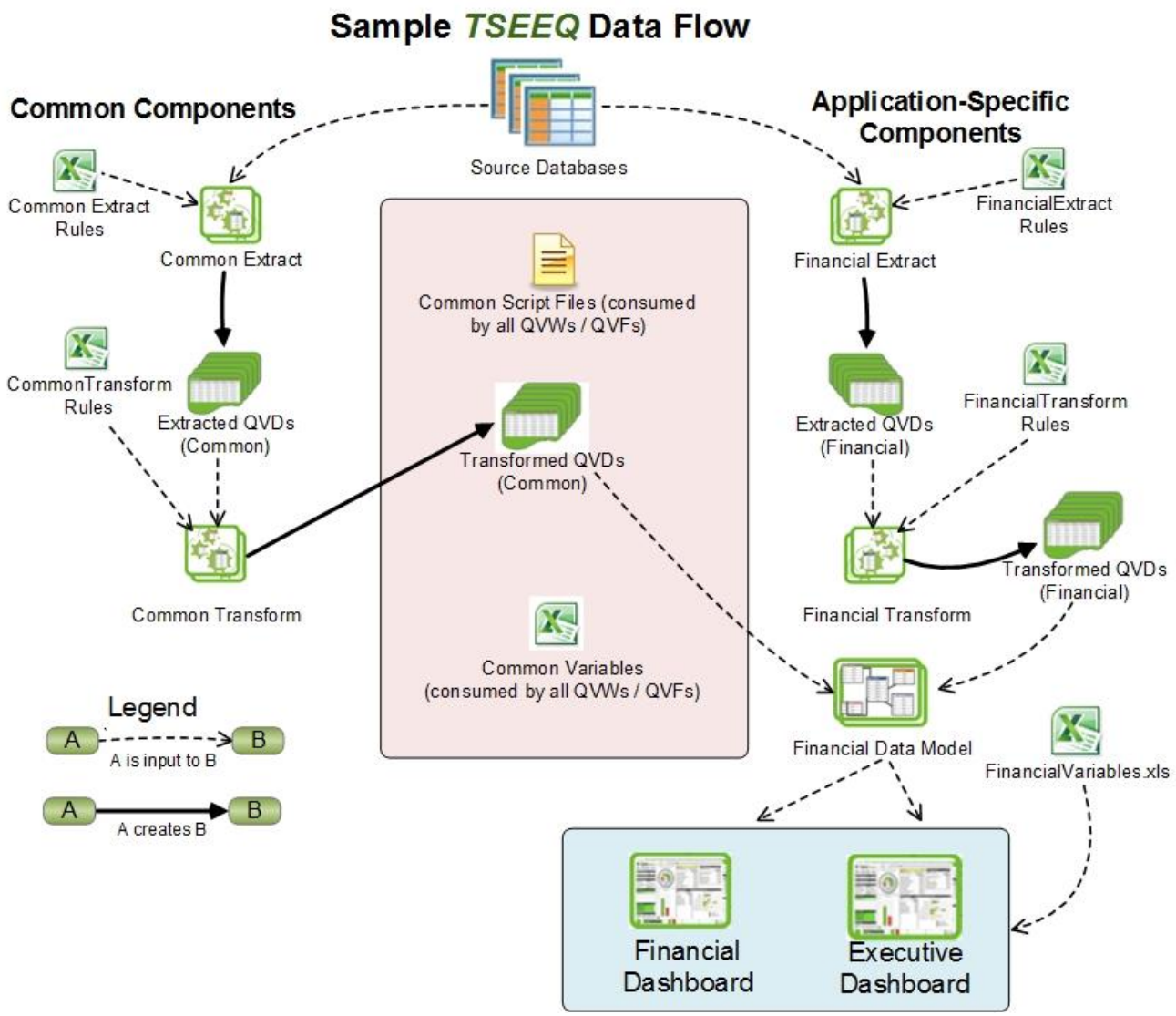
Notes on the *vApp* Variable

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.



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');
```

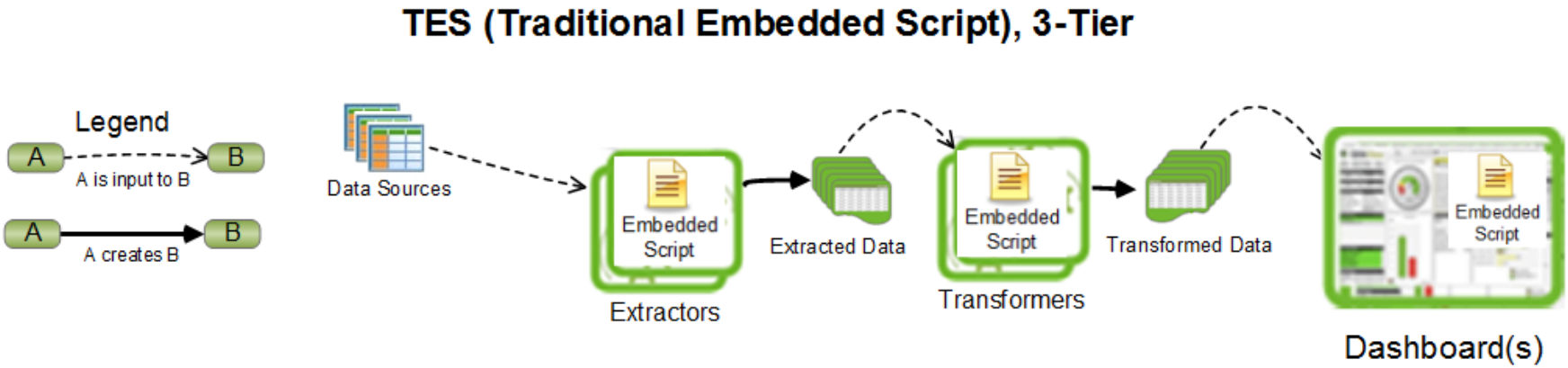
And the same single line of code immediately above could conceivably be used in every QVF that needs to read in *vApp*-specific variables: All Sales QVFs would read the Sales variables; all Financial QVFs would read the Financial variables.

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-tier data architecture as shown in this diagram:



In this 3-Tier **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-Tier 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!

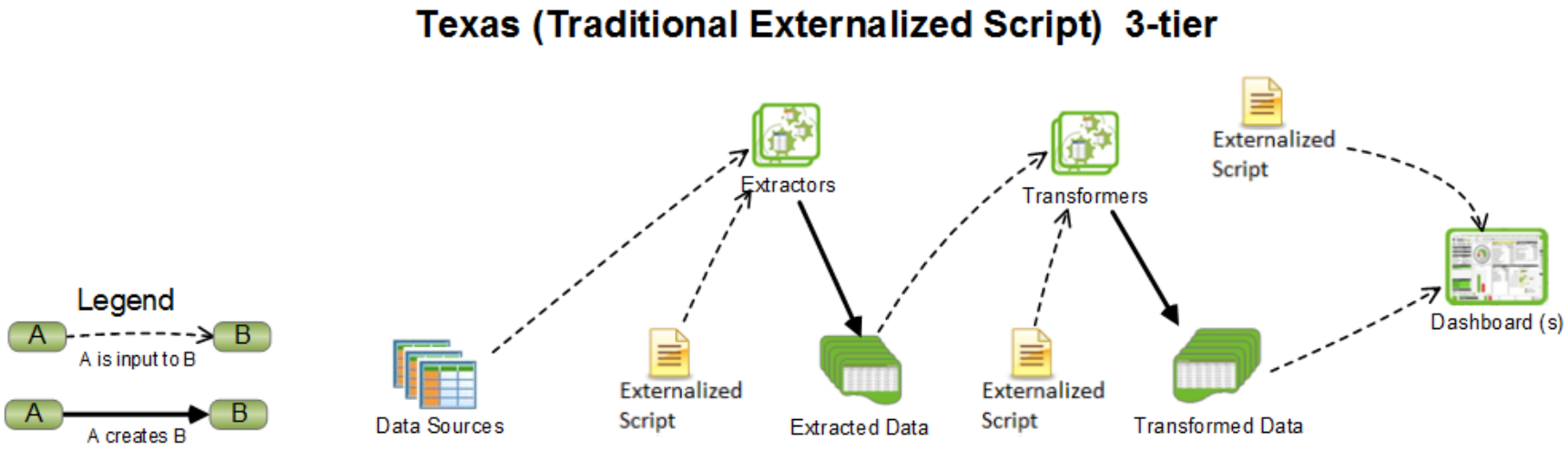
However, embedding script in the Extractors and Transformers results in cross-environment promotion complexity:

- 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[2].

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.

---

[2] 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.



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.

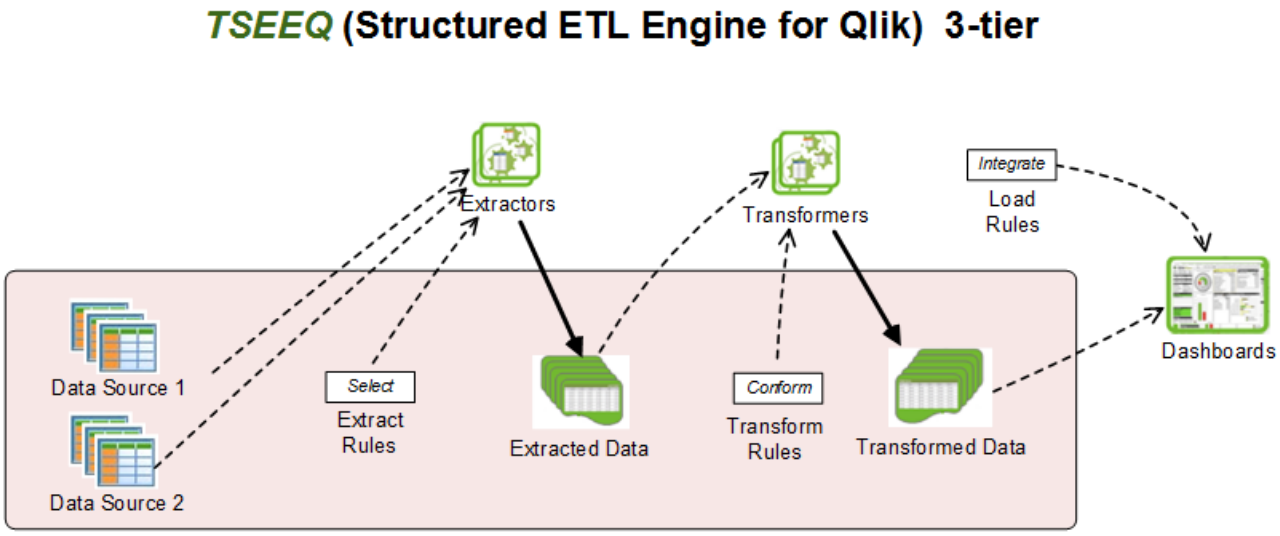
With **TeXas**, cross environment promotion of a given Dashboard rarely requires cross-environment promotion of the associated Extractors and Transformers. Rather, the externalized scripts referenced by the Extractors and Transformers are the assets to be promoted.

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.

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**.

Note that **TES, TeXas,** and ***TSEEQ*** are not mutually exclusive; a hybrid approach is feasible and useful in many cases. Any single Extractor or Transformer “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[3].



[3] 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