Qlik Deployment Framework 1.6.5

Qlik Sense Exercises

October, 2016

Table of Contents

Version 1.6.5

These exercises are intended for QDF version 1.6.5 or later please read *Qlik Deployment Framework-Qlik Sense additional notes.pdf* first, this document includes more details on how to deploy QDF with Qlik Sense. To understand what containers are and used for please read *Qlik Deployment Framework-Getting Started Guide.pdf*. These documents are available on Qlik Community as well as extracted during the deployment framework setup.

Getting started

Exercise number 1, Extract Qlik Deployment framework and create example containers

1. Unzip *QlikDeploymentFramework_Deploy_Tool.zip*
2. Run *QlikDeploymentFramework.exe* file
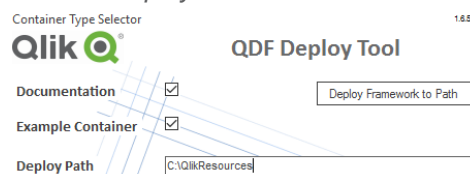
3. Under *Container Type Selector* menu select *Qlik Sense Only* this will extract a container layout adapted for Qlik Sense.
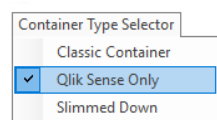
4. Modify Extract location if needed, default is *c:\QlikResources*
5. Press *Deploy Framework to Path* and *Yes* in the popup box, when done an information message will appear. The initial framework will be extracted and presented in the container box.
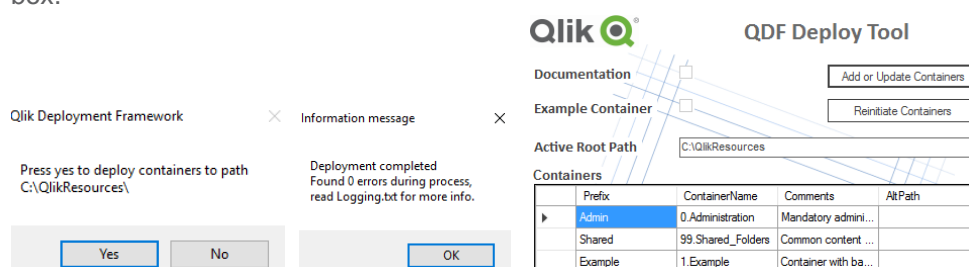
6. Now we can add additional containers, In Containers box type containers prefix and physical container name.
    a. Add a container with the prefix *Corp* with container name *4.AcmeCorp*
    b. Add a system container with the prefix *SAP* with container name *1000.System\1.SAP*
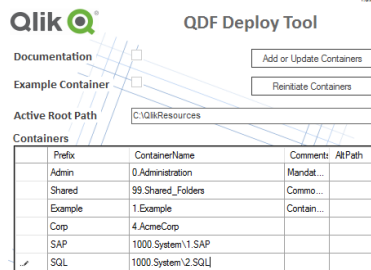   c. Add a system container with the prefix *SQL* with container name *1000.System\2.SQL*

7. Press *Add/Update*.
8. **Hint!** To open an explorer window in the deploy path location double click on the *Deploy path* box.
9. You can always open the tool again and add additional containers.
10. Close Tool with X.

Exercise number 2, basic QDF initiation

The example below is loading the complete framework using a single *folder connection* this is called single LIB mount. There is also the possibility to use individual folder connections for each container, this is called multiple LIB mounts. More in-depth QDF initiation instructions are available in the *Qlik Deployment Framework-Qlik Sense additional notes.pdf* document.

1. From Qlik Sense desktop or server hub create a new *QDF_Exercise* application

2. Open *Data Load* editor

3. Create a new *Folder Connection* in Qlik Sense. This is done by under *Data connections* selecting *Create new connections.*

    Data connections

    Create new connection

    And select *Folder Connection*

    Standard connectors
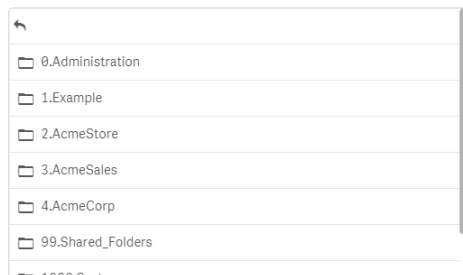
    Folder

    Folder

    ODBC
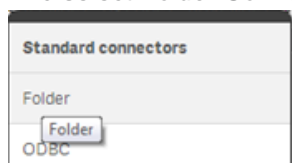
4. Point to the root location of QDF container architecture (root) and give it the name **Root**. Type in UNC or drive path here. The *Folder Connection* name is case sensitive, note that **Root** is a predefined name for single mount mode.

    Create new connection (folder)

    Path

    | C | > | QlikResources | > | |

    ↶

    📁 0.Administration

    📁 1.Example

    📁 2.AcmeStore

    📁 3.AcmeSales

    📁 4.AcmeCorp

    📁 99.Shared_Folders

   📁 1000.System

    Name

    Root

5. Add the QDF initiation script in the first tab just below the locale settings.
    $(Include=lib://Root\InitLink.qvs);

    ```
    1   SET ThousandSep=' ';
    2   SET DecimalSep=',';
    3   SET MoneyThousandSep=' ';
    4   SET MoneyDecimalSep=',';
    5   SET MoneyFormat='#.##0,00 kr;-#.##0,00 kr';
    6   SET TimeFormat='hh:mm:ss';
    7   SET DateFormat='YYYY-MM-DD';
    8   SET TimestampFormat='YYYY-MM-DD hh:mm:ss[.fff]';
    9   SET FirstWeekDay=0;
    10  SET BrokenWeeks=0;
    11  SET ReferenceDay=4;
    12  SET FirstMonthOfYear=1;
    13  SET CollationLocale='sv-SE';
    14  SET CreateSearchIndexOnReload=1;
    15  SET MonthNames='jan;feb;mar;apr;maj;jun;jul;aug;sep;okt;nov;dec';
    16  SET LongMonthNames='januari;februari;mars;april;maj;juni;juli;augusti;september;oktober;november;d
    17  SET DayNames='mån;tis;ons;tor;fre;lör;sön';
    18  SET LongDayNames='måndag;tisdag;onsdag;torsdag;fredag;lördag;söndag';
    19  $(Include=lib://Root\InitLink.qvs);
    ```

    Create new connection

    🔍 *Search*

    Root
    Folder
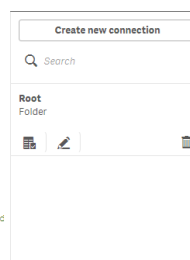
    This initiation is for single mount mode in the load editor, optional ways of initiation QDF are available in the *Qlik Deployment Framework-Qlik Sense additional notes.pdf* document.

6. Run the script, a similar text like below should be presented stating that single LIB mount have been identified.
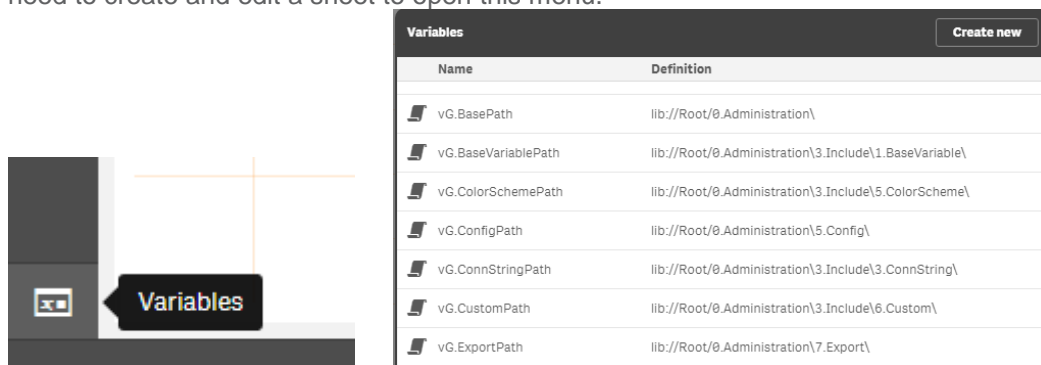
```
Started loading data

'### DF InitLink Started, trying to link to 1.Init.qvs
script'

'### DF Info, identified Sense root path lib://Root/
(single LIB mount)'

'### DF Info, identified Sense home container
lib://Root/0.Administration\'
```

7. Also Global Variables (*vG.xxxPath*) pointing to *lib://Root* should be available in variables. You need to create and edit a sheet to open this menu.

Variables		Create new
Name	Definition	
vG.BasePath	lib://Root/0.Administration\	
vG.BaseVariablePath	lib://Root/0.Administration\3.Include\1.BaseVariable\	
vG.ColorSchemePath	lib://Root/0.Administration\3.Include\5.ColorScheme\	
vG.ConfigPath	lib://Root/0.Administration\5.Config\	
vG.ConnStringPath	lib://Root/0.Administration\3.Include\3.ConnString\	
vG.CustomPath	lib://Root/0.Administration\3.Include\6.Custom\	
vG.ExportPath	lib://Root/0.Administration\7.Export\	

Variables

Function library in combination with global variables

QDF has a built in and very powerful function library, these exercises will show some of these functions. Documentation of the functions and syntax can be found in *Qlik Deployment Framework-Development Guide.pdf*.

Exercise number 1. DoDir function

Dodir will load files and/or folders in an optional table, syntax is
Call DoDir (Scan Path, [Table Name], [Folders Only], [Single Folder], [Qualified Felds], [Hide QDF Templates])
Scan Path It the folder/file structure to scan
Table Name Is the Table name, optional default name is *DoDirFileList*
Folders Only Is an optional switch if set to *'true'* only folders will be returned
Single Folder Is an optional switch if set to *'true'* only one single folder will be indexed
Qualified Felds Is an optional switch if set to *'true'* all field named will be Qualified *based on the Table Name*

1. Open Qlik Sense QDF_Exercise example application we just created. In the script below initiation (*InitLink*) add the command: *Call DoDir ('$(vG.BasePath)');*

2. The global variable *vG.BasePath* represents the base of your home container and should thereby list all files in the *0.Administration* container as we have not states a particular home.

3. Open Data Model Viewer to see the *DoDirFileList* Table *(*default).

DoDirFileList
FullyQualifiedName
DoDirFileSize
DoDirFileTime
DoDirFileName
DoDirContainerPath
DoDirFileExtension
DoDirFileNameCount

Preview of data

FullyQualifiedName	DoDirFileSize	DoDirFileTime	DoDirFileName
lib://Root/0.Administration\2.QVD\Info.txt	441	2016-08-11 14:51:14	Info.txt
lib://Root/0.Administration\0.Template\1.ContainerTemplate\5.Config\Index\Info.txt	500	2016-08-11 14:51:14	Info.txt
lib://Root/0.Administration\Info.txt	443	2016-08-11 14:51:14	Info.txt
lib://Root/0.Administration\9.Misc\Info.txt	368	2016-08-11 14:51:14	Info.txt

4. Change operators to *Call DoDir ('$(vG.BasePath)','MyTable','True');*
   You should now get a table named *MyTable*, the third parameter (True) will only load in folder names.

Exercise number 2, Link containers (mount)

One of the main pillar in QDF is to section content into containers. Containers can be reorganized (moved) without breaking any script code. That's why we are linking (or mounting) from the home container to any other containers (I have security access to) using the LCGV function. LCGV creates one or more global variables pointing to the remote container location.

Call LCGV ('Container Prefix', [' Specific folder [;Additional folders separated by ;]);

In this example you are an administrator that want to collect QVD data from the *Example* container.

1. Open Qlik Sense QDF_Exercise example application we just created. In the script below initiation (*InitLink*) add call LCGV`('Example','QVD');`

2. Run the script, you should get this line:
   `'### DF Info, found Example Container'`

3. In Variables check that the Global Variables for Example container and QVD storage is available.

   vG.ExampleQVDPath   lib://Root/1.example\2.QVD\

4. Also check that the URL is correct

5. Delete the variables

6. Last try call LCGV`('Example','QVD;Custom');` this will create two global variables, one for *QVD* and the other for *Custom (*in the *Example* container).

   vG.ExampleCustomPath   lib://Root/1.example\3.Include\6.Custom\

   vG.ExampleQVDPath   lib://Root/1.example\2.QVD\

7. Last run the DoDir function in the *Example* QVD repository to search for any available qvd files. *Call DoDir ('$(vG.ExampleQVDPath)*.qvd');*

8. As we are adding wildcard in the path there is a potential risk that if *vG.ExampleQVDPath* is missing the entire filesystem would be scanned. This could be avoided by using if statement to validate the global variable.
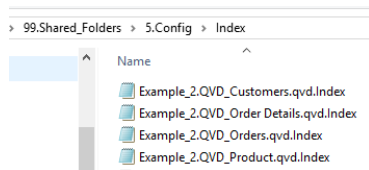

Exercise number 3, IndexAdd

There are some very powerful qvd indexing functions available in QDF, these exercises will dig deeper into this.
IndexAdd is a function that creates and maintains a central index based on meta-data from available qvd files. This metadata can afterwards be used to load and monitor data.

Call IndexAdd(['QVD path and name'],['Index folder name'] ,['Container name'] ,['Tags'] ,['Alternative Index path'])

- **QVD path and name** (Optional) Path and name of QVD file to Index, wild card (*) is supported
- **Index folder name** (Optional) Place the Index in a specific folder, default is to use qvd folder name
- **Container name** (Optional) Specify the QVD files container name, this is usually identified automatically
- **Tags** (Optional) Add index tag, recommendation is to use the *comment table* function instead as this will be more persistent.

1. Open Qlik Sense *QDF_Exercise* example application we just created

    In the script after **call LCGV**(*'Example'*,*'QVD;Custom'*); (that we added the last exercise), add

    call IndexAdd(*'$(vG.ExampleQVDPath)*'*);

2. Run the load script, *you* should have no errors

3. Open the 99.Shared_Folders container and drill under *5.Config\Index*, you should see 4 index files

    > 99.Shared_Folders > 5.Config > Index

    Name ^

    Example_2.QVD_Customers.qvd.Index
    Example_2.QVD_Order Details.qvd.Index
    Example_2.QVD_Orders.qvd.Index
    Example_2.QVD_Product.qvd.Index

4. This is the index that was created using IndexAdd (one for each qvd file), open one index file and you can read the qvd meta-data.

5. Under *0.Administration* container and *1.Application\3.IndexMonitor* folder there is a *IndexMonitor.qvf* file, open this file, connect the Shared lib to the Shared container, load and you will see the meta-data.

Data Tags	▶	Index Storage Name	▶	○ Field Names		○ File Name
				Price		Customers.qvd
Container Name	▶	QVTableName	▶	ProductID		Order Details.qvd

QVD Name ○	Table Name ○	Container Name ○	Nbr records	Nbr Fields	Time
Totals			**3153**	**40**	**-**
Customers.qvd	Customers	Example	91	11	2016-08-12 11:32:45
Order Details.qvd	Order Details	Example	2155	5	2016-08-12 11:32:46
Orders.qvd	Orders	Example	830	14	2016-08-12 11:32:46
Product.qvd	Product	Example	77	10	2016-08-12 11:32:45

Exercise number 4, IndexLoad

The second index function is IndexLoad that loads data based index criteria's like tags and field names.

Call IndexLoad(['QVD file name'],['Table Name'] ,['Index Folder Name'] ,['Tags'] ,['Fields'] ,['Index Only'] ,['Load Max Rows'] ,[['Alternative Index path'],['Load Expressions'])

- *QVD file name* (Optional) Name of QVD to load, wild cards (*01-2015*) is supported
- *Table Name* (Optional) Load in a table, can be combined with *QVD file name*
- *Index Folder Name* (Optional) use this specific index only, can be combined with *QVD file name*
- *Tags* (Optional) load data containing a specific tag, can be combined with *QVD file name*
- *Fields* (Optional) load selected fields separated by comma (,) can be combined with *QVD file name*
- *Index Only* (Optional) will only load in the Index, true will use default table name *(vL._QVDIndexTable).*
  Type table name from default *vL._QVDIndexTable*. This is used when developing apps where the Index is needed.
- **LoadMaxRows** (Optional) will limit how many rows that can be loaded. This will only stop sequential QVD file to load a big QVD will probably load above this limit.
- *Alternative Index path* (Optional) will change the default Index path *($(vG.SharedConfigPath)/Index)* This is not recommended as all functions would need the alternative path specified

- **Load Expressions** (Optional) If set to true it will load related expressions based on tags and container collected from Index, this operator calls the *LoadVariableCSV* function.

    1. Replace the *IndexAdd* statement with *IndexLoad*, in this first example we will load in the Orders table.
    call IndexLoad('','Orders');

    2. Open Data model viewer, only orders *table* should have been loaded in

    3. No lets load in some field data, replace above function with this one you should get the below data model
    call IndexLoad('','','','','OrderID,CustomerID,EmployeeID');

    4. By using *data tags* stored in the meta-data (using comment table) we can group and load data models and time periods. More on this in Exercise no 5.

    5. Run the function below to load *OrderData* tagged data.
    call IndexLoad('','','','' OrderData');

BONUS Exercise number 5, Add data tags

Data tags is a way of grouping qvd data files into data models or/and time periods. Multiple tags can be added and used with the IndexLoad function. First let's add some data tags, for this we need to *comment* tables and after store to qvd files, last we need to index the stored qvd files to collect the metadata.

    1. Open Qlik Sense QDF_Exercise example application, keep the function
    call IndexLoad('','','','OrderData'); that we added in the last exercise.

    2. Add comment table on both Orders and Order Details tables, we add the data tag MyDatamodel, here we can add multiple tags.
    Comment Table [Orders] with 'MyDatamodel';
    Comment Table [Order Details] with 'MyDatamodel';

    3. Store the tables as qvd files in home qvd path
    Store [Orders] into '$(vG.QVDPath)\MyData1.qvd';
    Store [Order Details] into '$(vG.QVDPath)\MyData2.qvd';

    4. Last index the qvd files:
    call IndexAdd('$(vG.QVDPath)*');

    5. To load in the indexed qvd files run the IndexLoad function below, remember to clear or comment out the earlier commands.
    call IndexLoad('','','','MyDatamodel');

Exercises on Custom Global Variables

In the QDF you can create variables that are reusable across several applications, these variables are called custom global variables. Custom global variables should use the prefix (*vG.*) this to separate global variables against application specific variables.

Exercise number 1, Create Custom Global Variables

Custom Global Variables are stored in *Custom.Variable.csv* (default) and loaded in during initiation, observe that it's only 'your home container' custom variable file that is loaded in. Variable files are stored under *3.Include\1.BaseVariable* within each container. The *Custom.Variable.csv* file under *Shared* container is loaded in by all applications within the framework during initiation.

1. Open the 99.Shared_Folders container and drill under *3.Include\1.BaseVariable*, open *Custom.Variable.csv* with your favorite editor (Sublime or Notepad++ works best) **do not use Excel as it distorts the variable file**.

2. You should see an empty file with 4 columns.

```
1   VariableName,VariableValue,Comments,Priority
2   ,,,
3   ,,,
4   ,,,
5   ,,,
6   ,,,
7
```

3. On line two Enter the Variable Name LET vG.TestVariable1

4. Add Variable Value *10+10* (no space) it should look like this:

```
1   VariableName,VariableValue,Comments,Priority
2   LET vG.TestVariable1,10+10,,
```

5. Enter second Variable Name *LET vG.TestVariable2*

6. Enter Variable Value *vG.TestVariable1+10* (no space) and save

7. Load Qlik Sense *QDF_Exercise* example application

8. *vG.TestVariable1* and *vG.TestVariable2* should have the values 20 and 30

vG.TestVariable1	20
vG.TestVariable2	30

9. Add a third variable, *SET vG.TestVariable3* and Variable Value *10+10* Load,

vG.TestVariable1	20
vG.TestVariable2	30
vG.TestVariable3	10+10

   vG.TestVariable3 have value 10+10 as we used SET instead of LET

10. Try a more advance expression, add the variable and value below into *Custom.Variable.csv*. (SET is default operator so we can leave that out)

    vG.TestVariable4 = "TextBetween('$(vG.ApplicationPath)','.','\') &'_'&Left(DocumentName(),len(DocumentName())-4)"

The variable file should now look something like this:

```
1  VariableName,VariableValue,Comments,Priority
2  LET vG.TestVariable1,10+10,,
3  LET vG.TestVariable2,vG.TestVariable1+10 ,,
4  SET vG.TestVariable3,10+10 ,,
5  vG.TestVariable4,"TextBetween('$(vG.ApplicationPath)','.','\') &' '&Left(DocumentName(),len(DocumentName())-4)",,
```

After reloading the Qlik Sense application *vG.TestVariable4* should look something like this:

vG.TestVariable4

```
TextBetween('$(vG.ApplicationPath)','.','\')
&'_'&Left(DocumentName(),len(DocumentName())-4)
```

6. Add a Text Object and put in the variable *$(vG.TestVariable4)* as measure
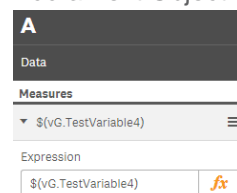
7. In the text box you should see something like *Administration_ QDF_Exercise* where the first concatenation is our home container.

Exercise number 3, attach global expression to a data model

We will continue by loading in some data and add expression to this data based on tags, this is possible to do in one single line.

1. Replace the earlier functions (except the InitLink initiation) with an IndexLoad statement that will load data based on **Sales** criteria (data tagged as Sales) and at the same time load in expressions based on the same criteria (expressions using the **Sales** tag).
   call IndexLoad('','','Sales','','','','Sales');
   *This command will load **Sales** tagged qvd data and add **Sales** tagged expressions*

2. Also lets add a calendar into the data model using another built-in function *CalendarGen*
   call CalendarGen('OrderDate');

3. This should create a data-model looking something like this.

4. The *Indexload* statement above will also load variables within the Example container that are tagged as KPI.
   In Example container under *1.Example\3.Include\1.BaseVariable* open the variable file named *Sales.Variables.csv,* here we have some variables tagged as Sales

```
1  VariableName,VariableValue,Comments,Priority
2  vKPI.SumYear1997,"Sum(If(Year(OrderDate)=1997, Quantity*UnitPrice))","Sum last year",Sales
3  vKPI.SumYear1998,"Sum(If(Year(OrderDate)=1998, Quantity*UnitPrice))","Sum this year",Sales
4  vKPI.OrderValue,"Sum(Quantity*UnitPrice)",Sales & Avg order value,Sales
5  vKPI.SalesIndex,"($(vG.SumYear1998)/$(vG.SumYear1997))",Sales Index summary,Sales
```

5. In the application check the variable view for these KPI variables, for example:

vKPI.OrderValue	Sum(Quantity*UnitPrice)
vKPI.OrderValue_Comments	Sales & Avg order value
vKPI.SalesIndex	($(vG.SumYear1998)/$(vG.SumYear1997))

Notice that an additional variable (_comments) have been added for each comment, this can be used for labels.

6. Let's add a **line chart** object based on the expressions loaded in. Drag the line chart onto the canvas.

7. Add **OrderData** calendar (should have an icon) that we created earlier as Dimension
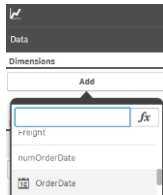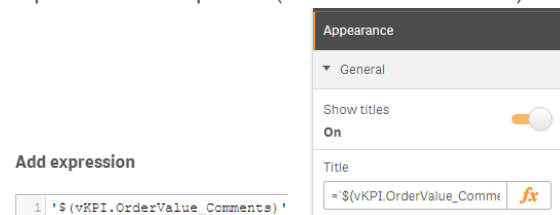

8. On the right add a measure, press the FX button to open the expression view. Type the *$(vKPI.OrderValue)* variable dollar expansion.
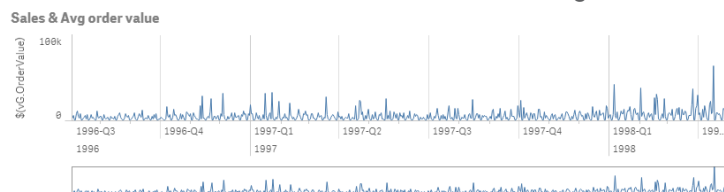

9. Last add the comment variable ='$(vKPI.OrderValue_Comments)' as Title using dollar expansion and quotas (as this is a text field).

When done the **line chart** should look something like this.

1. Create a new Variable file in 99.*Shared_Folders* container by adding the file *Test.Variable.csv* under 99.Shared_Folders*3.Include\1.BaseVariable.*

2. We need to copy the field names from *custom.variables.csv* as well into your new *Test.Variable.csv.*

3. Add some variables and values, last in the *Priority* field type *Sales* as shown below

```
VariableName,VariableValue,Comments,Priority
LET vG.TestVariable5,10+10,,Sales
LET vG.TestVariable6,vG.TestVariable1+10,,Sales
```

4. To use this variable file in your scripts run the sub function call below that will search for the Sales identifier within the Shared container.
call LoadVariableCSV('','Sales','Shared');