Loading Excel File into RED Warehouse

WhereScape RED can access and load into warehouse virtually any type of data. One of the most common sources that data loaded from is Excel file. WhereScape can quite gracefully handle almost all the cases of Excel data sources.

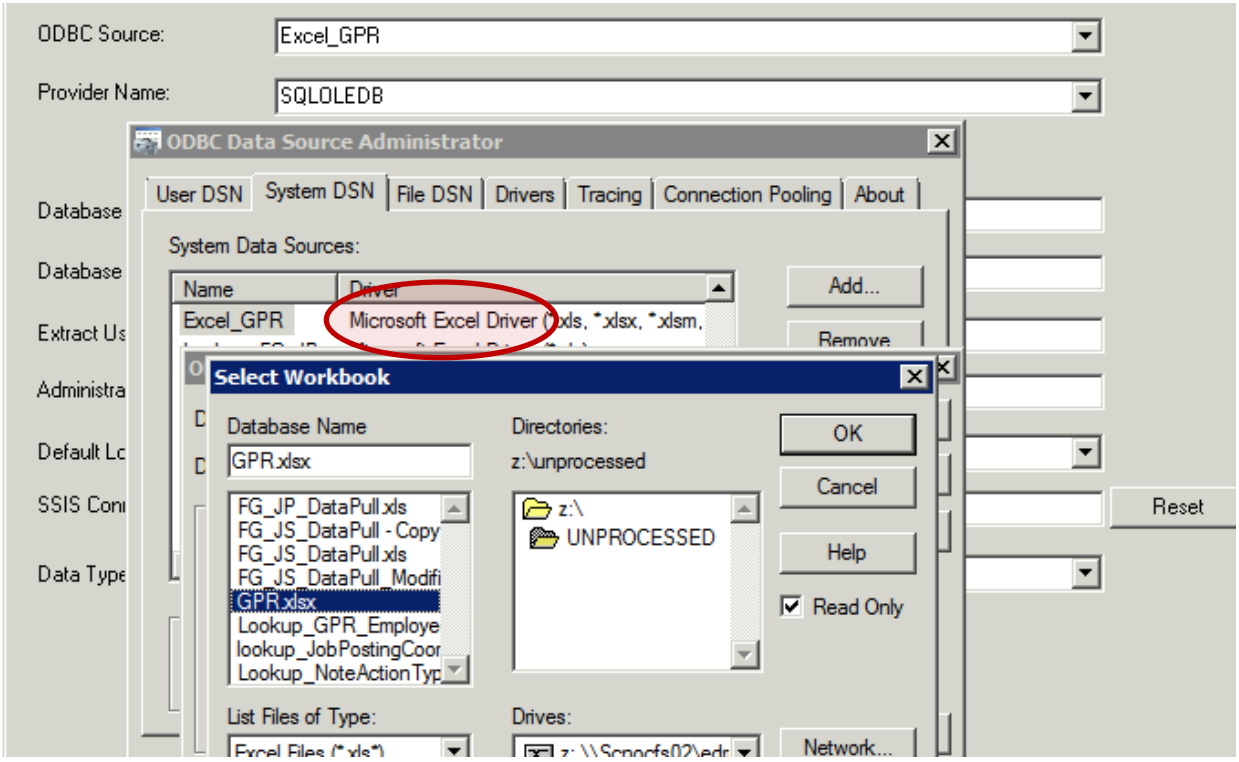
WhereScape has accumulated vast knowledge in dealing with multitude of data sources, including Excel files of all shapes and sizes. Most of this knowledge is embedded in the software itself, making it easy for our customers to get the needed data, its metadata and load it into WhereScape RED warehouse.

RED connections -- a first RED object created -- are used, among other purposes, to browse source data to obtain metadata and to load data into RED warehouse. WhereScape RED provides several types of connections to access Excel spreadsheets. While choosing the right method of loading Excel file, the consideration must be made based on volume and speed of loading, available connectivity options, ability to handle inconsistent data, error capturing, etc.

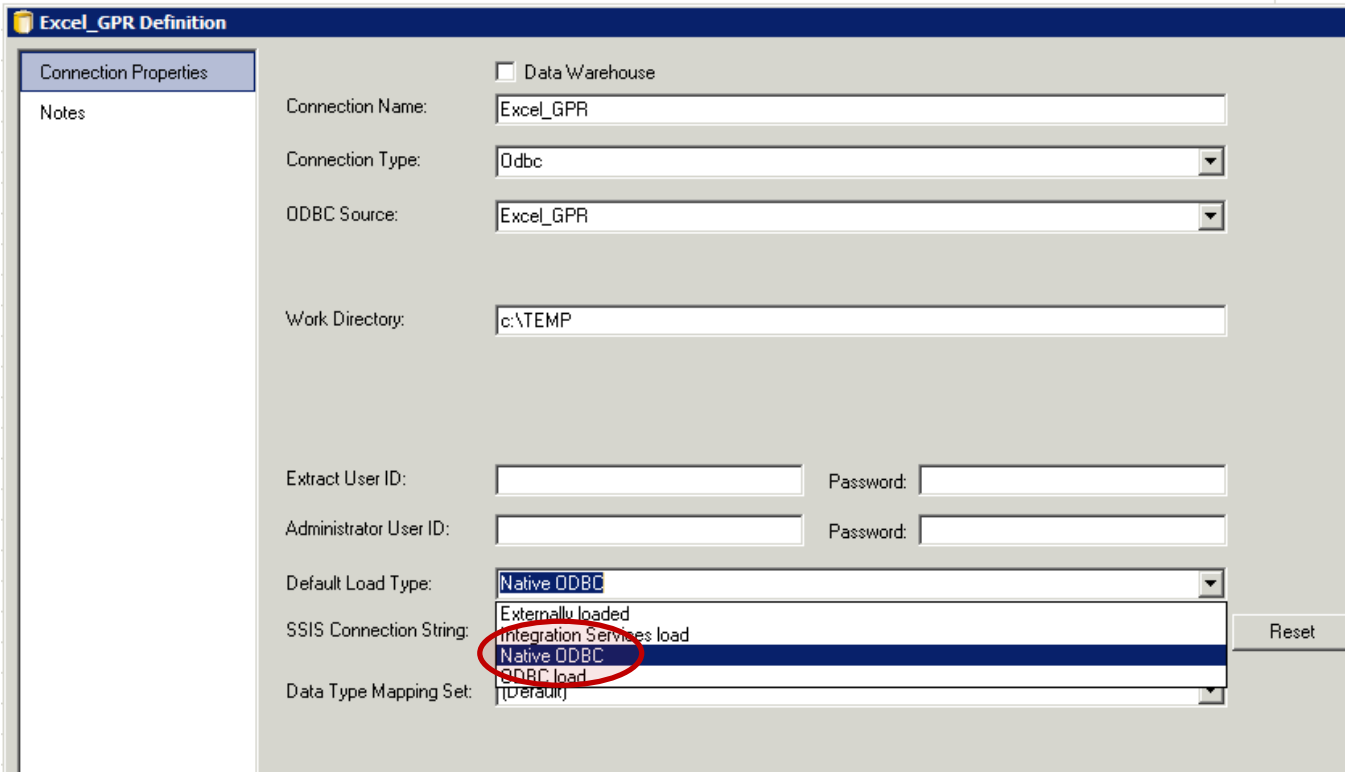
Below are few cases on how to configure loading of the Excel data into RED load_% tables.

1. ODBC-based Load

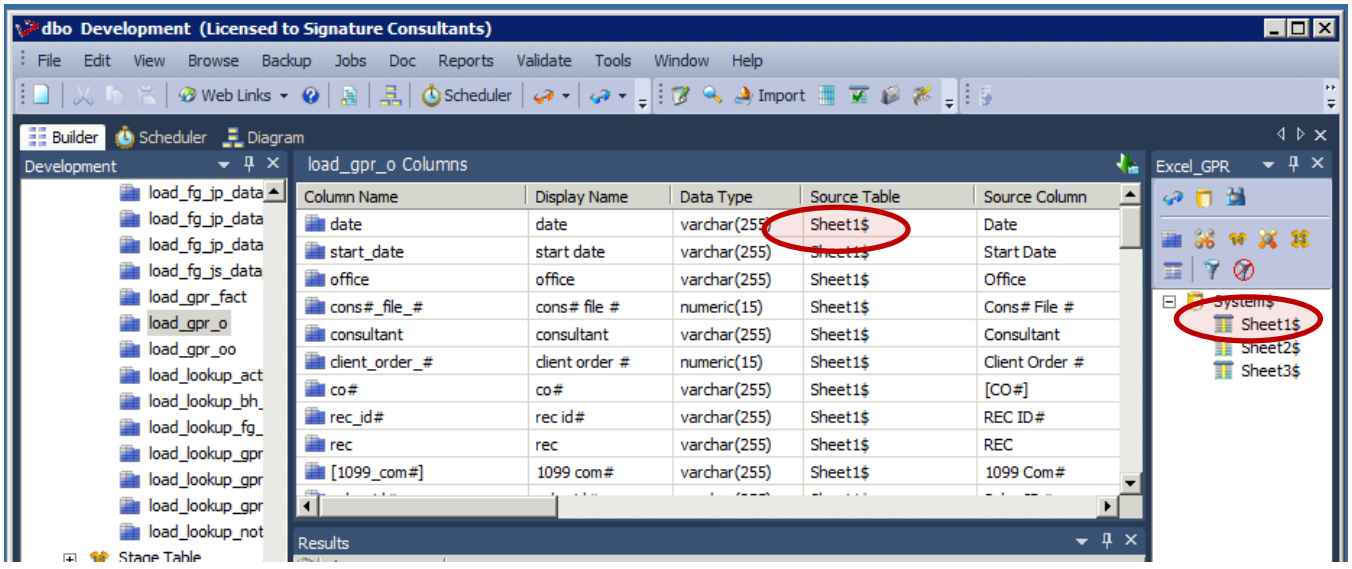
Consider Excel file -- a workbook gpr.xlsx -- that consists of multiple worksheets (Sheet1, Sheet2, and Sheet3). The first method we consider is ODBC-based Load. First what we need to do is to create ODBC connection using, for instance, Microsoft Excel Driver.



In the RED connection definitions choose “Native ODBC” option. When applicable, Native ODBC is a very fast option since it is using Bulk Insert in SQL Server, SQL*LOADER in Oracle or LOAD statement in DB2. But, because it is processing the whole intermediate file at once, the data has to be relatively “clean”: error is data cause the whole load to fail.



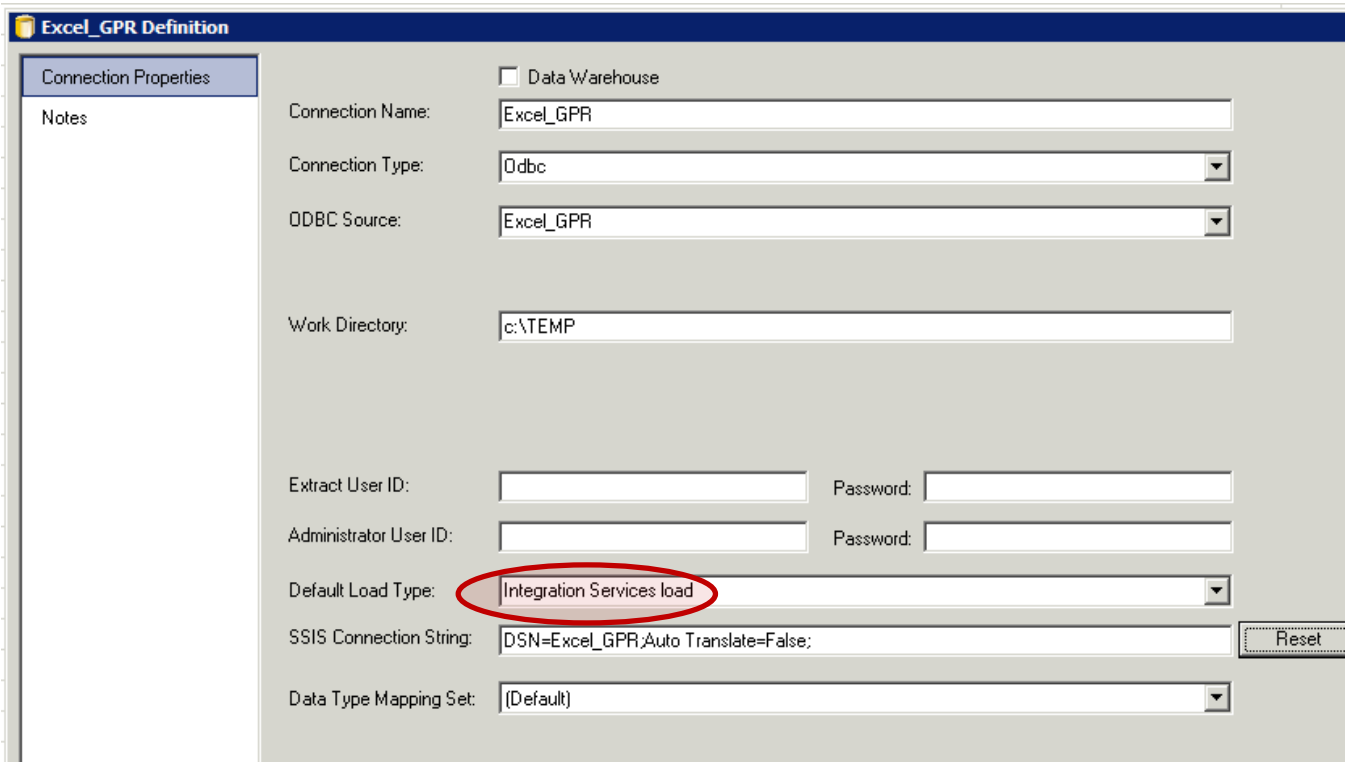
The load table will look somewhat like this, where an individual Sheet1$ is used as a source. The white spaces in columns name have to be resolved by using square brackets around the names.



When Integration Services Load option is chosen then the SSIS Connection String will be filled in once Reset is pressed. SSIS Connect String is a valid SSIS entry that can be used to connect to the data source or destination. The Reset button will attempt to construct a valid connection string from the connection information supplied in the Connection's details. This string consists of the Database ID, Database Link ID (Instance name), Provider Name, Extract User details and more. The string's parameters can be customized. WhereScape will generate a SSIS package with all the functionality provided by Microsoft software stock.

Example of a SSIS Connect String:

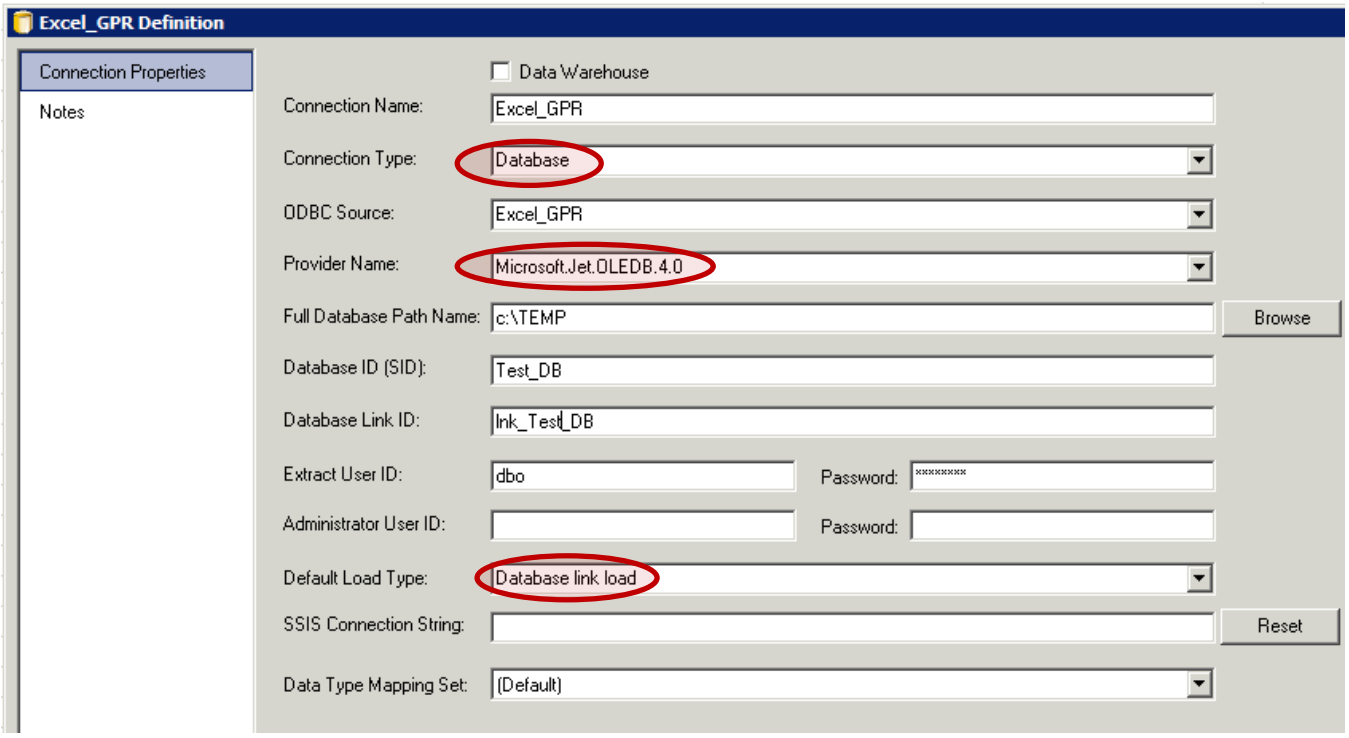
```
Provider=OraOLEDB.Oracle.1;Data Source=SOURCE;
Persist Security Info=True;Password=PASSWD;User ID=USER;Auto Translate=False;
```



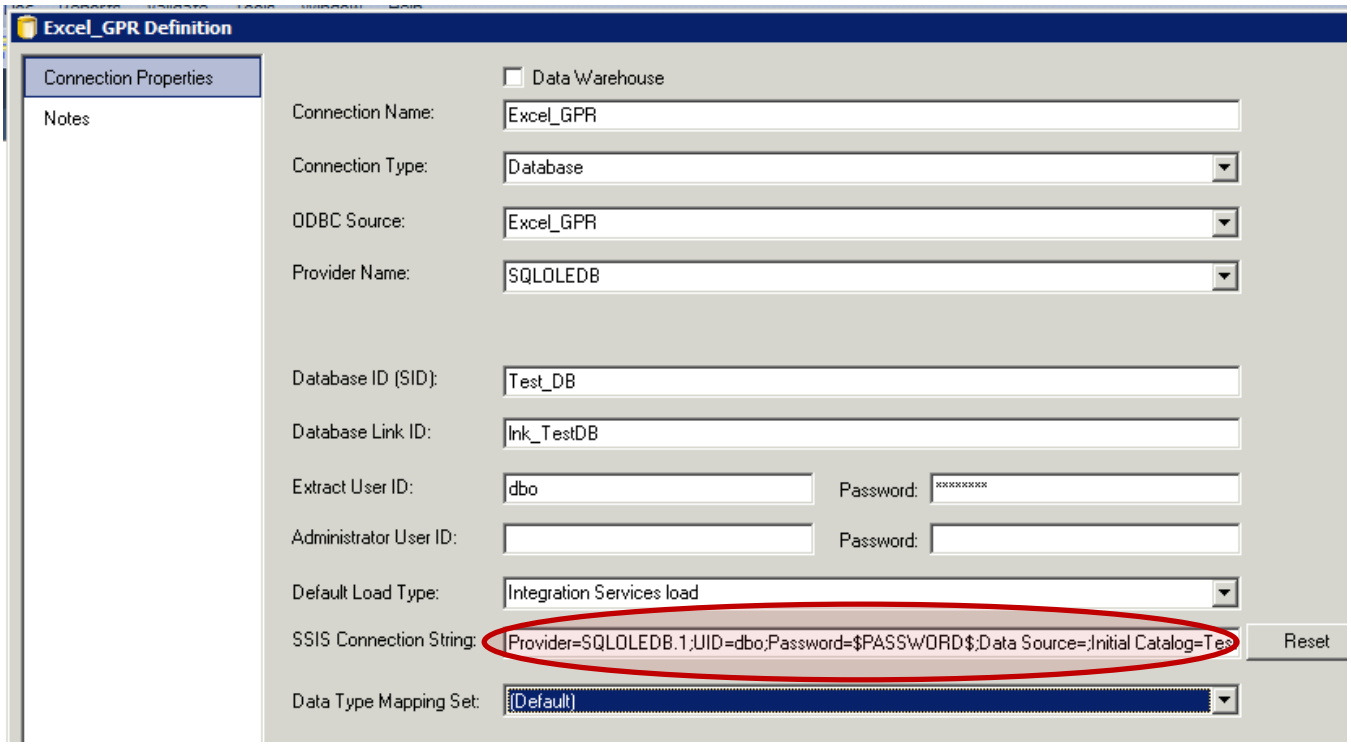
If "ODBC load" is used, then load will be done on row-by-row basis (ODBC Load - Single Row Inserts). This is the most "forgiving", yet the slowest method, which allows to partial or sample loading

2. RDBMS-based Load

This is very common to use DB-based loading method with Excel loading because it provides connectivity via database links, a very effective and fast technique. The following screen illustrates the Database connection; Excel file can be loaded using a database link loading through a linked server with use of Microsoft.Jet.OLEDB.4.0 provider.

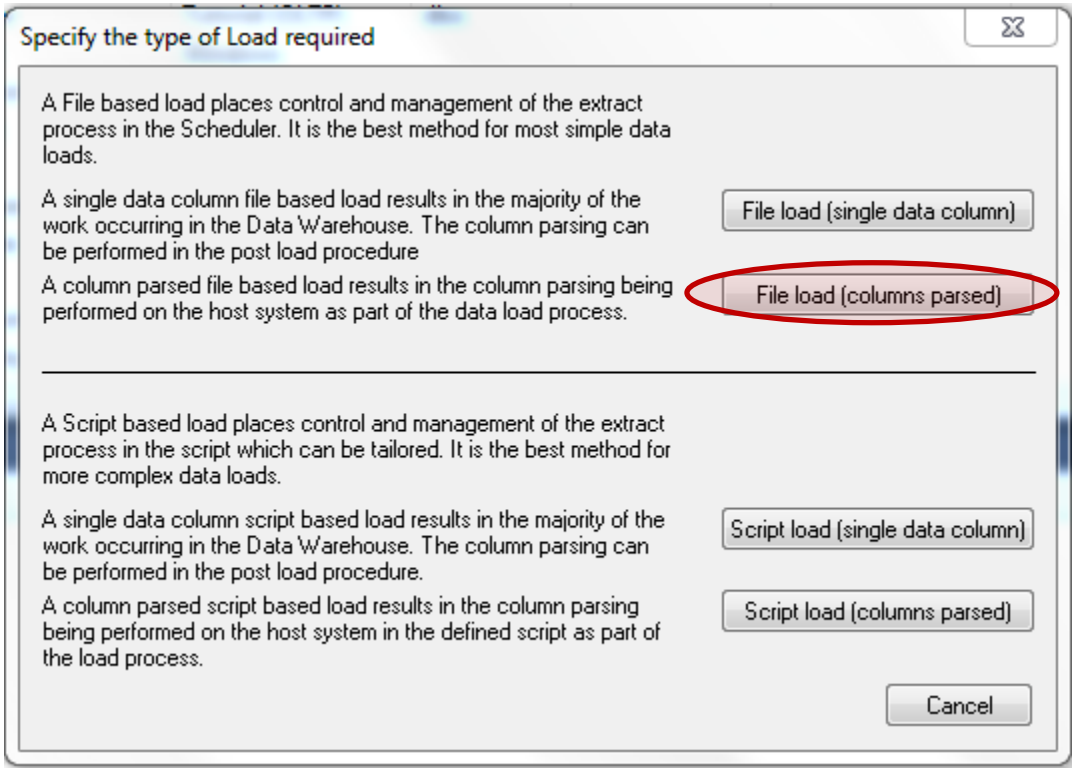


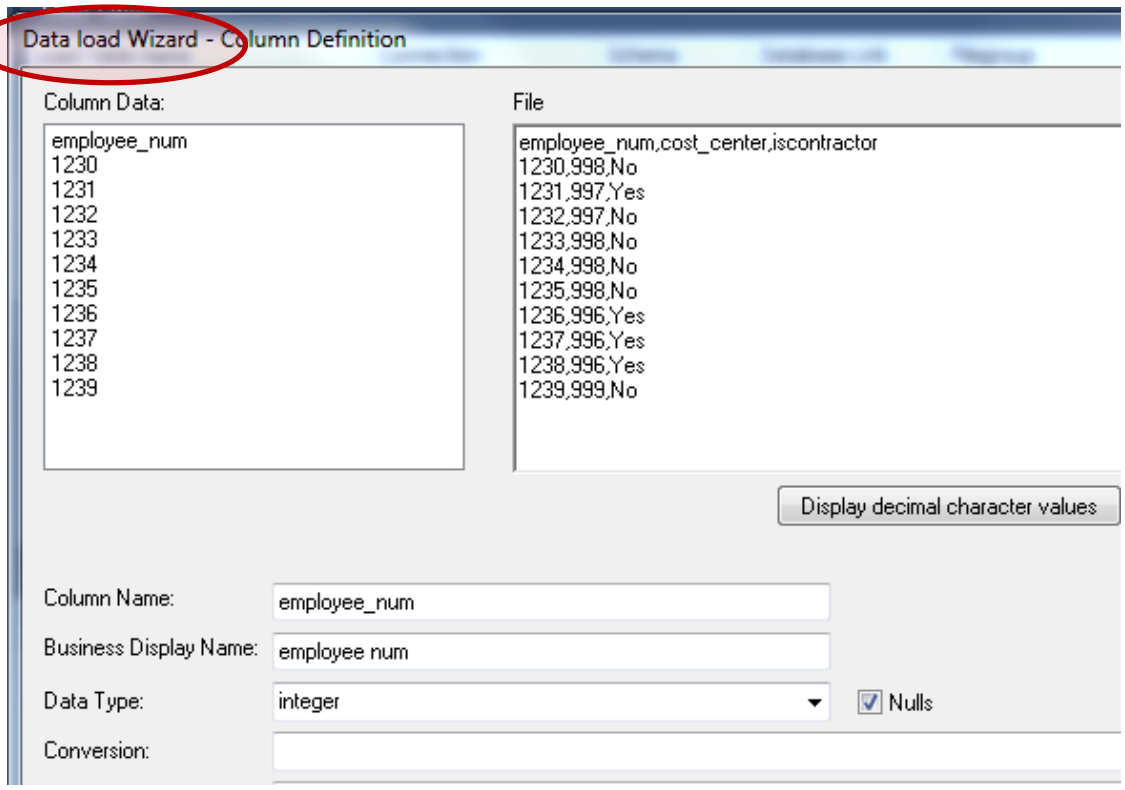
Similarly to the the ODBC-based load, the Database connection type can too have an option of “Integration Services Load”. Note the use of parameters in the SSIS Connection String (more on parameterized load further).



3. File-based Load

To load data from CSV file using this technology, we will use a “Windows” connection that is provided by Wherescape out-of-the-box. When the option “File Load (columns parsed)” is chosen, Wherescape will guide you through the wizard of creating columns, providing names and data types as well as option for data conversion.



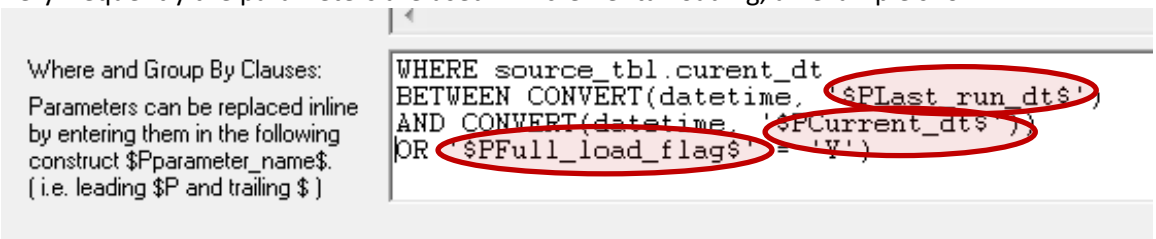


The Script based load provides endless possibilities to apply various business rules and programming logic while loading Excel file into load table. Those business logics may include removing/reformatting embedded text or images from a spreadsheet; looping through multiple sheets or multiple workbooks while loading data; using wild cards and many more…

4. Parameterized Approach to an Excel Loading

Parameters are a means of passing information between two or more procedures and/or between the WhereScape RED environment and procedures. They can be easily edited within the WhereScape RED framework. Parameters are widely used for loading in general and Excel -- in particular.

Very frequently the parameters are used in incremental loading, an example shown




```
Parameters:
Current_Dt = Timestamp of current business date, updates programmatically
Last_Run_Dt = Timestamp of last run date, updates programmatically
Full_load_flag = Indicates Full or incremental load
```

Another use for parameters in loading process:

```
Parameter: FEED_DIRECTORY=Feed directory that varies depending on environment
```

Below is a sample code that illustrates WhereScape's ability to utilize parameters to dynamically load multiple Excel workbooks with multiple tabs.

```
BEGIN
-- Select from reference table that holds info about the Excel files designated for loading

SELECT @v_count=count(*)
FROM XLS_PROC_REF WHERE ISNULL(xl_status,'I')='A'

IF @v_count=0

BEGIN
  SET @v_msgtext='There are no excel files to load.'
END
  ELSE

BEGIN
WHILE @v_count <> 0

BEGIN

    -- Set the lock if wish. Execute wherescape callable procedure that allow to write a parameter

EXEC dbo.WsParameterWrite 'processlockflag','Y','Set to Y if a load process is running.'

     -- Get the first workbook to process
     -- Some parameters used in the code below:
          -- XL_FILE=Excel File (workbook) name
          -- XL_DIR=Directory where Excel file located
          -- XL_SHEET=Excel Tab (worksheet) name, e.g. ["1101$"]
          -- XL_USER=Username to be used for extraction Excel Files
          -- XL_PWORD=Password to be used for extraction Excel Files

SELECT top 1 @v_xl_dir=xl_dir,
@v_xl_file=REPLACE(xl_dir  COLLATE Latin1_General_BIN,
            '\\fccokfil001\fccdata\accounting\Common\ACCTG\', 'Y:\ACCTG\' ),
@v_xl_sheet=xl_sheet, @v_xl_user=xl_user, @v_xl_pword=xl_pword, @v_fc_sub=fc_sub
FROM XLS_PROC_REF WHERE ISNULL(xl_status,'I')='A'

     -- Build the sql statement and execute it, using OPENDATASOURCE method
     -- This sql statement varies depending on XL_FILE type, i.e. Sub
     -- LOGIC as follows:

IF @v_fc_sub ='SUB4'

BEGIN
SET @v_sql='INSERT INTO load_COLL_ACCT SELECT '
SET @v_sql+= @v_xl_dir+','+@v_xl_file+','+@v_xl_sheet+','+@v_xl_user+','+@v_xl_pword
SET @v_sql+= ',F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,'+CONVERT(varchar,@v_dss_update_time, 9)
SET @v_sql+= ' FROM OPENDATASOURCE(''Microsoft.ACE.OLEDB.12.0'',
''Data Source=@v_xl_dir +''\''+ @v_xl_file
SET @v_sql+= char(59)+'user id=@v_xl_user'
SET @v_sql+= char(59)+'password=@v_xl_pword'
SET @v_sql+= char(59)+'Extended Properties=Excel 12.0'')'
SET @v_sql+= '... [''']+@v_xl_sheet+'$''']'
END

ELSE IF @v_fc_sub ='SUB5'
BEGIN
```

```
SET @v_sql='INSERT INTO load_COLL_ACCT SELECT '
SET @v_sql+= @v_xl_dir+','+@v_xl_file+','+@v_xl_sheet+','+@v_xl_user+','+@v_xl_pword
SET @v_sql+= ',F1,F2,F3,F4,F5,F6,'+CONVERT(varchar, @v_dss_update_time, 9)
SET @v_sql+= ' FROM OPENDATASOURCE(''Microsoft.ACE.OLEDB.12.0'',
             ''Data Source=c:\testing\'+@v_xl_file
SET @v_sql+= char(59)+'user id=@v_xl_user'
SET @v_sql+= char(59)+'password=@v_xl_pword'
SET @v_sql+= char(59)+'Extended Properties=Excel 12.0'')'
SET @v_sql+= '... [''']+@v_xl_sheet+'$''']'
END

exec sp_executesql @v_sql

UPDATE XLS_PROC_REF
SET xl_status='I'
WHERE xl_dir=@v_xl_dir AND xl_file=@v_xl_file AND xl_sheet=@v_xl_sheet
               AND xl_status='A' AND fc_sub=@v_fc_sub

SELECT @v_count=count(*)
FROM XLS_PROC_REF WHERE ISNULL(xl_status,'I')='A'
END
END
```